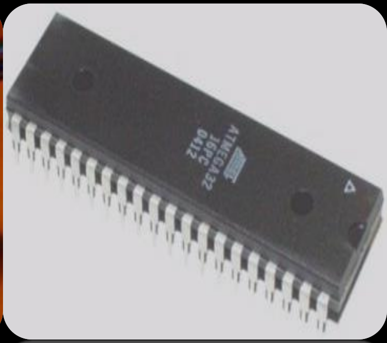
Microcontroller-based Stepper motor Control System with Computer-based User Interface(UI) and Internet-Control Accessibility For Remote Control





Submitted by :

Raziur Rahman

Rashed Hasan Bipu

Saugato Rahman Dhruba

Abul Hasan Fahad

Monjurul Feroz Meem

This presentation is divided into 5 parts

1. Main Idea

2. Introduction to components

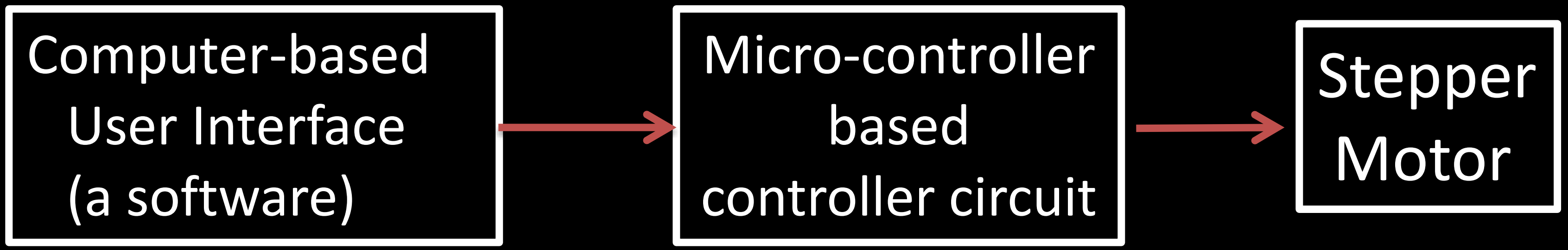
3. Operation

4. Cost Analysis

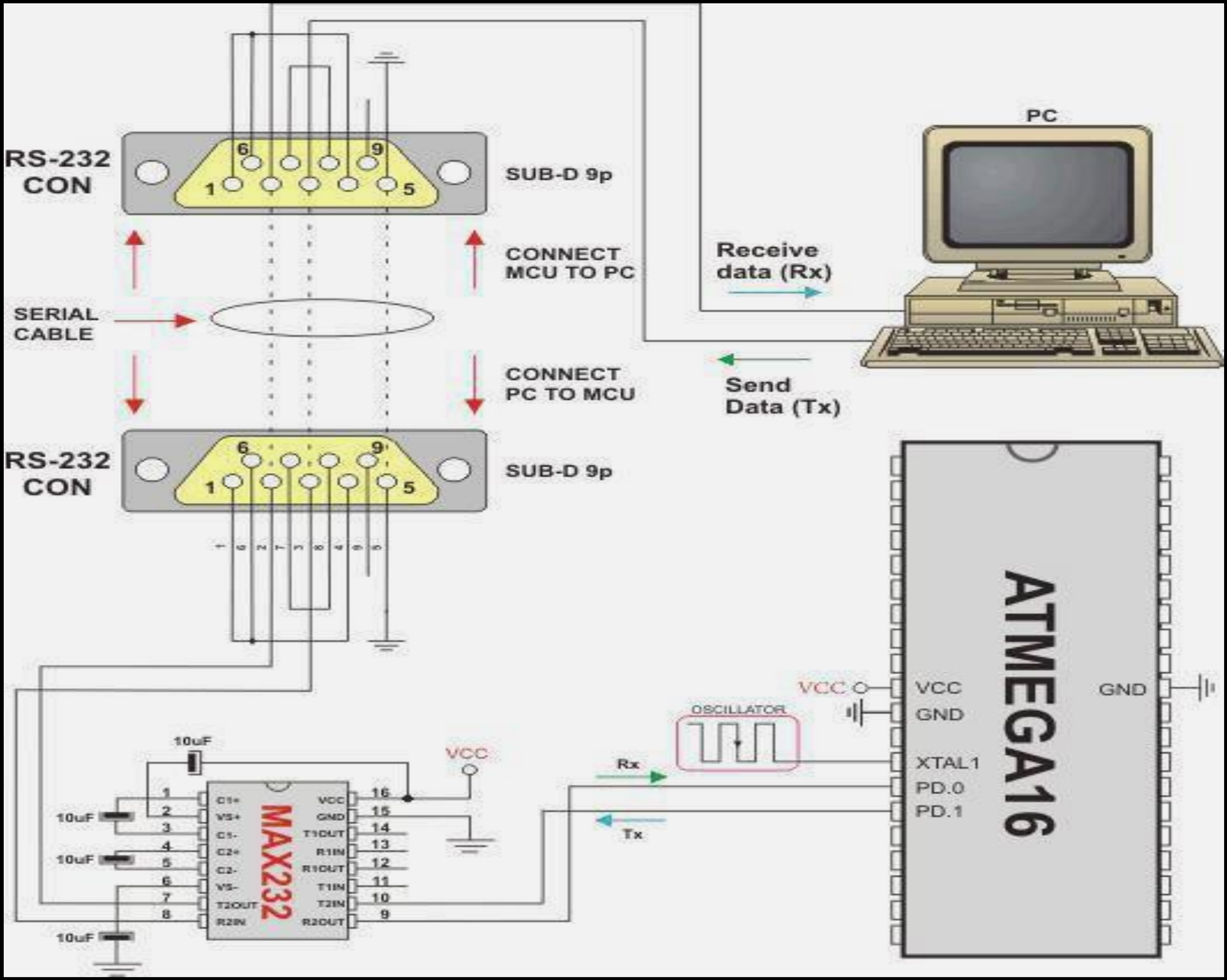
5. Achievements

Main Idea

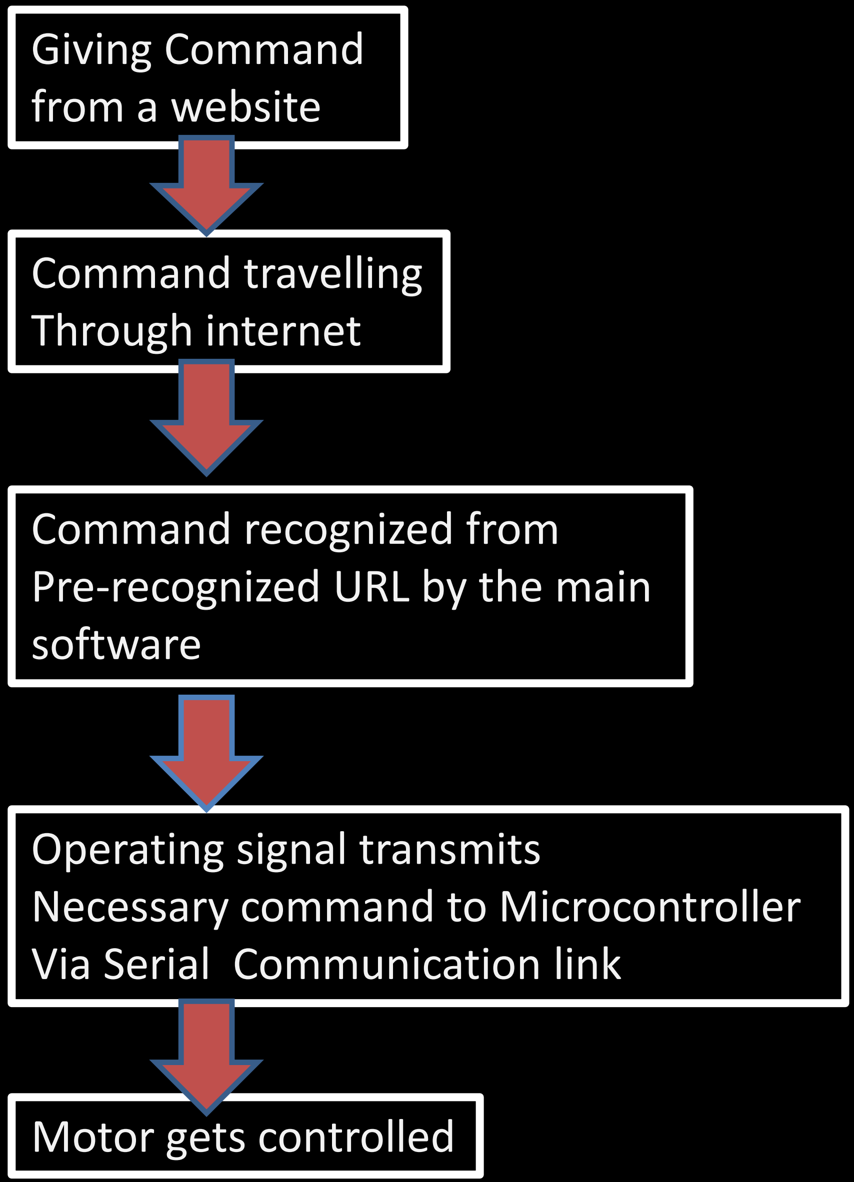
The total operation is described in the following block diagram



Graphical Representation of Block Diagram



Flow chart of Remote Motor Control Via. Internet



How to operate using Internet

Go to Mytextfile.com

Signing in with a Google account

Give Command from Mytextfile.com (Computer or mobile)

Command recognized by the Central computer Software

Command goes to microcontroller circuit via Serial communication Port

(USB-to-Serial Conversion and RS-232)

Motor control accomplished

What is actually needed?

