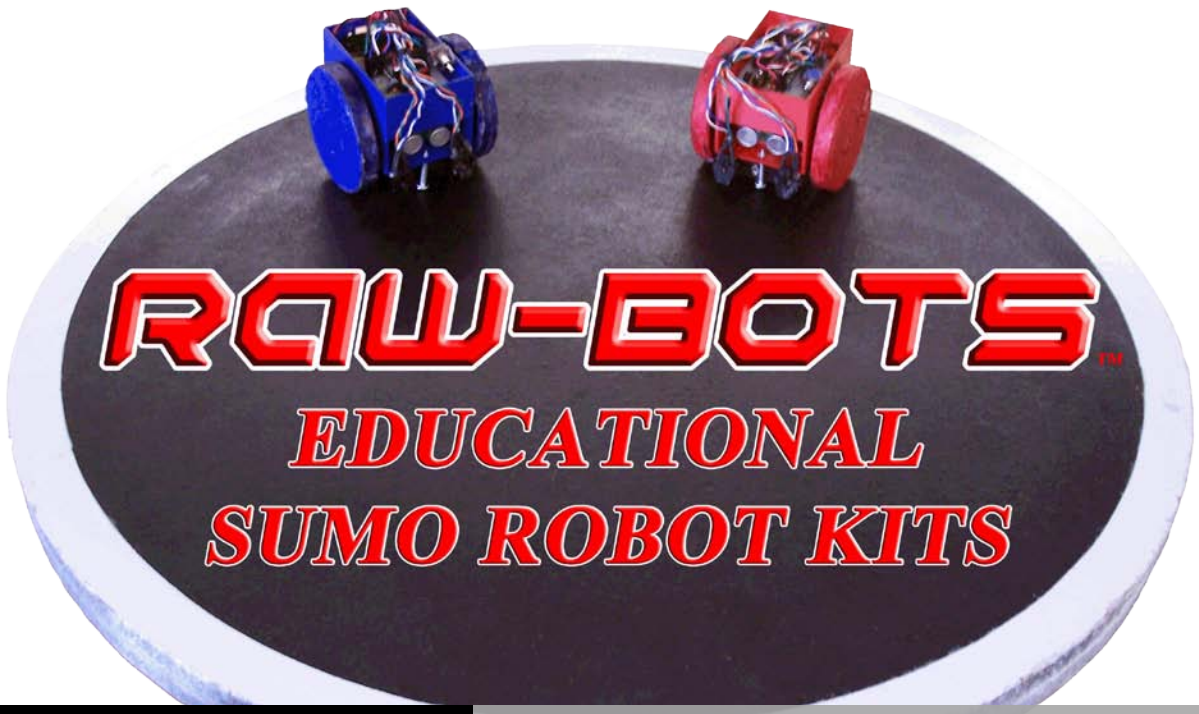


RAW-BOTS CIRCUIT BUILDER MANUAL

SUMO RAW-BOTS

RAW-Bots

"Circuit Builder Manual"

Welcome to the world of Robotics! In this kit, you will find all the components you need to design and build your own RAW-bot. As the word RAW might indicate, this special robot kit is not manufactured in a factory somewhere with all the electronics and components hidden away inside a plastic shell. Instead, this kit is designed for YOU to build your own robot from the inside out. *Wire-up the circuit board, plug in the components, power up the chip, write and download your own program, and then see the motors respond from the sensors just as you commanded.*

As a student in the world of robotics, this kit will give you a taste of many different types of engineering: Electrical, Mechanical, Computer Programming, Industrial (where you have to build your RAW-bot utilizing a budget and time-line), and of course Robotic Engineering.

So, explore, learn and use your imagination to bring your RAW-Bot to life from the inside out…

Getting Started…

Let's take a minute to learn and inventory all the components in your RAW-bot kit.

Check off the boxes below once you have located each item.

