DESIGN AND TEST OF DS89s52 BASED TRAINER BOARD

A dissertation submitted in partial fulfillment of requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering

ISLAMIC UNIVERSITY OF TECHNOLOGY
The Organization of Islamic Co-operation (OIC)

SUBMITTED BY

Md.Irfanul Quader
Tanzeeb Zaman
Yazdani Ul Islam

UNDER THE SUPERVISION OF

GOLAM SAROWAR
ASSISTANT PROFESSOR

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGNIEERING
ISLAMIC UNIVERSITY OF TECHNOLOGY, BANGLADESH.

DESIGN AND TEST OF DS89s52 BASED TRAINER BOARD

A Project Paper Presented To
The Academic Faculty

By

Md.Irfanul Quader (Student ID:082420)

Tanzeeb Zaman (Student ID:082426)

Yazdani Ul Islam (Student ID:082440)

In Partial Fulfillment of Requirement for the Degree of
Bachelor of Science in Electrical and Electronics Engineering

ISLAMIC UNIVERSITY OF TECHNOLOGY

October, 2012

Add the loose page:

ACKNOWLEDGEMENT

We would like to thank our supervisor **GOLAM SAROWAR** for his helpful directions, suggestions and corrections. Without his proper guidance, we do not believe this project would have been a successful one. We would like to express our heartiest gratitude to him for his tremendous support for us to understand the basics and construct the hardware for acquisition of the project .We also express our thankfulness to **Prof. Dr. Md. Shahid Ullah,** head of the department of Electrical and Electronic Engineering for providing us with best facilities in the department and his timely contributions.

Last but not the least we would like to thank all of our friends, senior and junior brothers were involved directly or indirectly in successful completion of this project.

- Md.Irfanul Quader
- Tanzeeb Zaman
- Yazdani Ul Islam

Objective:

The objective of this project is to design an AT89s52 microcontroller based trainer kit with the provisions for Port access, Serial and Parallel Interfacing with the PC. An onboard ADC is also provided.

Abstract:

The main difference between a microprocessor and microcontroller is that a Microcontroller has inbuilt peripherals like Timers, USART, Interrupt controller etc whereas in a Microprocessor these have to be interfaced as separate ICs. In this project we use the Microcontroller AT89s52 from the ATMEL corp. The AT89s52 is a low power; high performance CMOS 8-bit microcontroller with 4Kbytes of Flash programmable and erasable read only memory (EPROM) .The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51 instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on monolithic chip, the Atmel At89s52 is a powerful microcomputer which provides a highly-flexible and cost effective solution to many embedded control applications.

Table of Contents:

CHAPTER **Page**

DESIGN AND TEST OF DS89s52 BASED TRAINER

INTRODUCTION

Description of AT89s52 Microcontroller:

The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer.

By combining a versatile 8-bit CPU with in-system programmable flash on

a monolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, And clock circuitry. In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes.

The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

In systems based on an AT89s51/52 type microcontroller, we need a ROM burner to burn your program into the microcontroller. For the AT89s52, the ROM burner can erase the flash ROM in addition to burning a program into it. Since AT89s52 has flash ROM it doesn't required an EPROM.

Figure 1 shows the minimum connection for the 89s52 based system. It is notifying that "EA=V_{cc}" indicates that an 89s52 has on-chip ROM for the program. Again P_0 connection to pull-up resistors to ensure the availability of P_0 for the I/O operations. We also need to use a momentary switch RESET.

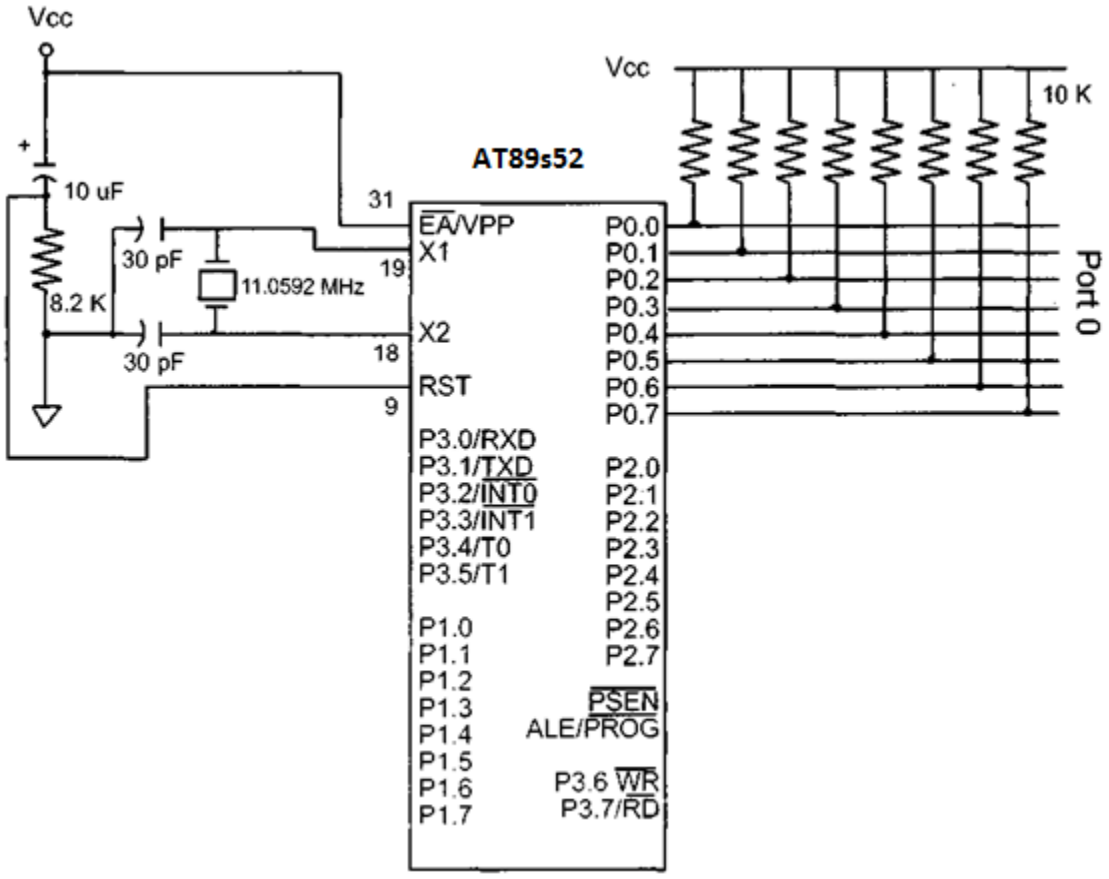


Figure: 1

The 89s52 chip from Maxim/Dallas semiconductor is an 8051 type microcontroller with on-chip flash ROM. It also has a built in loader allowing it to download programs into the chip via the serial port, therefore eliminating any need for an external ROM burner. The important feature makes the DS89s52 chip an ideal candidate for 8051-based home development systems.