1. Stepper Motor
2. Micro-controller
3. Computer-based User Interface
4. Internet Connectivity (Only for Remote control)
5. USB-to-Serial Converter
6. Computer or mobile phone for Remote control
7. Five volt supply

Introduction of Different Components

Stepper Motor

A **stepper motor** (or **step motor**) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application.

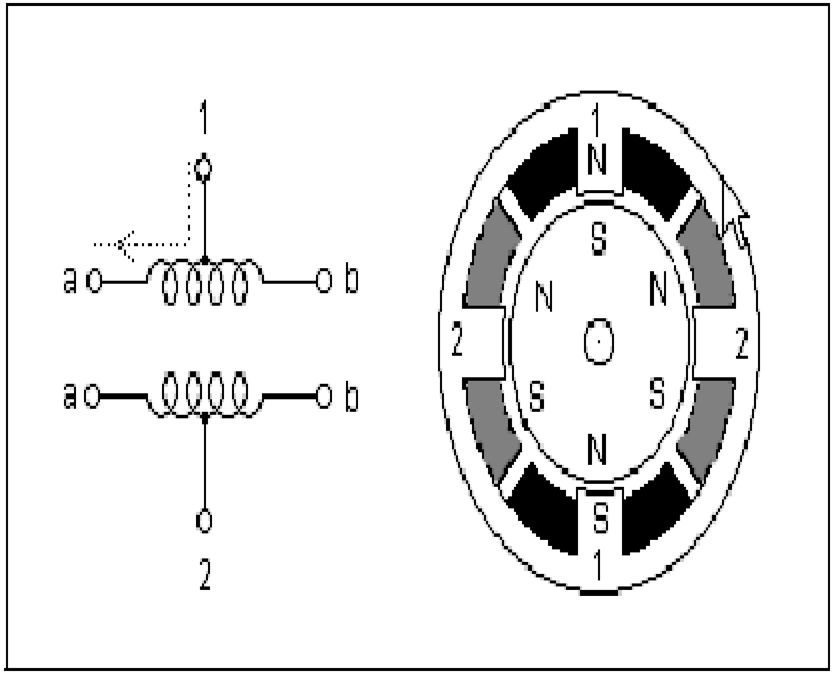
There are four main types of stepper motors:[

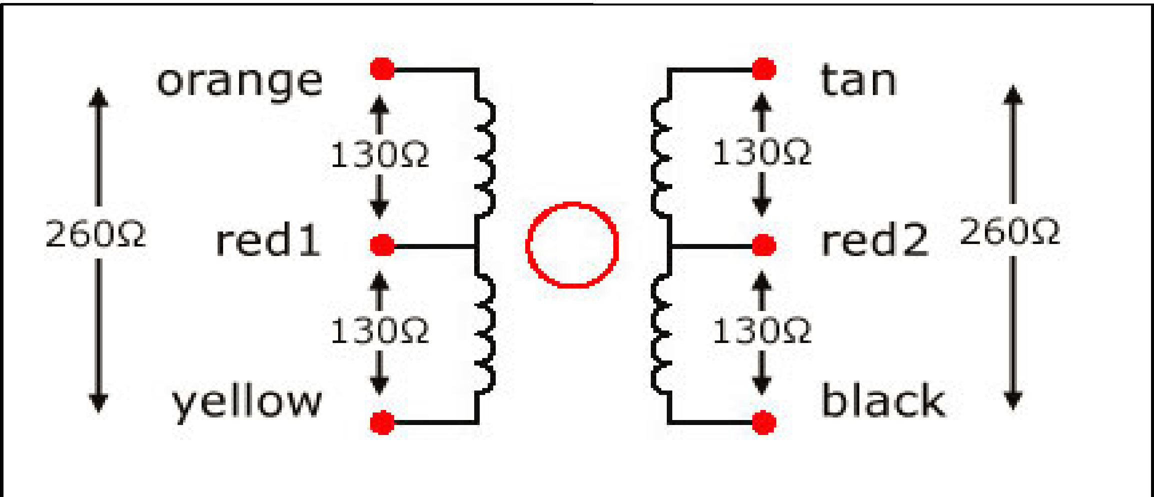
1. Permanent magnet stepper (can be subdivided into 'tin-can' and 'hybrid', tin-can being a cheaper product, and hybrid with higher quality bearings, smaller step angle, higher power density)
2. Hybrid synchronous stepper
3. Variable reluctance stepper
4. Lavet type stepping motor

Unipolar Motors

Unipolar stepping motors are composed of two windings, each with a center tap. The center taps are either brought outside the motor as two separate wires (as shown in Figure 2) or connected to each other internally and brought outside the motor as one wire. As a result, unipolar motors have 5 or 6 wires. Regardless of the number of wires, unipolar motors are driven in the same way. The center tap wire(s) is tied to a power supply and the ends of the coils are alternately grounded.

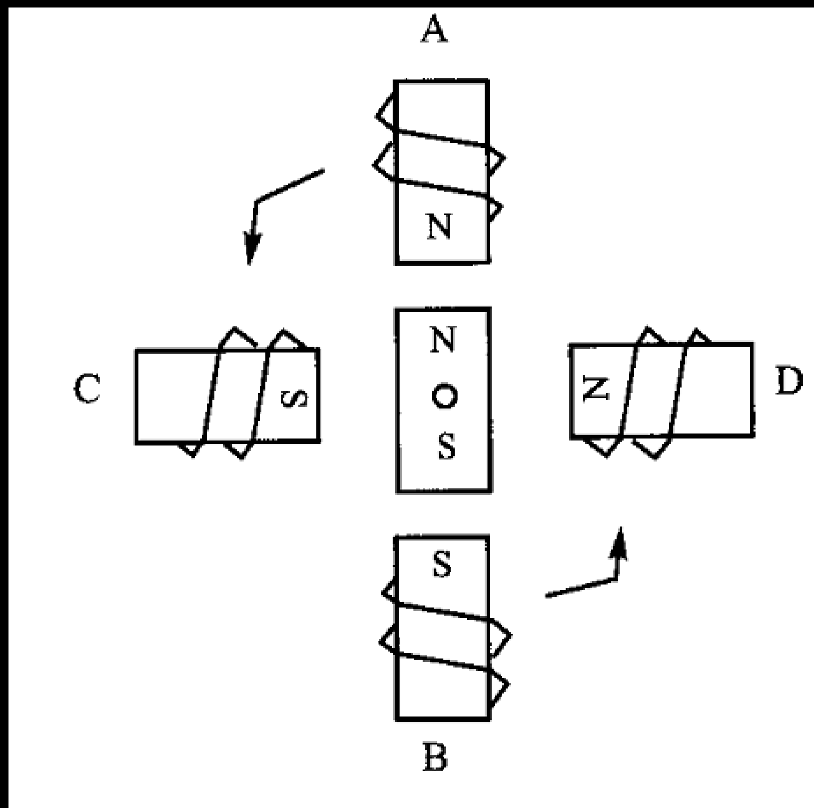
FIGURE 2: UNIPOLAR STEPPER MOTOR





Normal Four-Step Sequence

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Counter-clockwise
	1	1	0	0	1	
	2	1	1	0	0	
	3	0	1	1	0	
	4	0	0	1	1	