RAW-Bots Main Components

- ☐ Basic X Microprocessor and Circuit Board
- ☐ 2 Servo Motors
- ☐ Power Switch
- ☐ RS-232 Download Cable
- ☐ AA Battery Holder
- ☐ 9 Volt Clip
- ☐ 2 Touch Sensors
- ☐ 1 Distance IR Sensor (only in "RAW-Bot SUMO" kit)
- ☐ 2 Black/White Infra-Red (IR) Sensors (only in "RAW-Bot SUMO kit)

RAW-Bots Electrical Circuit Components

Main Circuit:

- ☐ 6 - Yellow Jumpers
- ☐ 2 - Orange Jumpers
- ☐ 1 - Long Red Jumper (2.0 in)
- ☐ 1 - Bare Jumper (very small)
- ☐ 2 - 0.1 uF Capacitors (yellow)
- ☐ 1 - 120 uF Capacitor (brown)
- ☐ 2 - Resistors

Now it's time to Build your Circuit!

1) ***Mark-out component locations***. Using a fine point marker, mark out your circuit as shown below. *(Be sure to count very carefully the number of holes over each component is from a reference point so as to match the picture below!)*

Components:

PS = Power Switch

9V = 9 Volt Clip

AA = AA Battery Pack

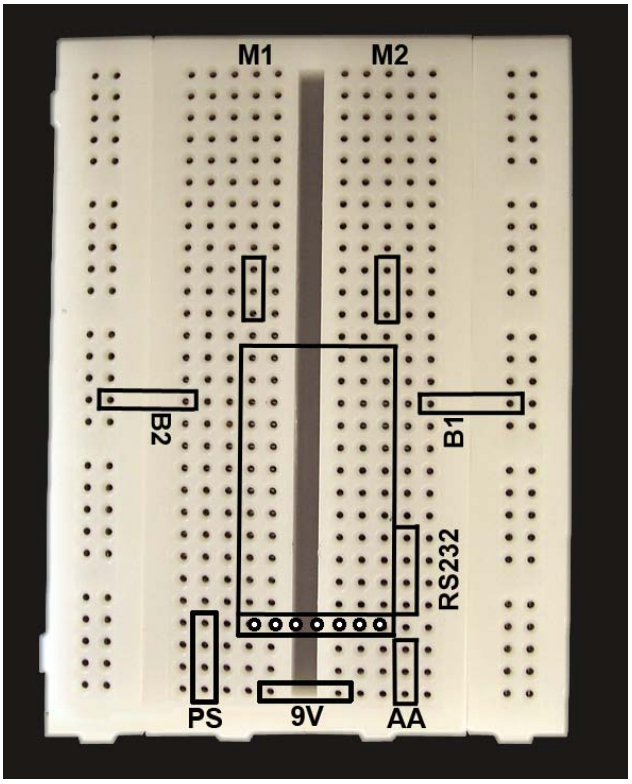
RS232 = Down load Cable

B1 = Bump Sensor 1

B2 = Bump Sensor 2

M1 = Motor 1

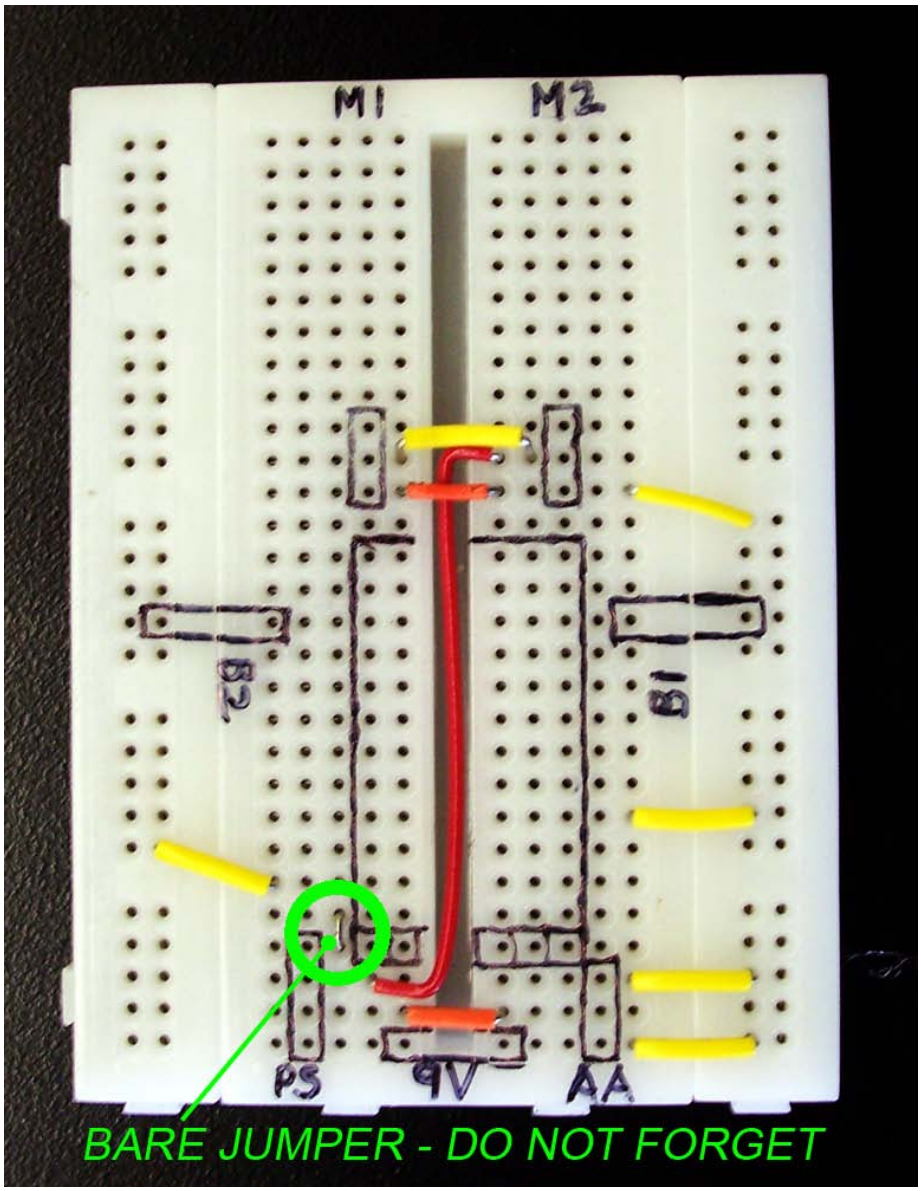
M2 = Motor 2



2) ***Install Jumpers:*** Before we begin to install your main components, we need to first install your jumpers in the circuit as shown below. *(Again, be very careful when positioning the jumpers by counting from a reference point on your circuit to match the picture below.)*

Qty:

1 - Bare Jumper = 0.1 inch (THIS IS A VERY IMPORTANT JUMPER, DO NOT FORGET)
6 - Yellow Jumpers = 0.4 inch
2 - Orange Jumpers = 0.3 inch
1 - Long Red Jumper = 0.2 inch



3) ***Install Capacitors and Resistors: Install Jumpers:*** Once you have installed your jumpers, it is time to install your capacitors and resistors.