Key features of the DS89s52

The important key features of DS89s52 are as follows:

1.80C52 compatible
- (a)8051 pin- and instruction-set compatible
- (b)Four bidirectional I/O ports
- (c)Three 16-bit timer counters
- (d) 256 bytes scratched RAM

2. On-chip flash memory
- (a) 16KB for DS89s52
- (b) 32KB for DS89s54
- (c) 64KB for DS89s55

3. In-system programmable through the serial port
- 1KB SRAM for MOVX

4. ROMSIZE feature
- (a) Selects internal program memory size from 0 to 64K
- (b) Allows access to entire external memory map
- (c) Dynamically adjustable by software

5. High-speed architecture
- (a) 1 clock per machine cycle
- (b) DC to 33MHz operation
- (c) Single-cycle instruction in 30 ns
- (d) Optional variable length MOVX to access fast/slow peripherals

6. Two full-duplex serial ports

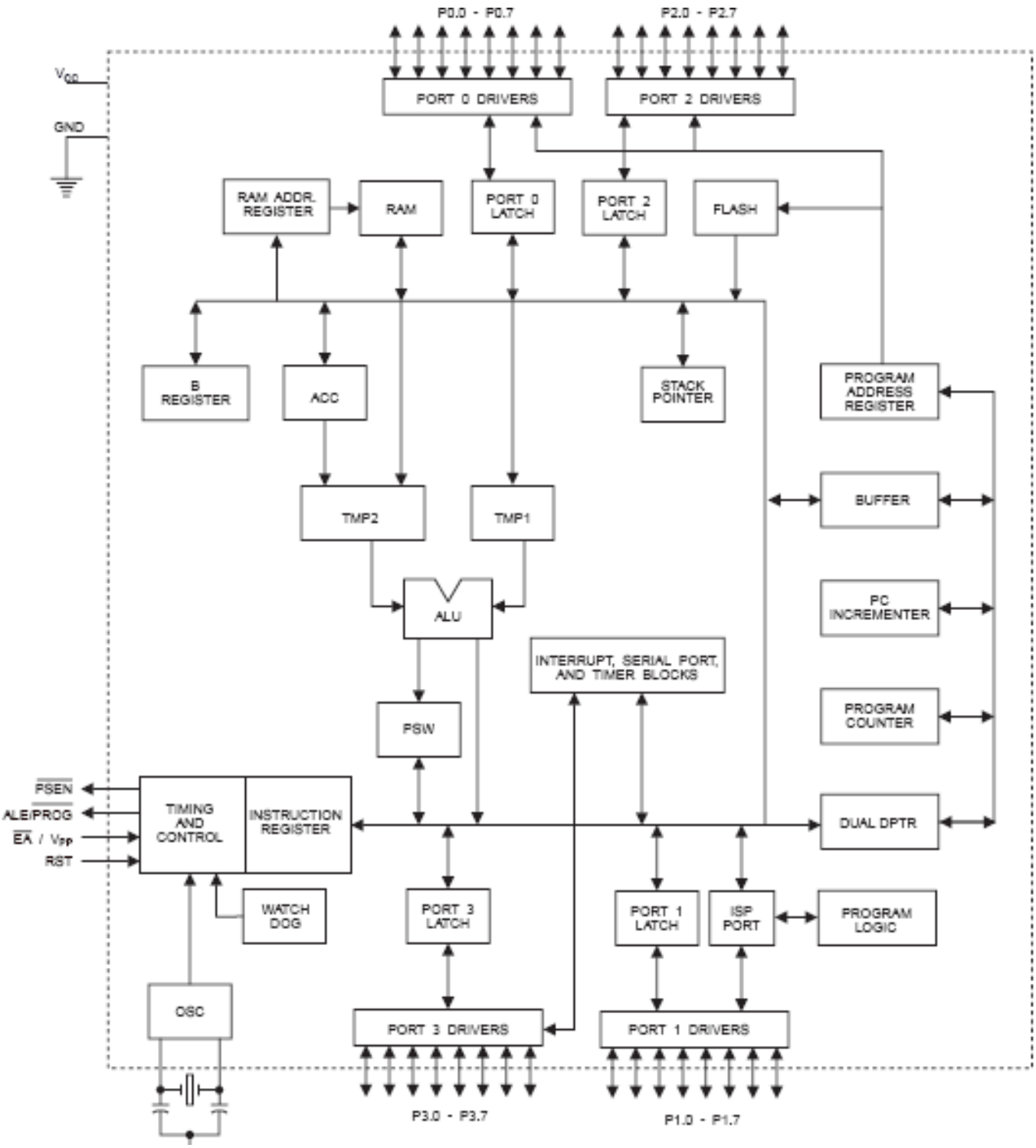
7. Programmable watchdog timer

8. 13 interrupt sources (six external)