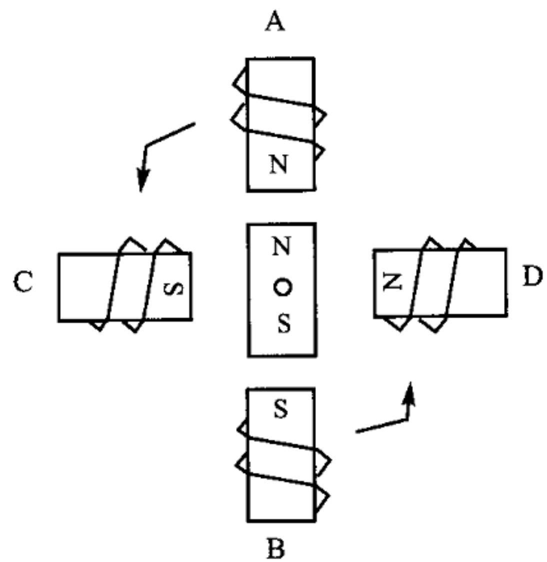


Half-Step 8-Step Sequence

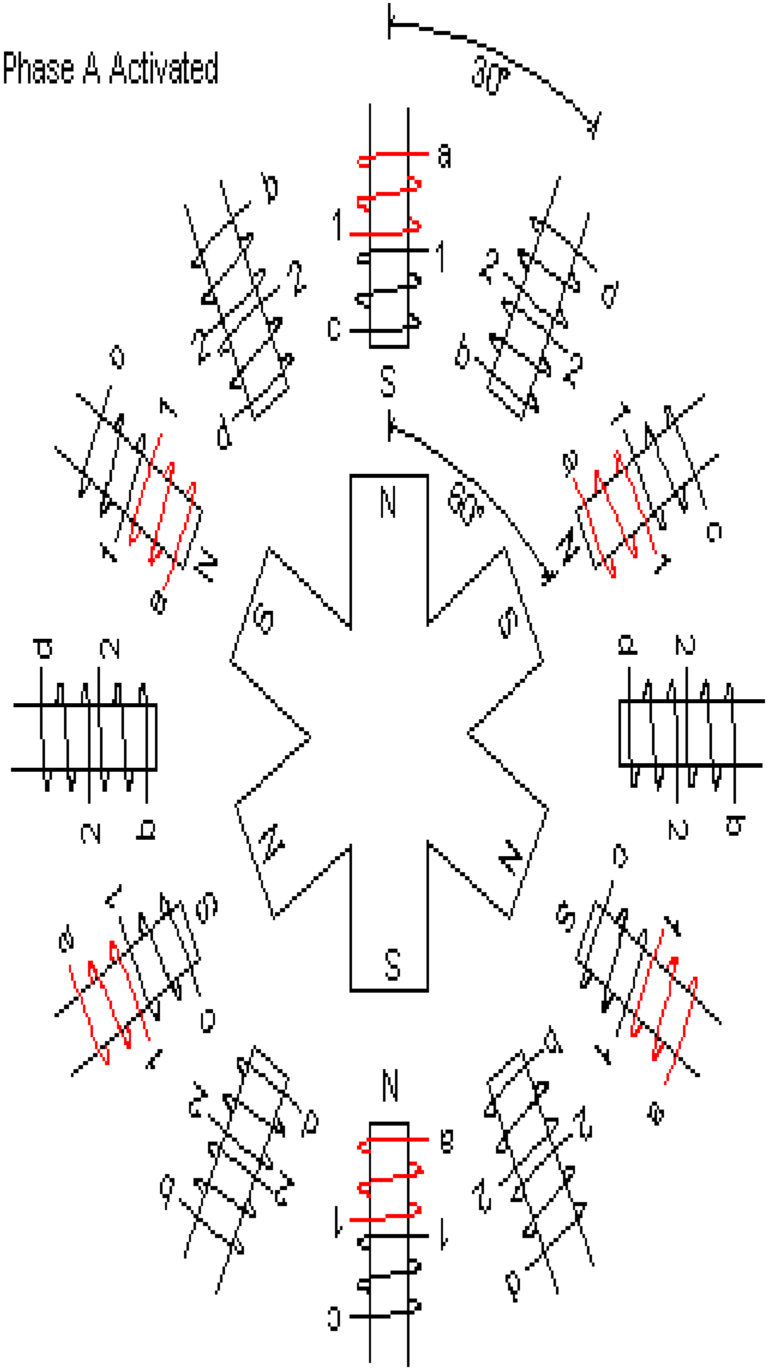
Clockwise

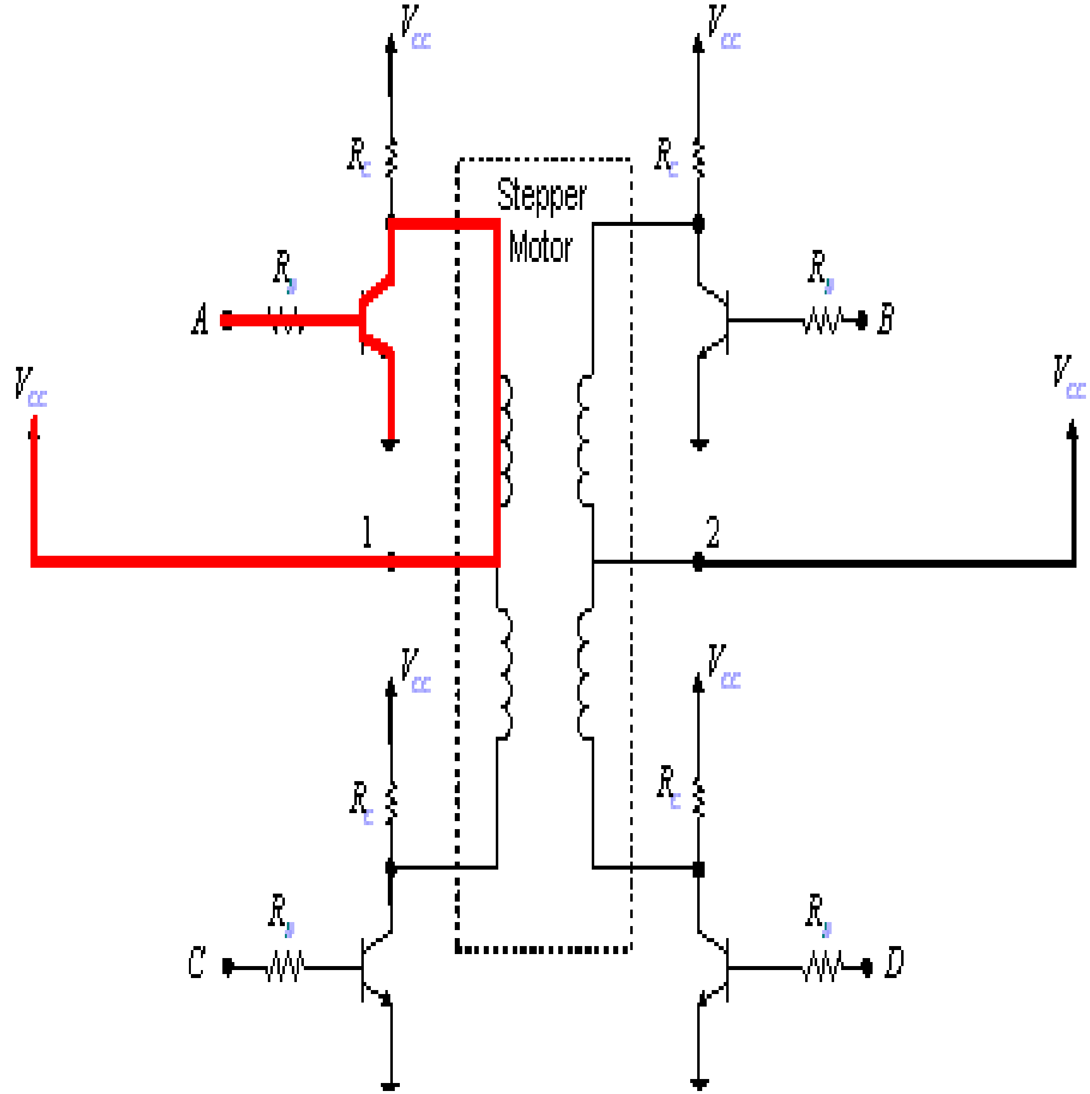
Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