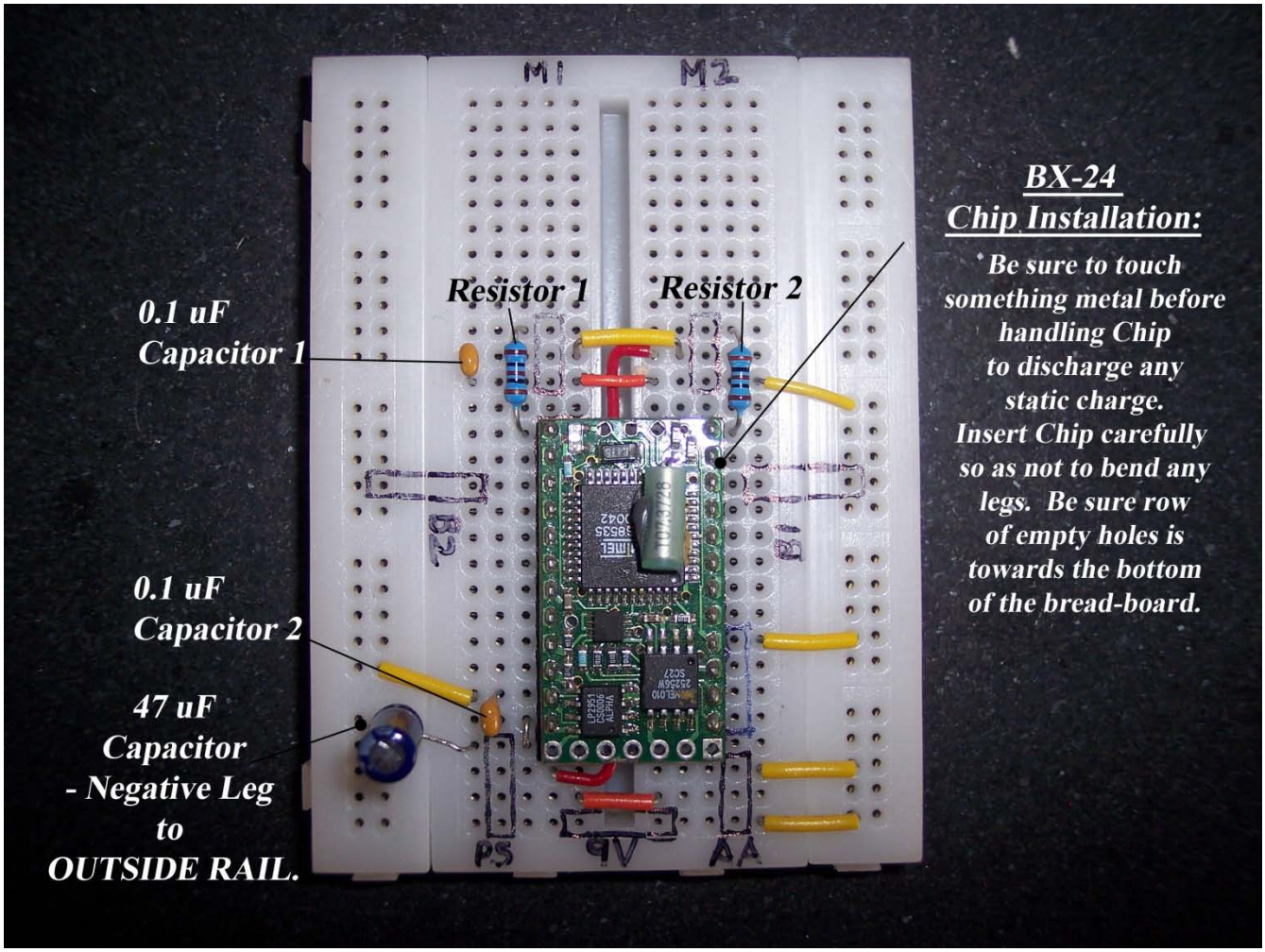
Qty:

1 – 120 uF Capacitor – Purple, larger Capacitor (WITH A NEGATIVE LEG).

3 – 0.1 Uf Capacitors – Yellow, smaller Capacitor (without a negative leg.)