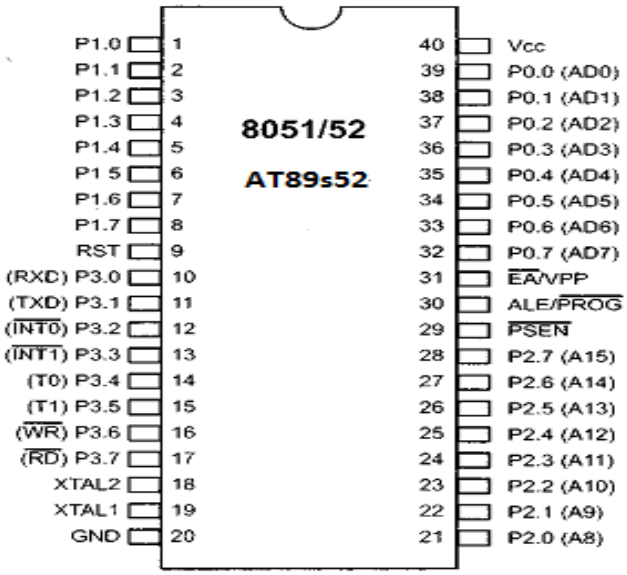
9. Five levels of interrupt priority

10. Power-fail reset & early warning power-fail interrupts

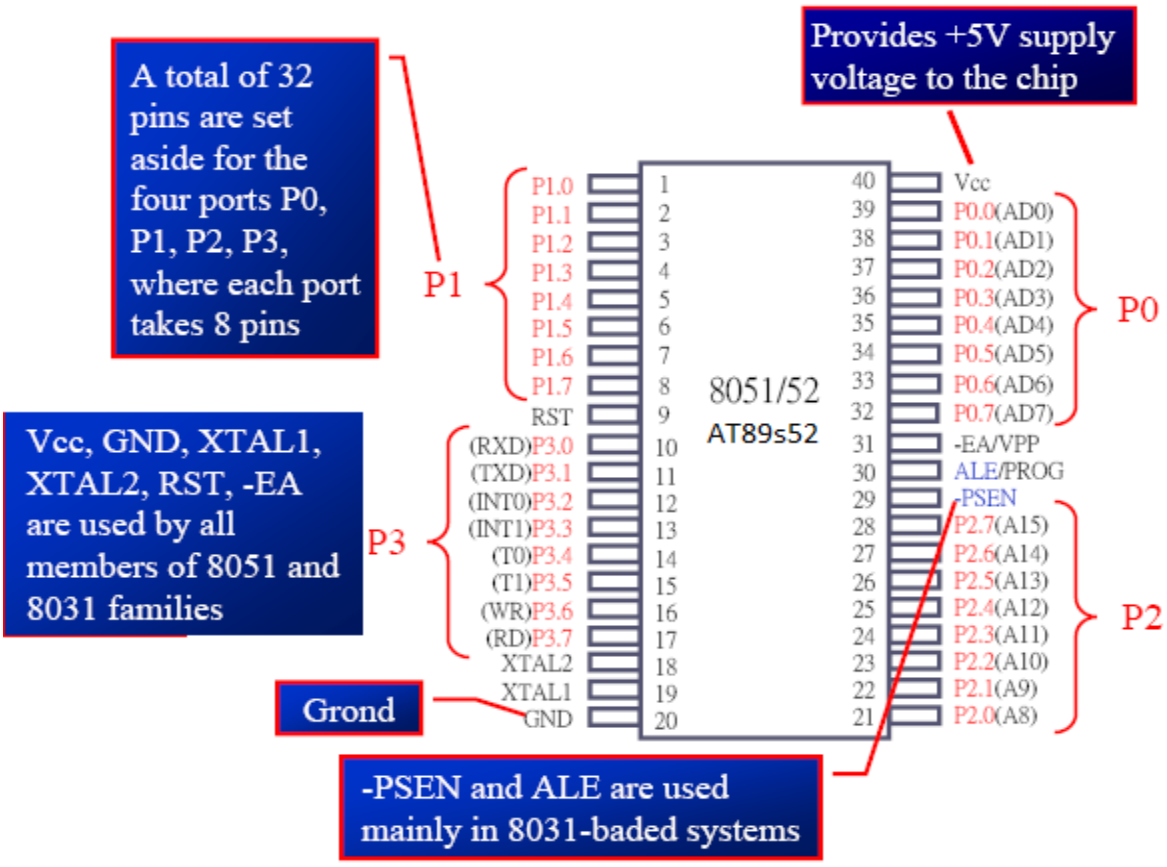
Block Diagram



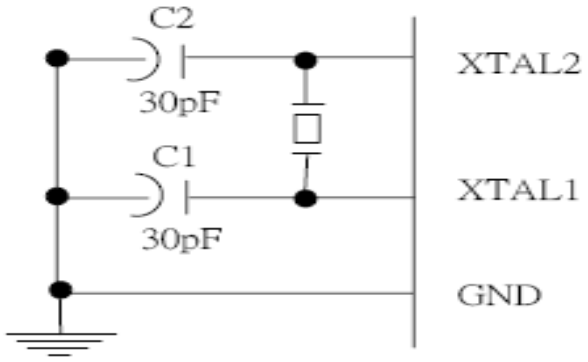
Pin Diagram of 8051



Pin Description of 8051:



The 8051 has an on-chip oscillator but requires an external clock to run it. A quartz crystal oscillator is connected to inputs **XTAL1** (pin19) and **XTAL2** (pin18). The quartz crystal oscillator also needs two capacitors of 30 pF value.



RESET CIRCUIT

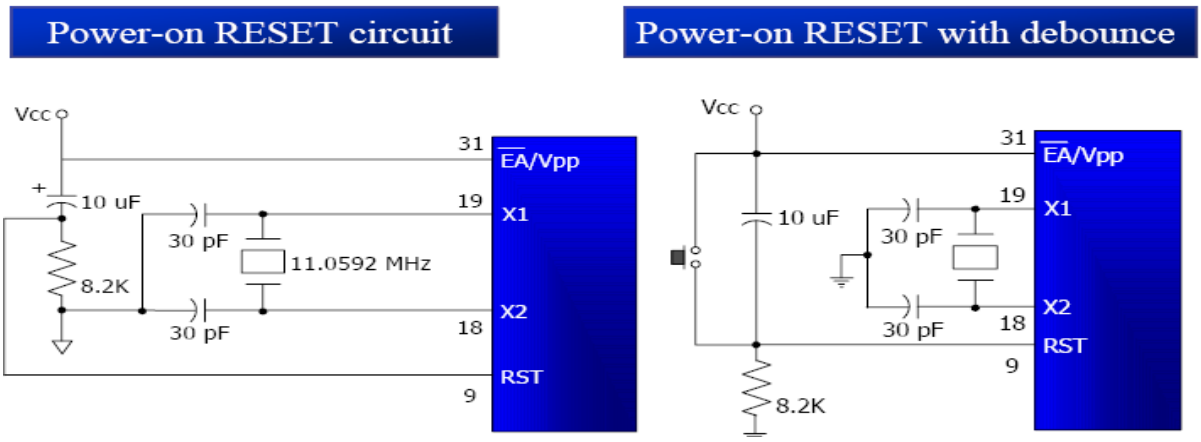
RESET pin is an input and is active high (normally low). Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities. This is often referred to as a power-on reset. Activating a power-on reset will cause all values in the registers to be lost.

RESET value of some 8051 registers

we must place the first line of source code in ROM location 0

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

In order for the RESET input to be effective, it must have a minimum duration of 2 machine cycles. In other words, the high pulse must be high for a minimum of 2 machine cycles before it is allowed to go low.



DS89s52 trainer connection

We selected the DS89s52 for an 8051 trainer because it is inexpensive but powerful and can easily wire-wrap it to be used at work and home. The connection for the DS89s52 Trainer is shown in the Figure-2

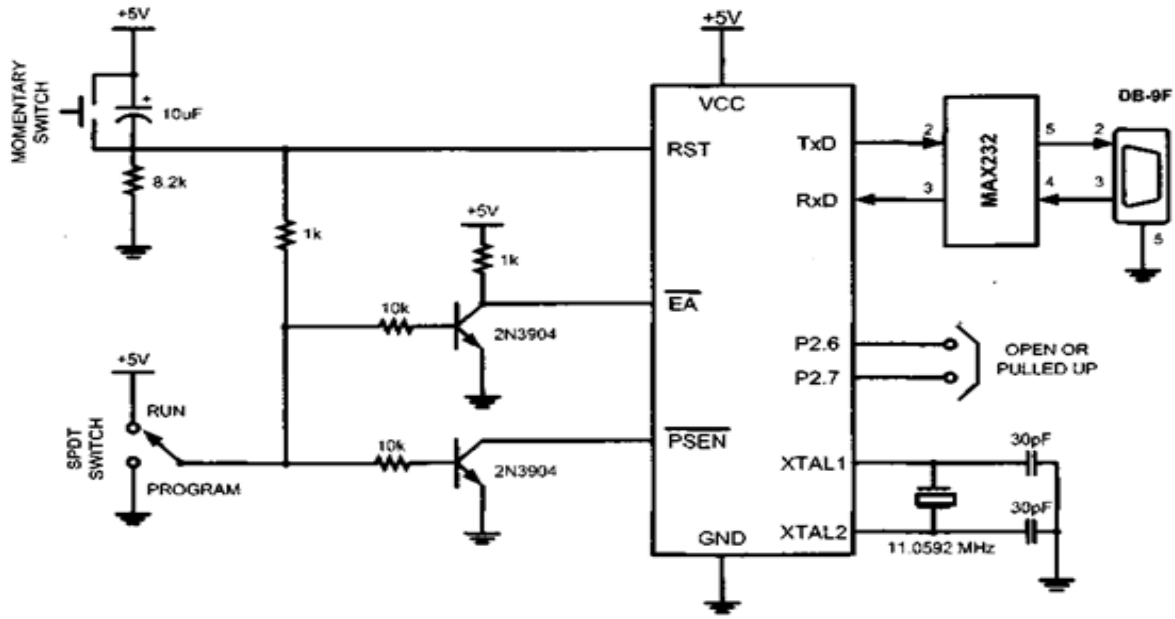


Figure: 2

Using the DS89s52 for development is more advantageous than using the 8751 or 89s51 system for the following two major reasons:

1. Using the Ds89s52 for an 8051 microcontroller allows us to program the chip without any need for a ROM burner, Because not everyone has access to a ROM burner, the DS89s52 is that an ideal home-development system. The advantage of the DS89s52 is that it can be programmed via the COM port of a PC(x86 IBM or compatible PC) while it is in the system. Contrast this with the 89s52 system in which we have to remove the chip, program it, and install it back in the system every time we want to change the program contents of the on-chip ROM. This results in a much longer development time for the 89C51 compared with the DS89s52 system.

2. The two serial ports on the DS89s52 allow us to use one for PC interfacing with the chip, and the other for the data acquisition.

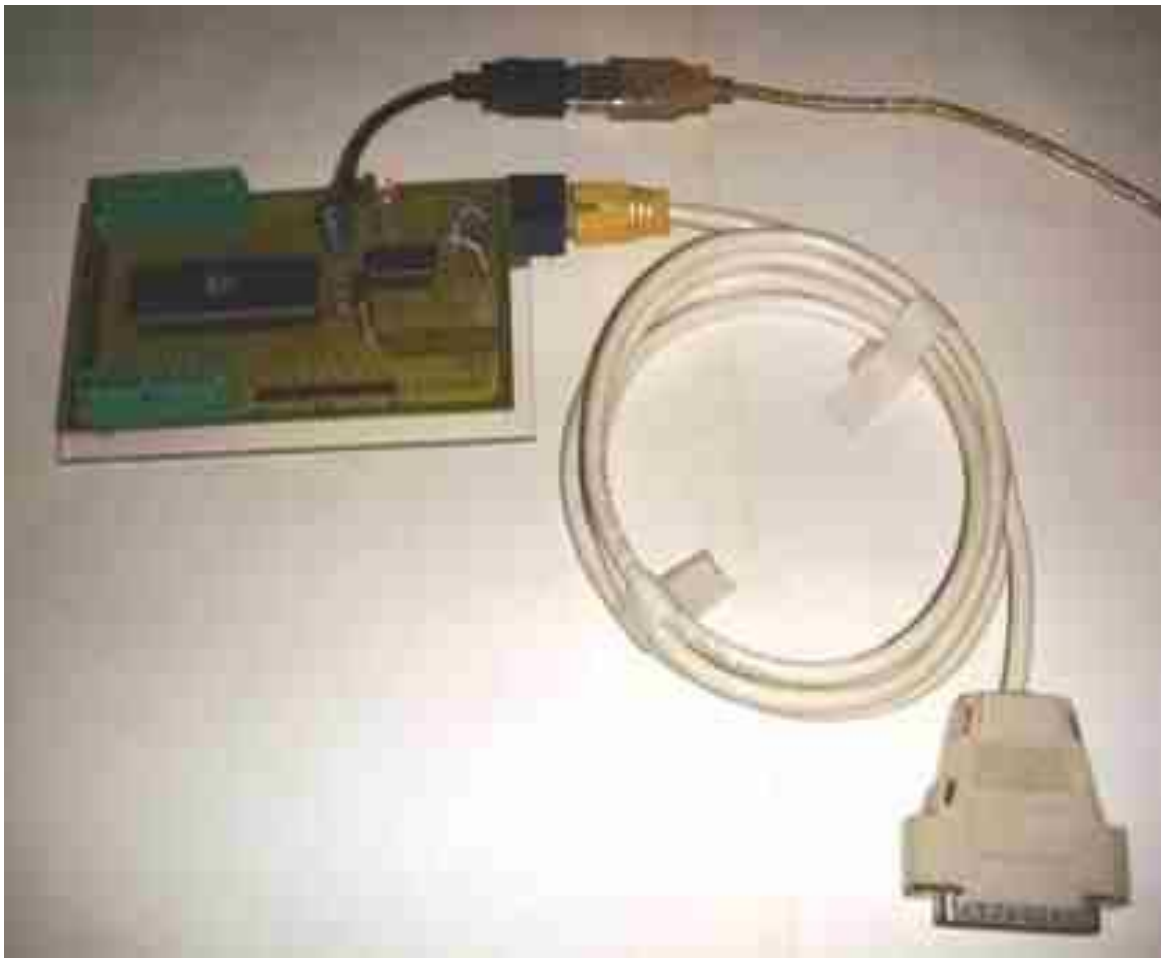
Notice from the figure 2 that the reset circuitry and serial port connection are the same as in any 8051-based system. However, the extra circuitry needed for programming is two transistors, a switch, and 10K and 1K-ohm resistors. In fact, we can add these components to our 8751 system and use it as a DS89s52 system by plugging a DS89s52 chip in the socket. The switch allows us to select between the programs and run options. We can load our program into the DS89s52 by setting the switch to $V_{cc,}$ and run the program by setting it to GND.

Figure 2 shows the connection for the 8051 Trainer. The Trainer has both of the serial ports connected and accessible via two DB-9 connectors. It also has 8 LEDs and 8 switches along with the P0-P3 ports, all of which are accessible via terminal blocks. It also comes with an on-board power regulator.

Circuit Components

- ATMEL MICROCONTROLLER : AT89S52
- SERIAL PORT CABLE
- MAX232
- Capacitor: 1~22uF 16V & 10uf 50V
- Resistor: 10k,1k, 230 ohm, 330 ohm
- Crystal Oscillator: 12 Mhz 11.0592 Mhz
- Seven Segment Display(cc)
- LED
- LCD (16*2)

Hardware View



Communication System of DS89s52 trainer

SERIAL COMMUNICATION

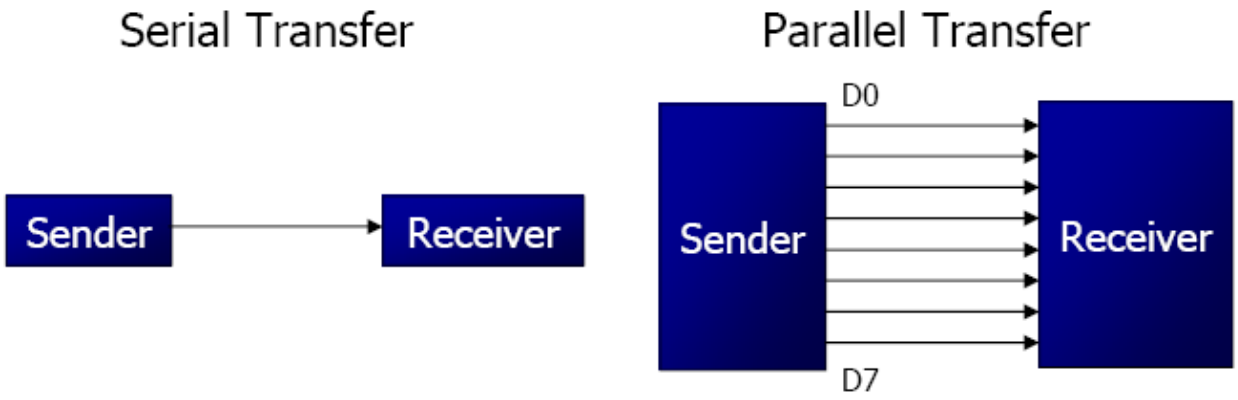
Computers transfer data in two ways:

1. **Parallel**

- Often 8 or more lines (wire conductors) are used to transfer data to a device that is only a few feet away.