Counter-clockwise



Phase A Activated

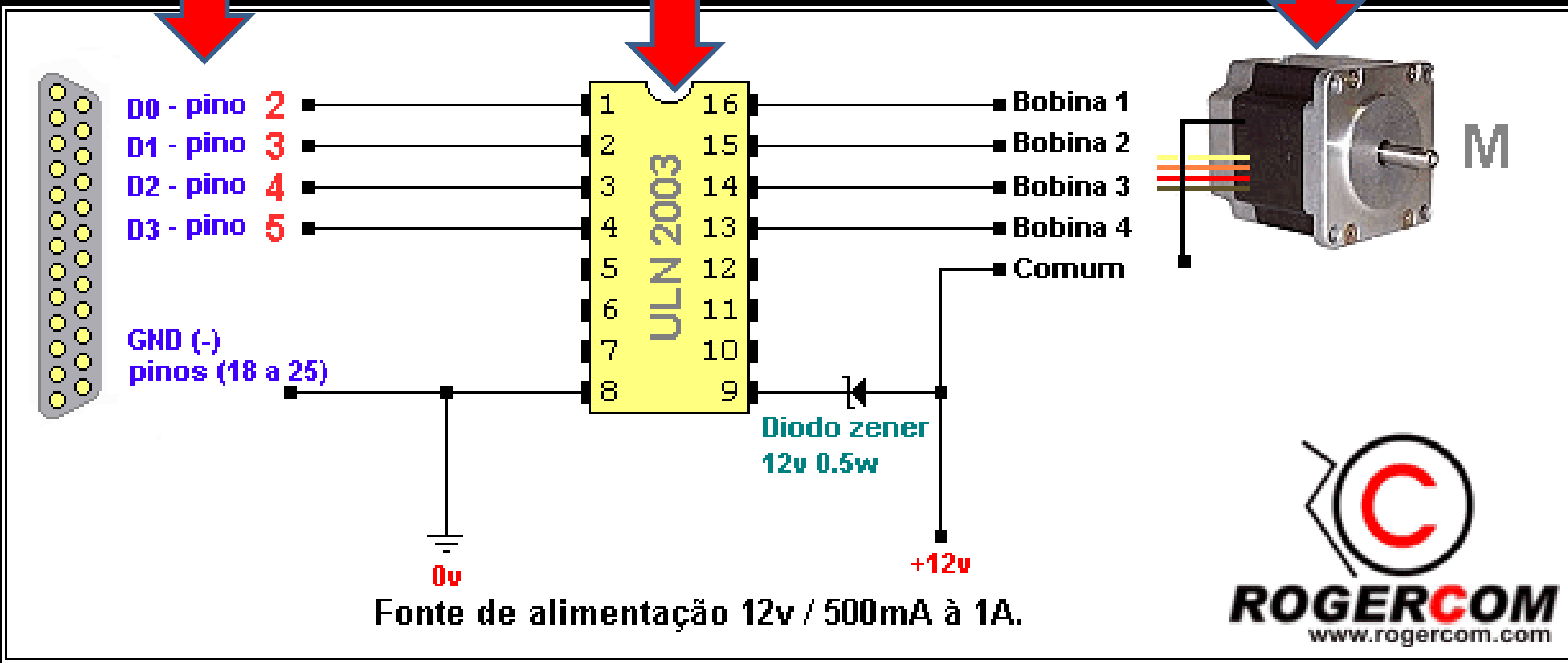




Pins from Microcontroller

This IC is here to provide extra current necessary to drive the motor. It supplies current necessary to excite the motor windings.

Motor, requiring high current

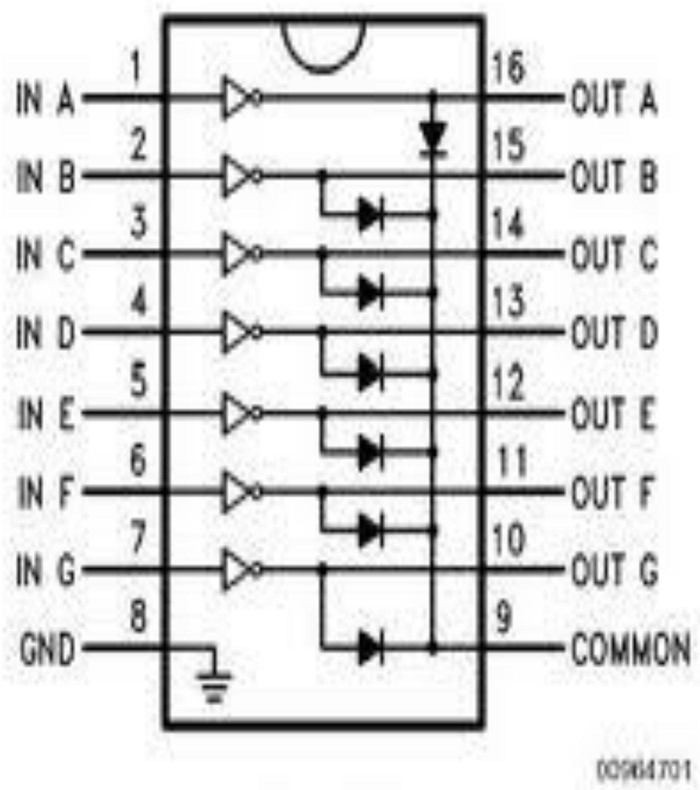


DESCRIPTION

The ULN2003 is a monolithic high voltage and high current Darlington transistor arrays. It consists of seven NPN darlington pairs that features high-voltage outputs with common-cathode clamp diode for switching inductive loads. The collector-current rating of a single darlington pair is 500mA. The darlington pairs may be parrleled for higher current capability. Applications include relay drivers,hammer drivers, lampdrivers,display drivers(LED gas discharge),line drivers, and logic buffers.

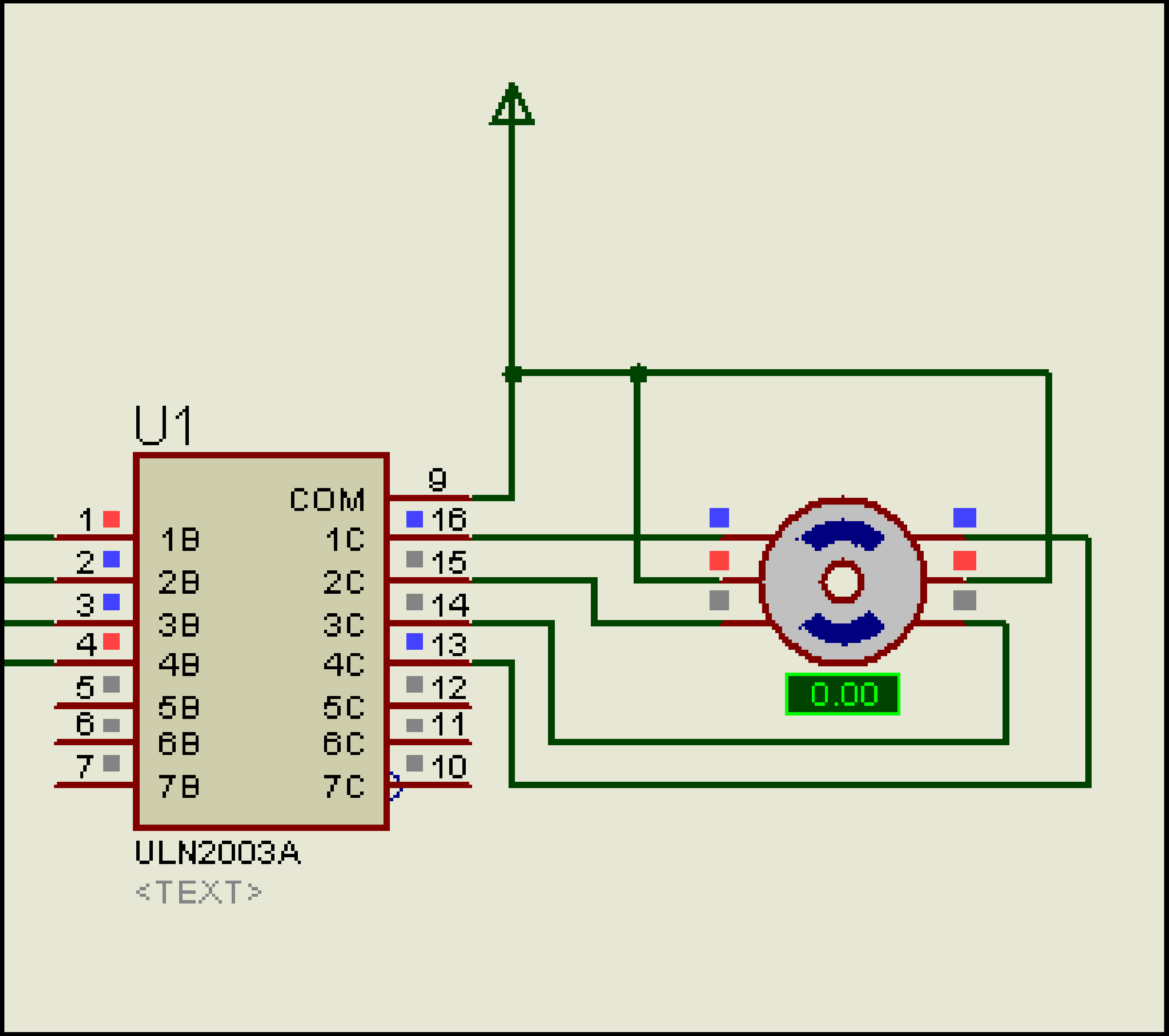
The ULN2003 has a 2.7kΩ series base resistor for each darlington pair for operation directly with TTL or 5V CMOS devices.