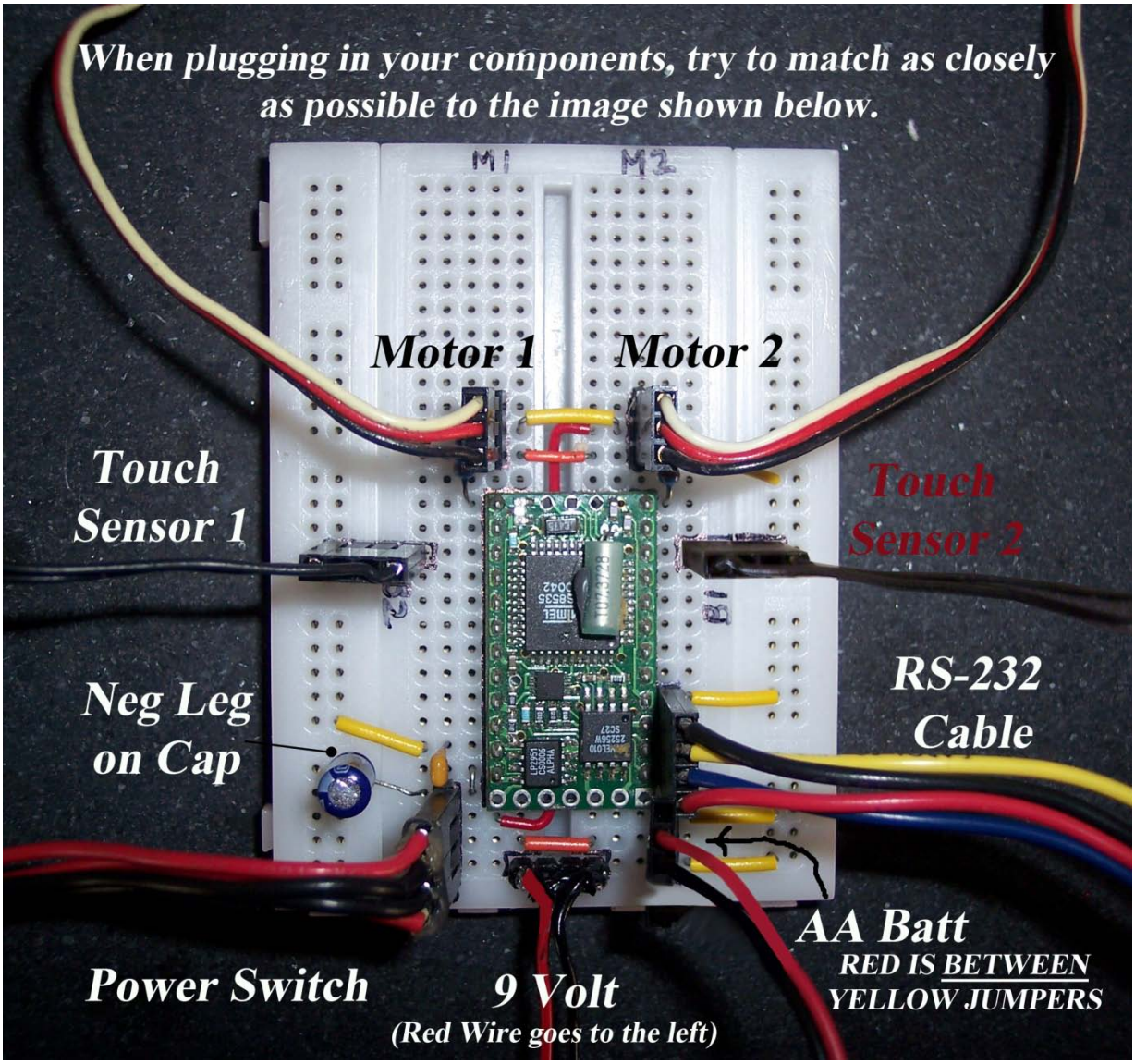
1 - BX-24 Microprocessor Chip - **BE SURE TO DISCHARGE ANY STATIC CHARGE ON YOUR BODY BEFORE HANDLING CHIP. DO THIS BY TOUCHING SOMETHING METAL FIRST.**