2. **Serial**

- To transfer to a device located many meters away, the serial method is used.
- The data is sent one bit at a time.



- At the transmitting end, the byte of data must be converted to serial bits using parallel-in-serial-out shift register.
- At the receiving end, there is a serial in-parallel-out shift register to receive the serial data and pack them into byte.
- When the distance is short, the digital signal can be transferred as it is on a simple wire and requires no modulation.
- If data is to be transferred on the telephone line, it must be converted from 0s and 1s to audio tones.

- This conversion is performed by a device called a modem, "Modulator/demodulator".

Serial data communication uses two methods

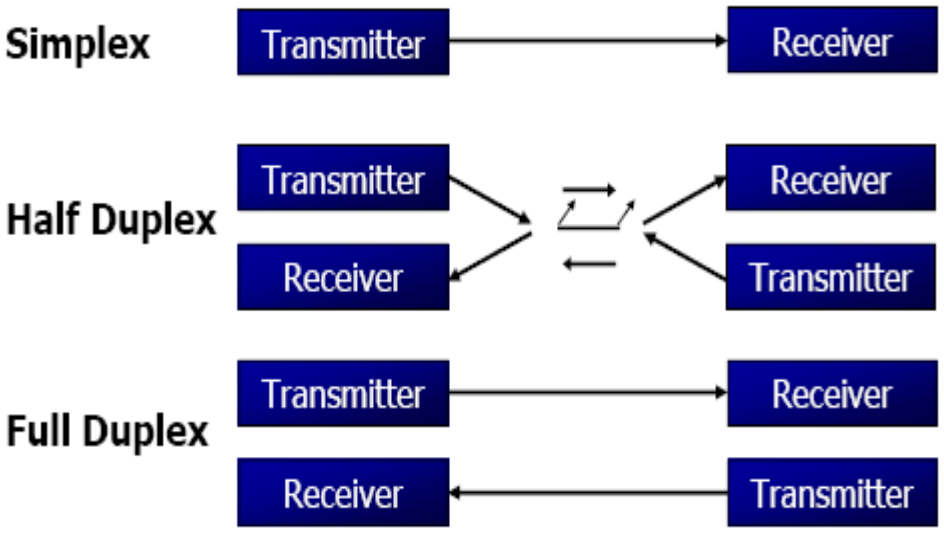
1. Synchronous method transfers a block of data at a time.
2. Asynchronous method transfers a single byte at a time.

It is possible to write software to use either of these methods, but the programs can be tedious and long.

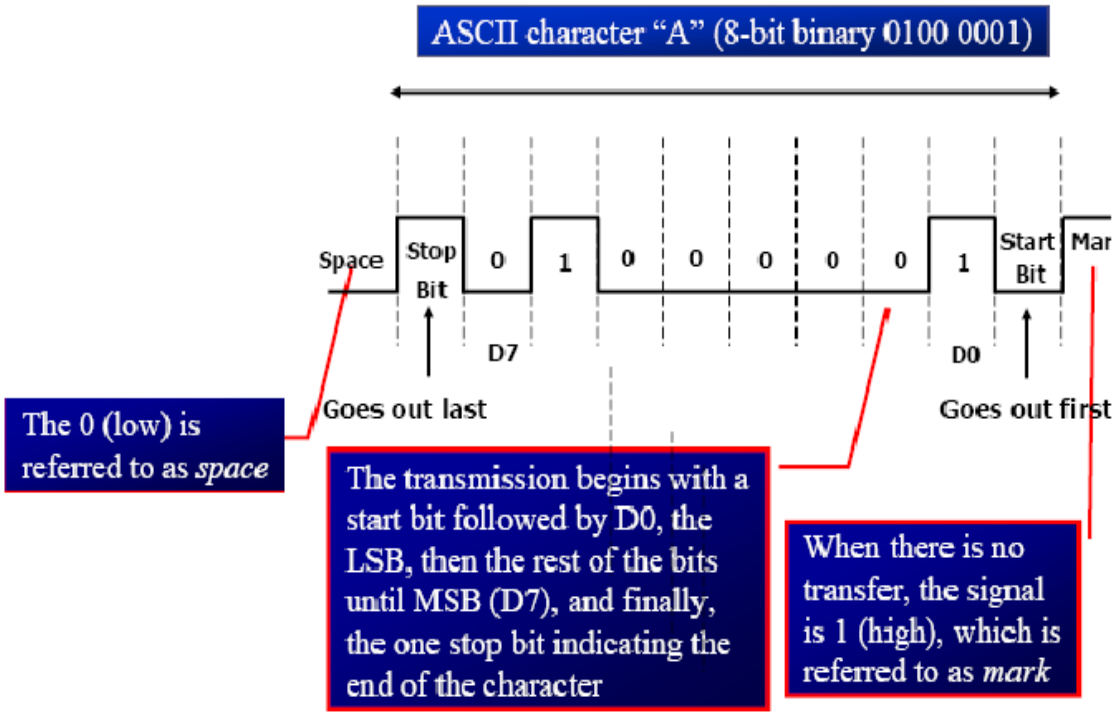
- There are special IC chips made by many manufacturers for serial communications
- UART (universal asynchronous Receiver-transmitter)
- USART (universal synchronous-asynchronous Receiver-transmitter)

Ways of Data Transmission

- If data can be transmitted and received, it is a ***duplex transmission***
- If data transmitted one way a time, it is referred to as ***half duplex***
- If data can go both ways at a time, it is ***full duplex***
  This is contrast to simplex transmission



- A ***protocol*** is a set of rules agreed by both the sender and receiver on
  - How the data is packed
  - How many bits constitute a character
  - When the data begins and ends
- ***Asynchronous*** serial data communication is widely used for character-oriented transmissions
  - Each character is placed in between start and stop bits, this is called framing.
  - Block-oriented data transfers use the synchronous method.
- The start bit is always one bit, but the stop bit can be one or two bits.
- The start bit is always a 0 (low) and the stop bit(s) is 1 (high).



Due to the extended ASCII characters,8-bit ASCII data is common. In older systems, ASCII characters were 7-bit. In modern PCs the use of one stop bit is standard. In older systems, due to the slowness of the receiving mechanical

device, two stop bits were used to give the device sufficient time to organize itself before transmission of the next byte.