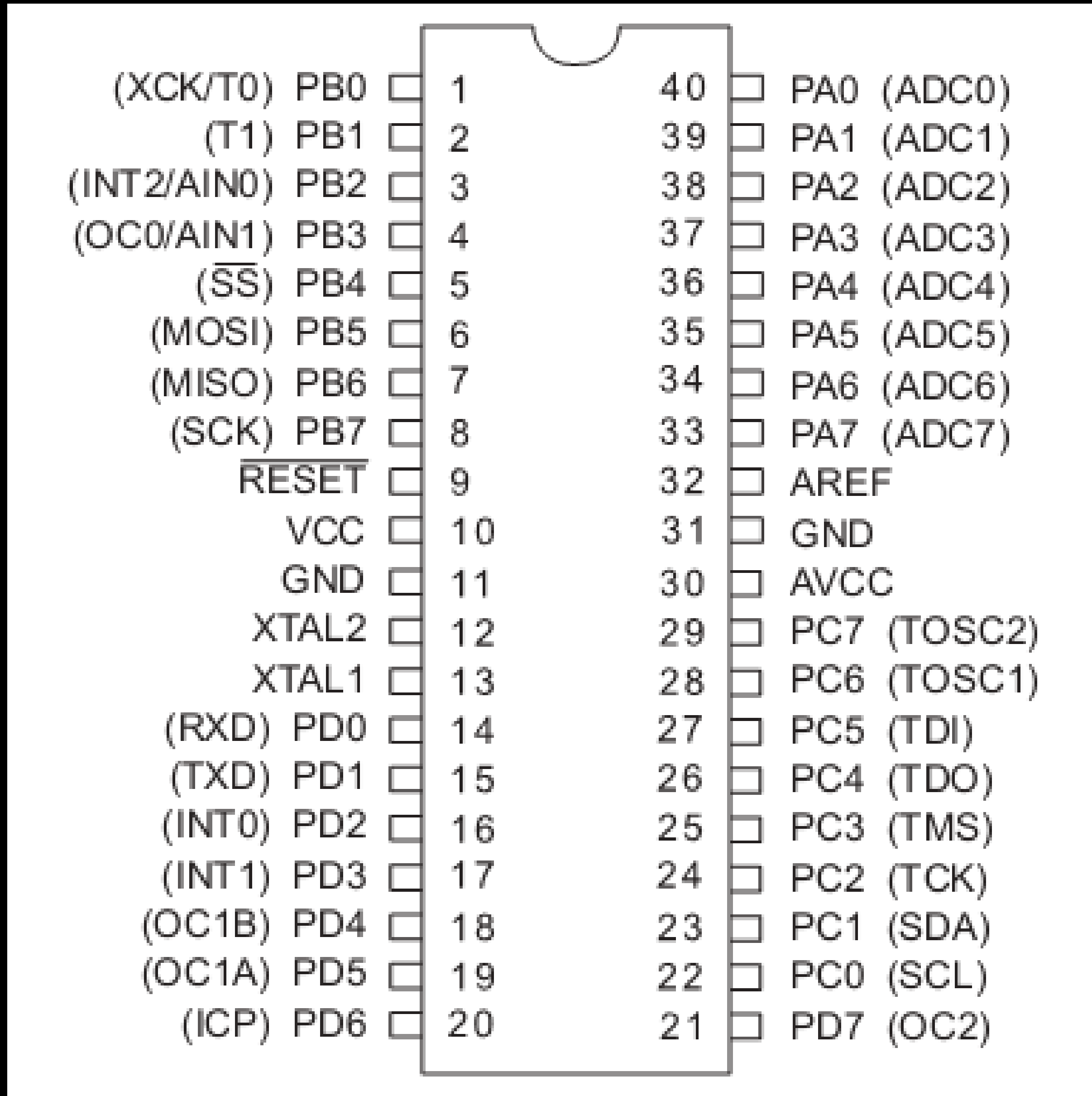
16-Lead DIP



Top View

Animation of Interfacing ULN-2003 and Stepper Motor (unipolar)




(XCK/T0) PB0
(T1) PB1
(INT2/AIN0) PB2
(OC0/AIN1) PB3
($\overline{SS}$) PB4
(MOSI) PB5
(MISO) PB6
(SCK) PB7
$\overline{RESET}$
VCC
GND
XTAL2
XTAL1
(RXD) PD0
(TXD) PD1
(INT0) PD2
(INT1) PD3
(OC1B) PD4
(OC1A) PD5
(ICP) PD6
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
PA0 (ADC0)
PA1 (ADC1)
PA2 (ADC2)
PA3 (ADC3)
PA4 (ADC4)
PA5 (ADC5)
PA6 (ADC6)
PA7 (ADC7)
AREF
GND
AVCC
PC7 (TOSC2)
PC6 (TOSC1)
PC5 (TDI)
PC4 (TDO)
PC3 (TMS)
PC2 (TCK)
PC1 (SDA)
PC0 (SCL)
PD7 (OC2)

Features

- High-performance, Low-power AVR 8-bit Microcontroller

- Advanced RISC Architecture

– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16 MHz
– On-chip 2-cycle Multiplier

- Nonvolatile Program and Data Memories

– 32K Bytes of In-System Self-Programmable Flash

– Optional Boot Code Section with Independent Lock Bits, In-System Programming by On-chip Boot Program , True Read-While-Write Operation
– 1024 Bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
– 2K Byte Internal SRAM
– Programming Lock for Software Security

Peripheral Features of ATmega32

– Two 8-bit Timer/Counters with Separate Pre-scalers and Compare Modes
– One 16-bit Timer/Counter with Separate Pre-scaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC
– Byte-oriented Two-wire Serial Interface
– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator

Other Features of ATmega32

- I/O and Packages
– 32 Programmable I/O Lines
– 40-pin PDIP, 44-lead TQFP, and 44-pad MLF

- Operating Voltages
– 2.7 - 5.5V for ATmega32L
– 4.5 - 5.5V for ATmega32

- Speed Grades
– 0 - 8 MHz for ATmega32L
– 0 - 16 MHz for ATmega32
- Power Consumption at 1 MHz, 3V, 25°C for ATmega32L
– Active: 1.1 mA
– Idle Mode: 0.35 mA
– Power-down Mode: < 1 µA

Ports in Atmega32

- Each port pin in AVR ATmega32 consists of three register bits: DDxn, PORTxn, and PINxn.

- The DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

- The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is config-ured as an input pin.

- If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

- If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an out-put pin, the port pin is driven low (zero).

PORT PIN CONFIGURATIONS

Table 20. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Serial Vs. Parallel Communication

- Although a serial link may seem inferior to a parallel one, since it can transmit less data per clock cycle, it is often the case that serial links can be clocked considerably faster than parallel links in order to achieve a higher data rate.
- In many cases, serial is a better option because it is cheaper to implement. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins and are therefore less expensive.

USB-to-Serial Conversion

What is RS-232

In telecommunications, **RS-232** is the traditional name for a series of standards for serial binary single-ended data and control signals connecting between a *DTE* (Data Terminal Equipment) and a *DCE* (Data Circuit-terminating Equipment). It is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors. The current version of the standard is *TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, issued in 1997.

What is USB

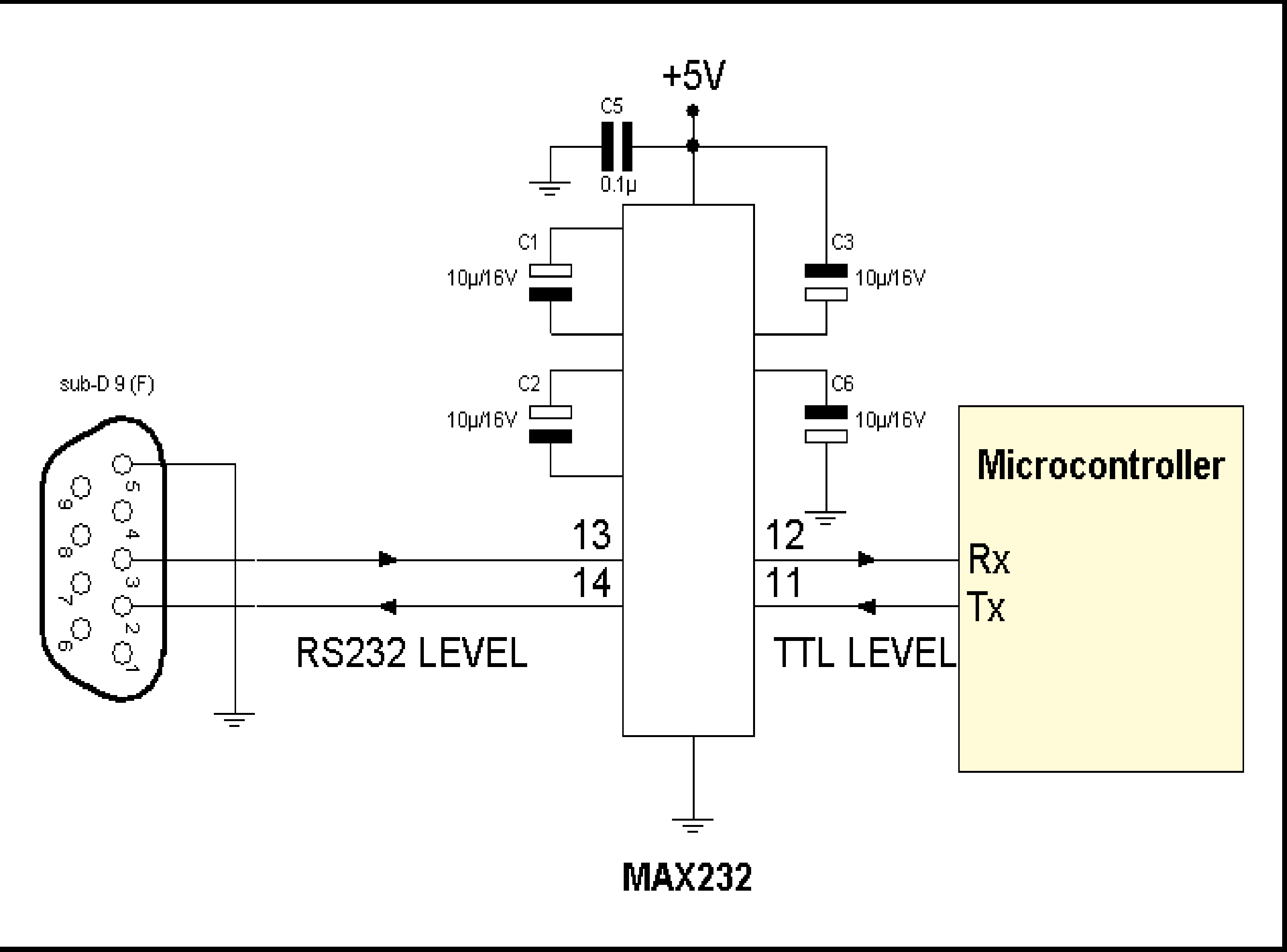
Universal Serial Bus (**USB**) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication and power supply between computers and electronic devices. It was invented by Ajay Bhatt.
USB was designed to standardize the connection of computer peripherals, such as keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smartphones, PDAs and video game consoles. USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