INSERT CHIP CAREFULLY SO AS NOT TO BEND ANY LEGS. MAKE SURE ROW OF EMPTY HOLES ARE TOWARDS THE BOTTOM OF THE BREAD-BOARD

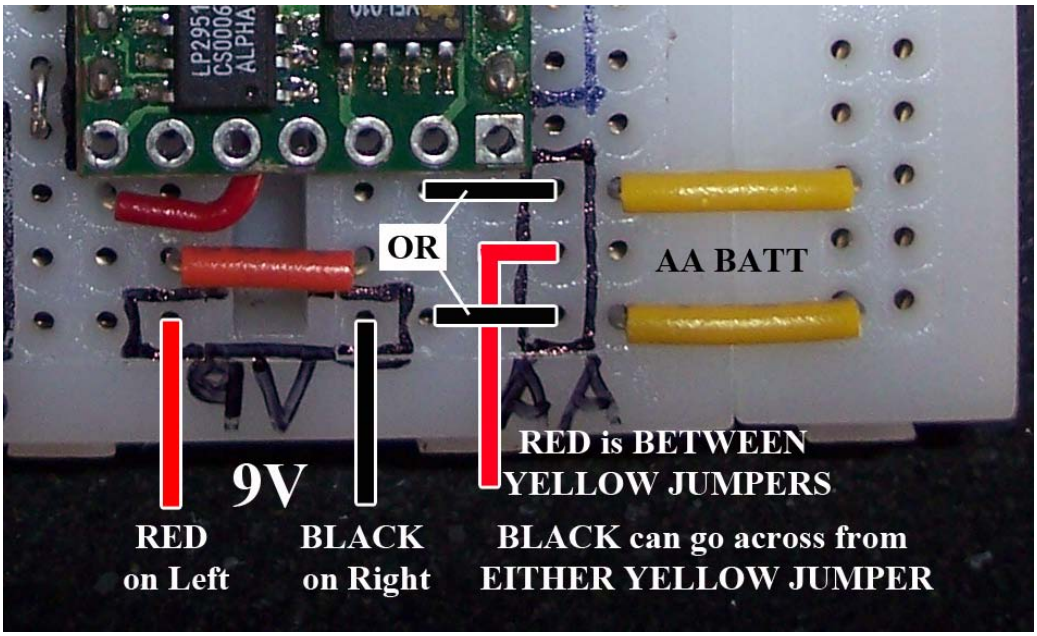


4) ***Install all external components to the circuit.*** Lastly, plug in all of your

External components into your board as shown below. (Pay close attention to the orientation of the components with respect to what color wires go where. Incorrect orientation of the wires will result in "unworking" devices.



NOTE: The AA Batt Plug is a '3' position Plug. Just be sure that the black wire of this plug is across from one of the yellow jumper wires. (To know you are correct, the red wire will be between the 2 yellow jumpers) Which ever of these yellow jumpers is across from the black wire of the AA plug will GROUND the entire circuit. SEE CLOSE UP PICTURE NEXT PAGE!



ALWAYS PLUG IN YOUR 9 VOLT AND AA BATTS LIKE ABOVE.

Your main circuit is now complete. Your next step is to input the batteries, download the starter code and determine if your motors will turn and respond when you press the bump sensors. If not, check your circuit for accuracy and

CHECK YOUR BATTERY VOLTAGES. Great work! It is programming time.

Downloading your Program (Code)

Once your circuit has been built and all of your components plugged in, it is now time to add your batteries to power the circuit.

You will need: Qty: 1 9 Volt Battery and Qty: 4 AA Batteries.

The 9 Volt powers your microprocessor and the AA's power your motors.

1) Plug in the batteries and turn on your chip. The chip may or may not have a flashing light at this point. (It depends on whether the last user turned on a light on the chip in their downloaded program. You will be over-writing their old program). AGAIN, if the chip is used and there is no light, THAT IS OK.

IF the chip is brand new, you should see flashing red and green lights at this point running a program called 'Hello World' from the factory. If not, then you need to check your circuit focusing on your batteries and power switch.

2) Now that the chip has power, plug the download cable (RS-232) into a COM PORT on your computer. In most cases it will be COM 1. *(however on laptops, many times your COM ports are 3. Also, you may have to purchase a "USB to COM port" cable. This cable is around $10 and requires some installation with simple software installs. You can purchase this at most major electronic stores like FRY's Electronics.)*

3) IF YOUR COMPUTER ALREADY HAS THE BASICX SOFTWARE, SKIP THIS STEP.

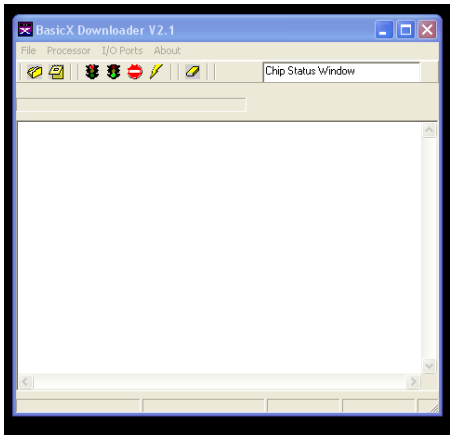
   Now download and install the program for writing the code for the BX-24 microprocessor called BASICX.