The rate of data transfer in serial data communication is stated in bps (bits per second). Another widely used terminology for bps is baud rate. It is modem terminology and is defined as the number of signal changes per second. In modems, there are occasions when a single change of signal transfers several bits of data. As far as the conductor wire is concerned, the baud rate and bps are the same, and we use the terms interchangeably.

The data transfer rate of given computer system depends on communication ports incorporated into that system.

- IBM PC/XT could transfer data at the rate of 100 to 9600 bps.
- Pentium-based PCs transfer data at rates as high as 56K bps.
- In asynchronous serial data communication, the baud rate is limited to 100K bps.

RS232 Pins

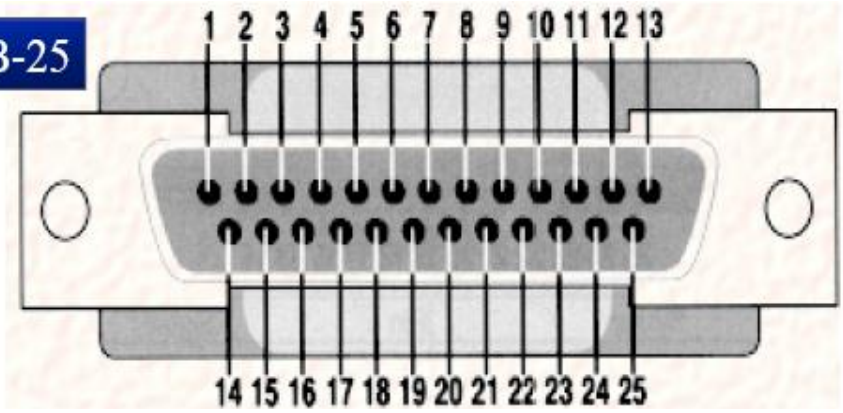
An interfacing standard RS232 was set by the Electronics Industries Association (EIA) in 1960.The standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible. In RS232, a 1 is represented by -3 ~ -25 V, while a 0 bit is +3 ~ +25 V, making -3 to +3 undefined. The table provides the pins and their labels for the RS232 cable.

RS232 DB-25 Pins

Pin	Description	Pin	Description
1	Protective ground	14	Secondary transmitted data
2	Transmitted data (TxD)	15	Transmitted signal element timing
3	Received data (RxD)	16	Secondary receive data
4	Request to send (-RTS)	17	Receive signal element timing
5	Clear to send (-CTS)	18	Unassigned
6	Data set ready (-DSR)	19	Secondary receive data
7	Signal ground (GND)	20	Data terminal ready (-DTR)
8	Data carrier detect (-DCD)	21	Signal quality detector
9/10	Reserved for data testing	22	Ring indicator (RI)
11	Unassigned	23	Data signal rate select
12	Secondary data carrier detect	24	Transmit signal element timing
13	Secondary clear to send	25	Unassigned

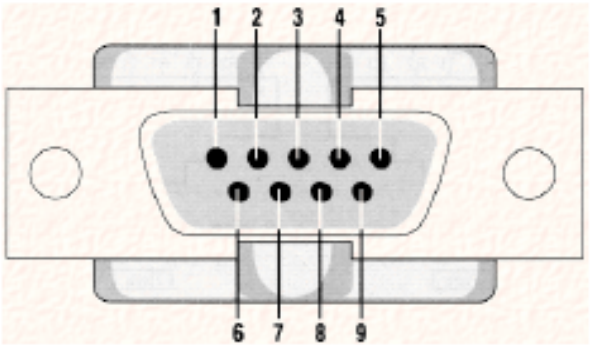
Table: RS232 pins (DB-25)

RS232 Connector DB-25



Since not all pins are used in PC cables, IBM introduced the DB-9 version of the serial I/O standard.

RS232 Connector DB-9



RS232 DB-9 Pins

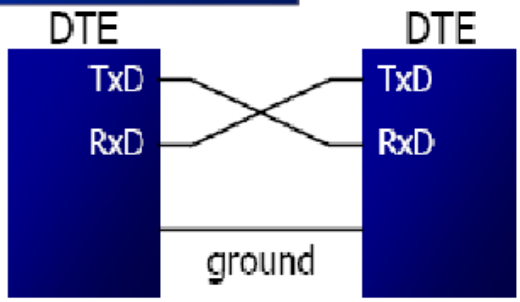
Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

Current terminology classifies data communication equipment as

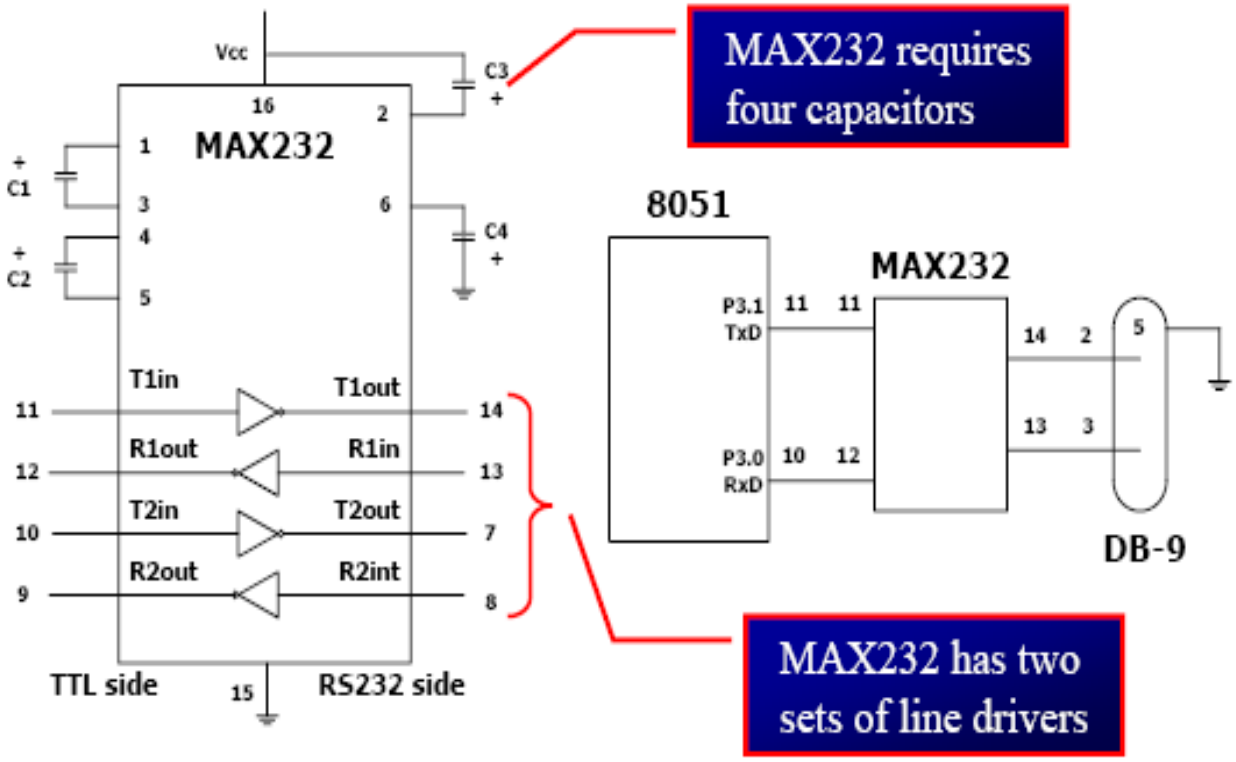
- DTE (data terminal equipment) refers to terminal and computers that send and receive data
- DCE (data communication equipment)refers to communication equipment, such as modems

The simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground.