MAX232

Because the RS232 is not compatible with today's microprocessors and microcontrollers, we need a line driver (voltage converter) to convert the RS232's signals to TTL voltage levels that will be acceptable to the AVR's TX and RX pins. One example of such a converter is MAX232 from Maxim Corp. (www.maxim-ic.com). The MAX232 converts from RS232 voltage levels to TTL voltage levels, and vice versa. One advantage of the MAX232 chip is that it uses a +5 V power source, which is the same as the source voltage for the AVR. In other words, with a single +5 V power supply we can power both the AVR and MAX232, with no need for the dual power supplies that are common in many older systems.

RS232 Line Type & Logic Level	RS232 Voltage	TTL Voltage to/from MAX232
Data Transmission (Rx/Tx) Logic 0	+3 V to +15 V	0 V
Data Transmission (Rx/Tx) Logic 1	-3 V to -15 V	5 V
Control Signals (RTS/CTS/DTR/DSR) Logic 0	-3 V to -15 V	5 V
Control Signals (RTS/CTS/DTR/DSR) Logic 1	+3 V to +15 V	0 V

Interfacing Serial Port with Microcontroller



AVR SERIAL PORT PROGRAMMING

In the AVR microcontroller five registers are associated with the USART

- UDR (USART Data Register)
- UCSRA
- UCSRB
- UCSRC (USART Control Status Register)
- UBRR

PC Baud Rates in HyperTerminal
1,200
2,400
4,800
9,600
19,200
38,400
57,600
115,200

RX and TX pins in the ATmega32

The ATmega32 has two pins that are used specifically for transferring and receiving data serially. These two pins are called TX and RX and are part of the Port D group (PD0 and PD1) of the 40-pin package. Pin 15 of the ATmega32 is assigned to TX and pin 14 is designated as RX. These pins are TTL compatible; therefore, they require a line driver to make them RS232 compatible. One such line driver is the MAX232 chip. This is discussed next.

Desired Baud Rate = Fosc/ (16(X + 1))

Table 11-4: UBRR Values for Various Baud Rates (Fosc = 8 MHz, U2X = 0)

Baud Rate	UBRR (Decimal Value)	UBRR (Hex Value)
38400	12	C
19200	25	19
9600	51	33
4800	103	67
2400	207	CF
1200	415	19F