Null modem connection

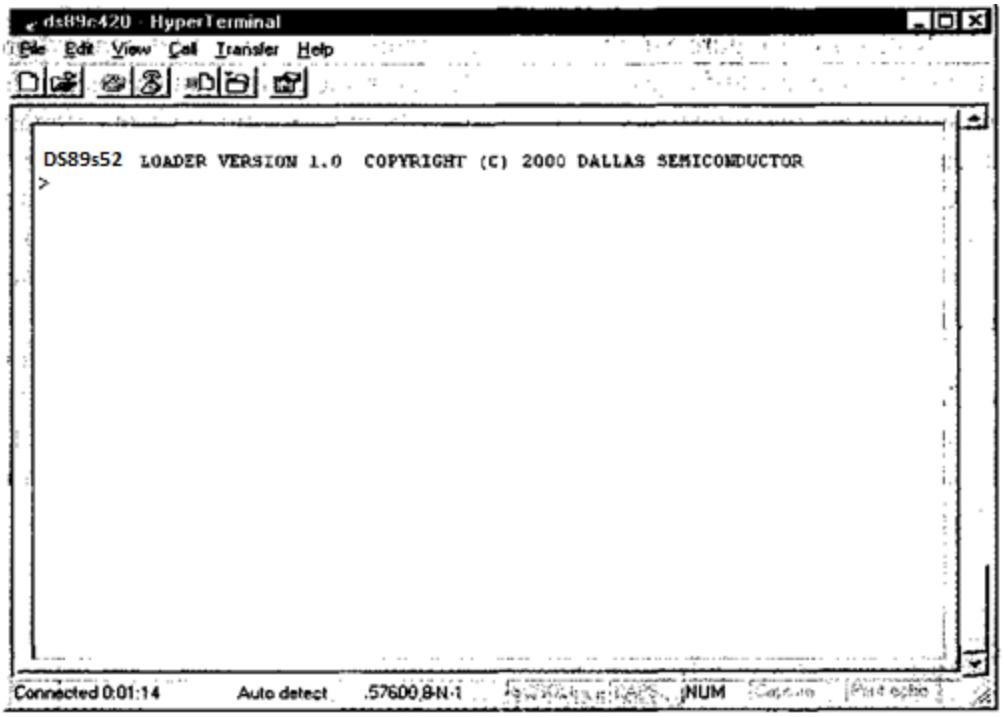


A line driver such as the **MAX232** chip is required to convert RS232 voltage levels to TTL levels, and vice versa.8051 has two pins that are used specifically for transferring and receiving data serially. These two pins are called TxD and RxD and are part of the port 3 group (P3.0 and P3.1).These pins are TTL compatible; therefore, they require a line driver to make them RS232 compatible. We need a line driver (voltage converter) to convert the R232's signals to TTL voltage levels that will be acceptable to 8051's TxD and RxD pins.



Communicating with DS89s52 Trainer(Software view):

After we build our DS89s52-based system, we can communicate with it using the HyperTerminal software. HyperTerminal comes with Microsoft Windows 98. NT. 2000 and XP.



Assuming that your serial cable has a DB-9 connector on both ends, we take the following steps to establish communication between the DS89C4xO Trainer and HyperTerminal.

1. **With** the **trainer**'s power off. We connect the COM1 port on the back of our PC to one end of the serial cable.
2. The other end of the serial cable is connected to the DB-9 connection on the DS89C4xO **Trainer** designated as SERIAL=0. After we connect our DS89C420 **Trainer** to our PC. Power up the **trainer**. Set the switch to the program position.
3. In Windows Accessories, click on HyperTerminal. (If you get a modem installation option, choose '"No".)
4. Type a name, and click OK (or HyperTerminal will not let you go on).
5. For "Connect Using" select COM1 and click OK. Choose COM2 if COM1 is used by the mouse.

6. Pick 9600 baud rate, 8-bit data, no parity bit and 1 stop bit.
7. Change the “Flow Control” to NONE or Xon Xoff and click OK. (Definitely do not choose the hardware option.)
8. Now you are in Windows HyperTerminal. and when you press the ENTER key a couple of times, the DS89C4xO will respond **with** the following message:
   DS89s52 LOADER VERSION 1.0 COPYRIGHT (C) 2000 DALLAS SEMICONDUCTOR>

Loading and running a program with the DS89s52 Trainer:

After we get the “>” from the DS89s52, we are ready to load the program into it and run. First, we have to make sure that the file we are loading is in Intel hex format. The Intel hex format is provided by our 8051 assembler/compiler.

Erase command for the DS89s52:

To reload the DS89C4xO chip with another program, we first need to erase its contents. The K (Klean) command will erase the entire contents of the flash ROM of the chip. Remember that you must use the “>K” command to erase the ROM before you can reload any program. We can verify the operation of the “>K” command by using the Dump command to display ROM contents on screen. We have to see all FFs in all the locations of ROM after applying the “>K” command.

Loading the program:

After making sure that we have the switch on the program position and you have the “>” prompt on screen, we go through the following steps to load a program:

1. At the “>” prompt, enter L (L is for Load). Example: “>L” and press Enter.
2. In HyperTerminal, click on the Transfer menu option. Click on Send Text File.
3. Select file from disk. Example: “C:test.hex”
4. Wait until the loading is complete. The appearance of the “GGGG>” prompt indicates that the loading is good and finished.
5. Now we use D to dump the contents of the flash ROM of the DS89C4xO onto the screen. Example: >D 00 4F

The dump will give us the opcodes and operands of all the instructions in our program. We can compare this information with the information provided by the list file. In the next section, we will examine the Intel hex file and compare it with the list file of the test program.

Running the program:

We change the switch to the run position, press the **reset** button on the DS89s52 system and the program will execute. Use a logic probe (or scope) to see the PO, PI, and P2 bits toggle "on" and "off continuously with some delay in between the "ON" and "OFF" states.

Test program for the DS89s52 in Assembly and C:

We are to test DS89s52 hardware connection, we can run a simple test i which all the bits of PO, PI, and P2 toggle continuously with some delay in-between the "on" and "off states. The programs for testing the trainer in bot Assembly and C are provided below. Notice that the time delay is for a DS89s52 based on the 11.0592 MHz crystal frequency. This time delay must be modified for the AT89s51/52 chips since DS89s52 uses a machine cycle of 1 clock period instead of the 12 clock periods used by the AT89Cs51/52 chip.

Programming Code:

Trainer Test Program in Assembly

```
            ORG 0H
MAIN:       MOV P0, #55H
            MOV P1, #55H
            MOV P2, #55H
            MOV R5, #250
            ACALL MSDELAY
            MOV P0, #0AAH
            MOV P1, #0AAH
            MOV P2, #0AAH
            MOV R5, #250
            ACALL MSDELAY
            SJMP MAIN
;---------- 250 MILLISECOND DELAY ---
MSDELAY:
HERE3:      MOV R4, #35
HERE2:      MOV R3, #79
HERE1:      DJNZ R3, HERE1
            DJNZ R4, HERE2
            DJNZ R5, HERE3
            RET
            END MAIN
```

Trainer Test Program in C

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
  {
    while(1)                //repeat forever
      {
        P0=0x55;            //send value to port
        P1=0x55;

         P2=0x55;
         MSDelay(250);      //call 250 ms function
         P0 = 0xAA;         //set value to port
         P1 = 0xAA;
         P2 = 0xAA;
         MSDelay(250);      //call 250 ms function
       }
  }

 void MSDelay(unsigned int itime)
   {
   unsigned int i, j;
   for(i=0;i<itime;i++)
     for(j=0;j<1275;j++);
  }
```

Explaining the HEX file:

Intel HEX file is a widely used file format designed to standardize the loading of executable machine codes into a ROM chip. Therefore, loaders that come with every ROM burner (programmer) support the Intel HEX file format.

```
LOC   OBJ        LINE
0000             1                ORG 0H
0000 758055      2     MAIN:      MOV P0, #55H
0003 759055      3                MOV P1, #55H
0006 75A055      4                MOV P2, #55H
0009 7DFA        5                MOV R5, #250
000B 111C        6                ACALL MSDELAY
000D 7580AA      7                MOV P0, #0AAH
0010 7590AA      8                MOV P1, #0AAH
0013 75A0AA      9                MOV P2, #0AAH
0016 7DFA        10               MOV R5, #250
0018 111C        11               ACALL MSDELAY
001A 80E4        12               SJMP MAIN
                 13 ;--- THE 250 MILLISECOND DELAY.
                 14    MSDELAY:
001C 7C23        15    HERE3:     MOV R4, #35
001E 7B4F        16    HERE2:     MOV R3,#79
0020 DBFE        17    HERE1:     DJNZ R3, HERE1
0022 DCFA        18               DJNZ R4, HERE2
0024 DDF6        19               DJNZ R5, HERE3
0026 22          20               RET
                 21               END
```

Figure 4: List File for Test Program (Assembly)

The hex file provides the following:

- The number of bytes of information to be loaded
- The information itself
- The starting address where the information must be placed

Program list file for test program:

The list file for the test program is given in **Figure 4**. The LOG and OBJ fields in Figure 4 must be noted. The location is the address where the opcodes (object codes) are placed. The LOG and OBJ information is used to create the hex file. Next, we will analyze the hex file belonging to the list file of Figure 4.

Analyzing Intel hex file:

Figure 5 shows the Intel hex file for the test program whose list file is given in **Figure 4**. Since the ROM burner (loader) uses the hex file to download the opcode into ROM. the hex file must provide the following: (1) the number of bytes of information to be loaded, (2) the information itself, and (3) the starting address where the information must be placed. Each line of the hex file consists of six parts. In **Figure 5**. We have separated the parts to make it easier to analyze. The following describes each part.

```
:10000000758055759055 75A0557DFA111C7580AA9F
:100010007590AA75A0AA7DFA111C80E47C237B4F01
:07002000DBFEDCFADDF62235
:00000001FF
```

Seperating the fields we get the following

```
:CC   AAAA TT   DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD   SS
:10   0000 00   758055759055 75A0557DFA111C7580AA   9F
:10   0010 00   7590AA75A0AA7DFA111C80E47C237B4F   01
:07   0020 00   DBFEDCFADDF622                      35
:00   0000 01   FF
```

Figure5: Intel Hex File Test Program as Provided by the Assembler

1. ":" Each line starts with a colon.
2. CC, the count byte. This tells the loader how many bytes are in the line. CC can range from 00 to 16 (10 in hex).
3. AAAA is for the address. This is a 16-bit address. The loader places the first byte of data into this memory address.
4. TT is for type. This field is either 00 or 01. If it is 00, it means that there are more lines to come after this line. If it is 01, it means that this is the last line and the loading should stop after this line.
5. DD D is the real information (data or code). There is a maximum of 16 bytes in this part. The loader places this information into successive memory locations of ROM.
6. SS is a single byte. This last byte is the checksum byte of everything in that line. The checksum byte is used for error checking. Notice that the checksum byte at the end of each line represents everything in that line and not just the data portion.

Now we compare the data portion of Intel Hex file in figure: 5 with the information under the OBJ field of the list file in figure: 4. It is notable that they are identical, as they should be. The extra information is added by the Intel Hex file formatter. We can run the C language version of the test program and verify its operation. Our C compiler will provide us both the list file and the Intel Hex file if we want to explore the Intel Hex file concept.

Some Troubleshooting Tips:

Running the test program on our DS89s52-based trainer (or 8051 system) should toggle all the I/O bits with some delay. If our system does not work, we need follow these steps to find the problem.

1. With the power off, we have checked our connection for all pins, especially V_{cc} and GND.
2. Check RST (pin #9) using an oscilloscope. When the system is powered up, pin *#9* is low. Upon pressing the momentary switch it goes high. We have to make sure the momentary switch is connected properly.
3. Observe the XTAL2 pin on the oscilloscope while the power is on. We should see a crude square wave. This indicates that the crystal oscillator is good.
4. If all the above steps pass inspection, we check the contents of the on-chip ROM starting at memory location 0000. It must be the same as the opcodes provided by the list file of **Figure 4**. Our assembler produces the list file, which lists the opcodes and operands on the left side of the assembly instructions. This must match exactly the contents of our on-chip ROM if the proper steps were taken in burning and loading the program into the on-chip ROM.

Conclusions:

We have successfully run the program based on 8051 microcontroller to check the functionality of the DS89s52 trainer board. Though we faced some problems regarding the programming could we finally overcome this with the help of some software like Visual Basic , Winasm, TOP_{2007} etc. We have checked the all possible ways of interfacing the microcontroller with PC through the DB9 connector. Though there might be some errors while running the Trainer board due to some defects on the hardware but this can be easily solved. Therefore, We hope that our work will definitely be regarded as an achievement in the field of Microcontroller based circuit designing.

References:

- ✓ *The 8051 Microcontroller and Embedded Systems Using Assembly and C (2nd Edition)* -**Muhammad Ali Mazidi, Rolin D. McKinlay**.
- ✓ *Atmel CISC Microcontroller Trainer/Writer kit* - **Golam Mostafa**
- ✓ *ATmega/32 RISC Microcontroller Interfacing and System Design*- **Golam Mostafa**
- ✓ *http://what-when-how.com/8051-microcontroller/basics-of-serial-communication*
- ✓ *http://what-when-how.com/8051-microcontroller/design-and-test-of-ds89s52xo-trainer*
- ✓ *Programming and Customizing The AVR Microcontroller*-**Dhananjay V. Gadre**
- ✓ *8051 Microcontrollers*- David **Calcutt, Fred Cowan, Hassan Parchizadeh**
- ✓ *Programming and Customizing the 8051 Microcontroller*- **Myke Predko**
- ✓ **ATMEL AT89s52 Data Sheet** (*SFR map and RESET value*)