Note: For Fosc = 8 MHz we have UBRR = (500000/BaudRate) – 1

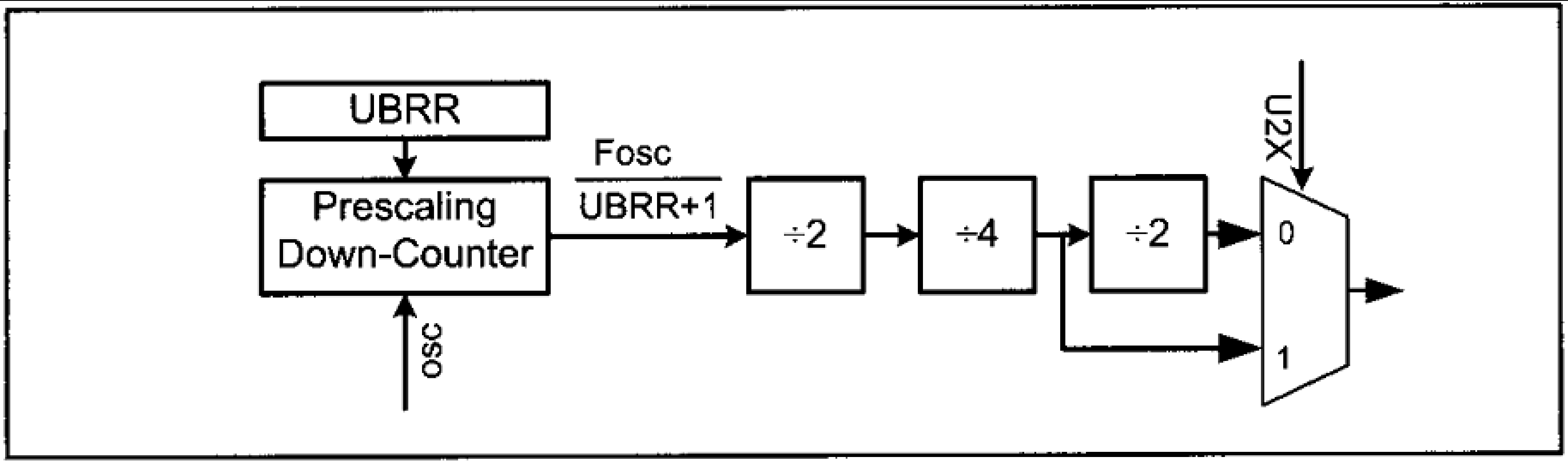


Figure 11-9. Baud Rate Generation Block Diagram

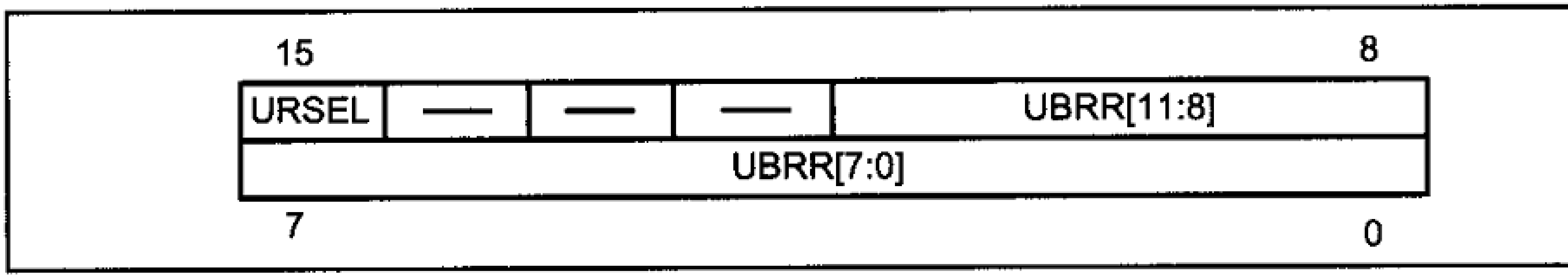


Figure 11-10. UBRR Register

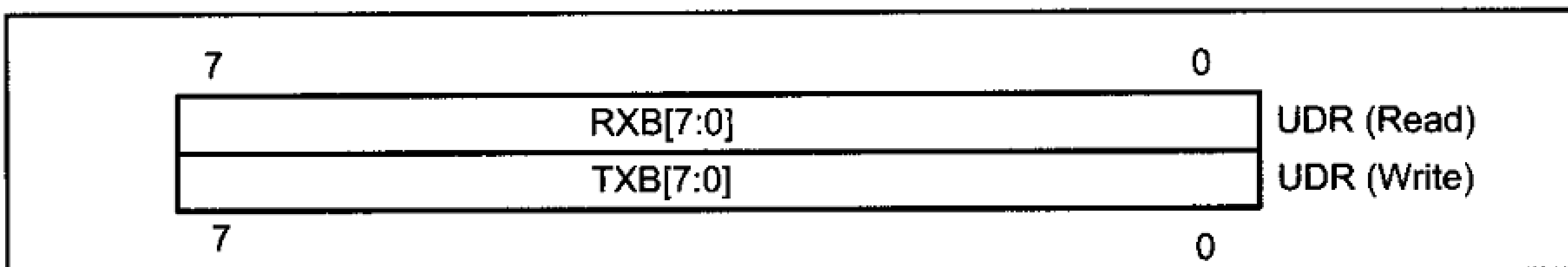
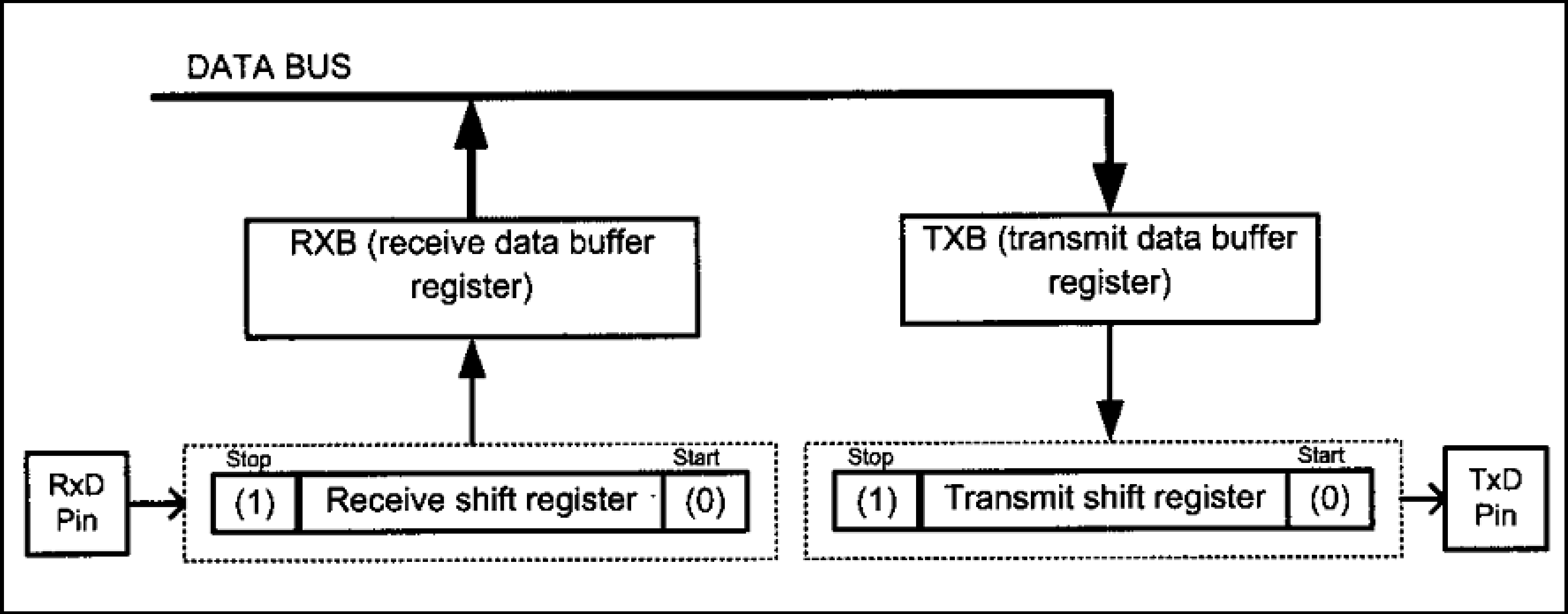


Figure 11-11. UDR Register

Simplified USART Transmit Block Diagram



UCSR registers and USART configurations in the AVR

RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

RXC (Bit 7): USART Receive Complete
This flag bit is set when there are new data in the receive buffer that are not read yet. It is cleared when the receive buffer is empty. It also can be used to generate a receive complete interrupt.

TXC (Bit 6): USART Transmit Complete
This flag bit is set when the entire frame in the transmit shift register has been transmitted and there are no new data available in the transmit data buffer register (TXB). It can be cleared by writing a one to its bit location. Also it is automatically cleared when a transmit complete interrupt is executed. It can be used to generate a transmit complete interrupt.

UDRE (Bit 5): USART Data Register Empty
This flag is set when the transmit data buffer is empty and it is ready to receive new data. If this bit is cleared you should not write to UDR because it overrides your last data. The UDRE flag can generate a data register empty interrupt.

FE (Bit 4): Frame Error
This bit is set if a frame error has occurred in receiving the next character in the receive buffer. A frame error is detected when the first stop bit of the next character in the receive buffer is zero.

DOR (Bit 3): Data OverRun
This bit is set if a data overrun is detected. A data overrun occurs when the receive data buffer and receive shift register are full, and a new start bit is detected.

UCSR registers and USART configurations in the AVR

DOR (Bit 3): Data OverRun

This bit is set if a data overrun is detected. A data overrun occurs when the receive data buffer and receive shift register are full, and a new start bit is detected.

PE (Bit 2): Parity Error

This bit is set if parity checking was enabled (UPM1 = 1) and the next character in the receive buffer had a parity error when received.

U2X (Bit 1): Double the USART Transmission Speed

Setting this bit will double the transfer rate for asynchronous communication.

MPCM (Bit 0): Multi-processor Communication Mode

This bit enables the multi-processor communication mode. The MPCM feature is not discussed in this book.

Notice that FE, DOR, and PE are valid until the receive buffer (UDR) is read. Always set these bits to zero when writing to UCSRA.

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

RXCIE (Bit 7): Receive Complete Interrupt Enable
To enable the interrupt on the RXC flag in UCSRA you should set this bit to one.

TXCIE (Bit 6): Transmit Complete Interrupt Enable
To enable the interrupt on the TXC flag in UCSRA you should set this bit to one.

UDRIE (Bit 5): USART Data Register Empty Interrupt Enable
To enable the interrupt on the UDRE flag in UCSRA you should set this bit to one.

RXEN (Bit 4): Receive Enable
To enable the USART receiver you should set this bit to one.

TXEN (Bit 3): Transmit Enable
To enable the USART transmitter you should set this bit to one.

UCSZ2 (Bit 2): Character Size
This bit combined with the UCSZ1:0 bits in UCSRC sets the number of data bits (character size) in a frame.

RXB8 (Bit 1): Receive data bit 8
This is the ninth data bit of the received character when using serial frames with nine data bits. This bit is not used in this book.

TXB8 (Bit 0): Transmit data bit 8
This is the ninth data bit of the transmitted character when using serial frames with nine data bits. This bit is not used in this book.

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

URSEL (Bit 7): Register Select
This bit selects to access either the UCSRC or the UBRRH register and will be discussed more in this section.

UMSEL (Bit 6): USART Mode Select
This bit selects to operate in either the asynchronous or synchronous mode of operation.

0 = Asynchronous operation
1 = Synchronous operation

UPM1:0 (Bit 5:4): Parity Mode
These bits disable or enable and set the type of parity generation and check.

00 = Disabled
01 = Reserved
10 = Even Parity
11 = Odd Parity

USBS (Bit 3): Stop Bit Select
This bit selects the number of stop bits to be transmitted.

0 = 1 bit
1 = 2 bits

UCSZ1:0 (Bit 2:1): Character Size
These bits combined with the UCSZ2 bit in UCSRB set the character size in a frame and will be discussed more in this section.

UCPOL (Bit 2): Clock Polarity

To set the number of data bits (character size) in a frame you must set the values of the UCSZ1 and USCZ0 bits in the UCSRB and UCSZ2 bits in UCSRC. Table 11-5 shows the values of UCSZ2, UCSZ1, and UCSZ0 for different character sizes. In this book we use the 8-bit character size because it is the most common in x86 serial communications. If you want to use 9-bit data, you have to use the RXB8 and TXB8 bits in UCSRB as the 9th bit of UDR (USART Data

Table 11-5: Values of UCSZ2:0 for Different Character Sizes

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	1	1	9

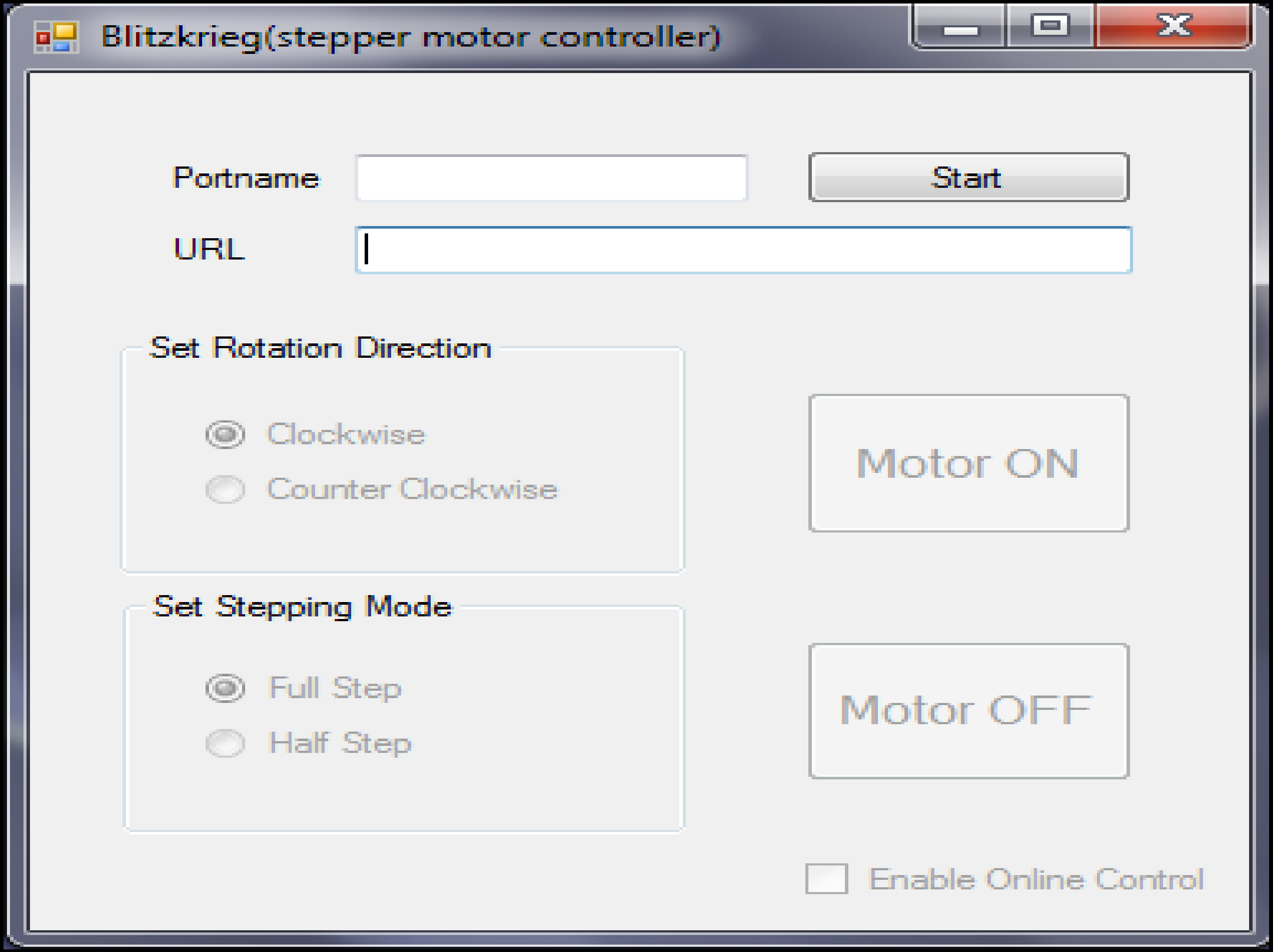
Note: Other values are reserved. Also notice that UCSZ0 and UCSZ1 belong to UCSRC and UCSZ2 belongs to UCSRB

Programming the AVR to transfer data serially

In programming the AVR to transfer character bytes serially, the following steps must be taken:

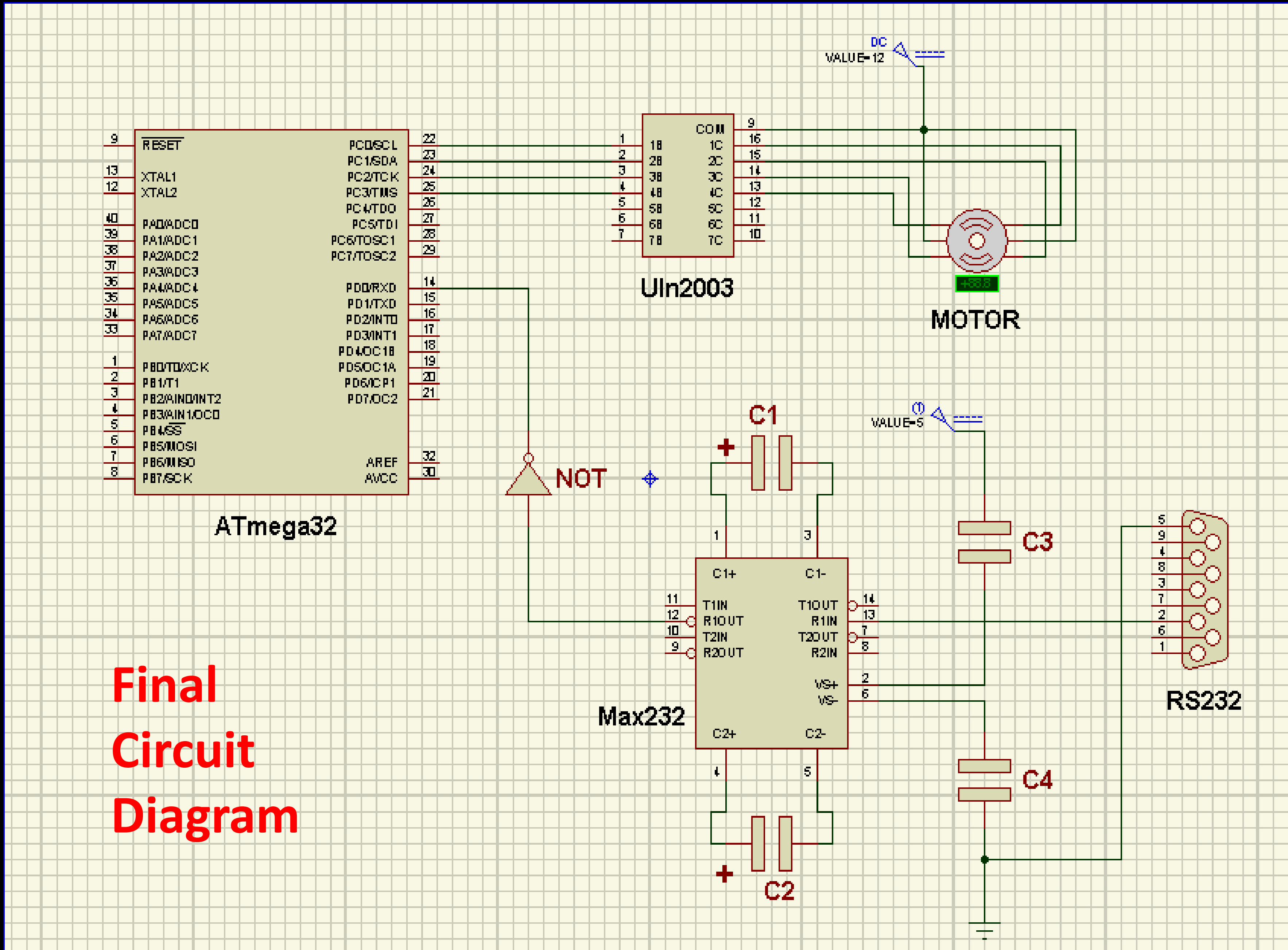
1. The UCSRB register is loaded with the value 08H, enabling the USART transmitter. The transmitter will override normal port operation for the TxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity, and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.
4. The character byte to be transmitted serially is written into the UDR register.
5. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for the next byte.
6. To transmit the next character, go to Step 4.

USER INTERFACE DEVELOPED BY C# Programming Language



Features of USER INTERFACE

Final Circuit Diagram



Operation: Step-by-Step

Step 0:

Install the *Prolific PL-2303 vista driver*. Connect the cable to the USB port when asked. (This step is needed to be done only when the device is connected to the PC *for the very first time*.)

Step 1:

Reconnect the cable. Go to ‘Device Manager’ and take note of the newly added port’s name under ‘Ports’. It should be ‘COMX’ (X may be any number like 1, 2, 3 etc) and can be found inside the Parenthesis.

Step 2:

Power the device via a 2 pin plug.

Operation: Step-by-Step

Step 3:

Open the program ‘*Blitzkrieg (stepper motor controller)*’. Type the port’s name you have noted in step 2 in the <u>‘Portname’</u> textbox. (The name MUST be written in capital letter.)

Step 4:

Click “Start”

Step 5:

Click the exit cross to exit the program.

Extra Feature: Control via Internet

For this you need a stable internet connection to the PC which will be used to operate the device. Follow the previously stated *steps 1, 2 and 3*.

Step 4:

Open the internet browser of the PC/ Cell phone/ Tablet PC by which you want to control the stepper motor.

Go to www.mytextfile.com

Step 5:

Sign in with your Gmail/ Google account.

Step 6:

Click ‘Settings’. Tick the checkbox beside ‘Publish text file to secret private URL:’. Click ‘Save Settings’.

Extra Feature: Control via Internet

Step 7:

Again go to Settings and you will find *a URL* beside 'Publish text file to secret private URL:'. Copy the URL to the 'URL' textbox of the *'Blitzkrieg (stepper motor controller)'* program opened previously in the PC connected with our device.

Step 8:

Click Start and then Tick the checkbox beside 'Enable Online Control'. Now we can start controlling the stepper motor via internet.

Extra Feature: Control via Internet

Step 9:

On the www.mytextfile.com site you will find a text editor window. Here if you write a key-letter and then click ‘Save’ the motor will act accordingly. The key-letters are-

‘A’ for clockwise full step rotation
‘B’ for clockwise half step rotation
‘C’ for anti-clockwise full step rotation
‘D’ for anti-clockwise half step rotation
‘O’ for Motor Off

Remember, there must be only one letter and it must be written at the first position of the text editor. Also, the key-letter is to be written in Capital.

Cost

Name of the Component	Price(BTD)
1.Transformer	120
2.Bridge rectifier	40
3.7805IC	7
4.Heat sink	35
5.RS232 Connector	250
6.RS232 Male-female cable	200
7.MAX 232	65
8.ATMEGA 32	385
9.Stepper Motor	230
10.ULN 2003	23
11.Capacitors(8 piece)	25
12.Wire	30
13.Bread Board(2 piece)	300(150*2)

Total **1710**

Achievements

1. Thorough Idea of Microcontroller Programming
2. Developing System Using Microcontroller
3. Application of Stepper Motor in Practical purpose
4. Idea of remote-control of systems using Internet
5. Learning C# programming Language
6. Team Work