   Go to www.basicx.com - downloads – Complete Setup 1 of 1 – 21 MB

   This program is free.

4) Now open Basic X Programming Software.

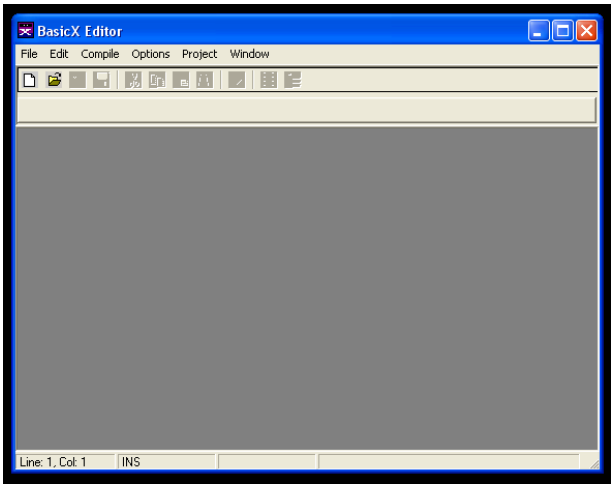
You will first see the Download/Communication Screen pop-up.



Download and Communication Screen

This is the screen where you select your download port for communicating with the chip and where you will see data back from the chip such as sensor information. For this SUMO kit, you will see your Sonar Sensor and Downward looking IR Sensor data print to this screen. NOTE: IGNORE THE WARNING 'DATA SERIAL ERROR'… IF YOUR CODE WILL NOT DOWNLOAD, CHECK YOUR BATTERIES AND WIRING OF YOUR CIRCUIT, FOR THIS IS WHERE THE PROBLEM USUALLY LIES.
Go to: FILE – OPEN – EDITOR

A second screen will now pop up as shown below called the EDITOR.

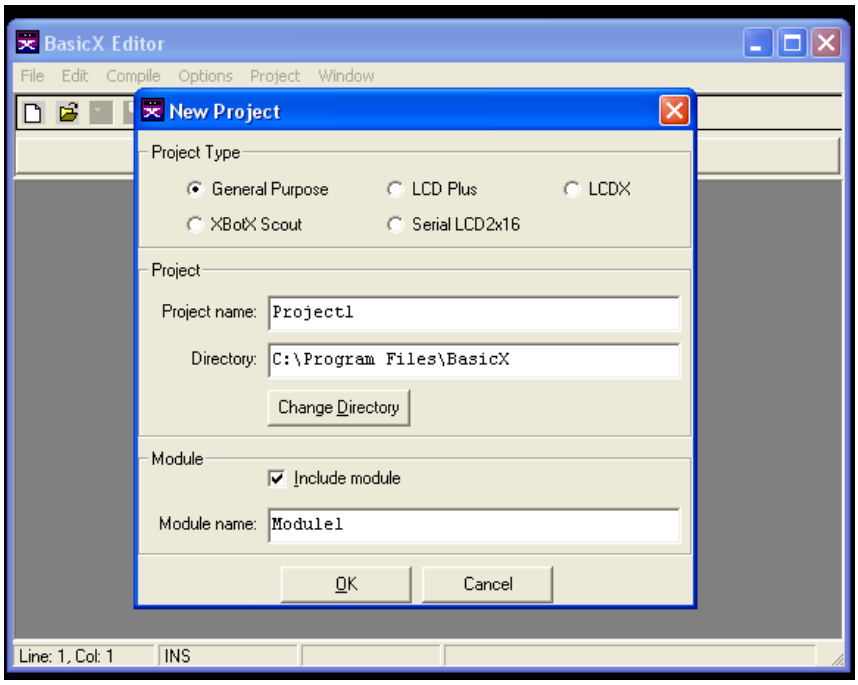


Editor – Software writing screen

The Editor Screen is the place where you create, modify, compile and initiate the downloading of the program to your microprocessor.

Your first need to create your Project Name.

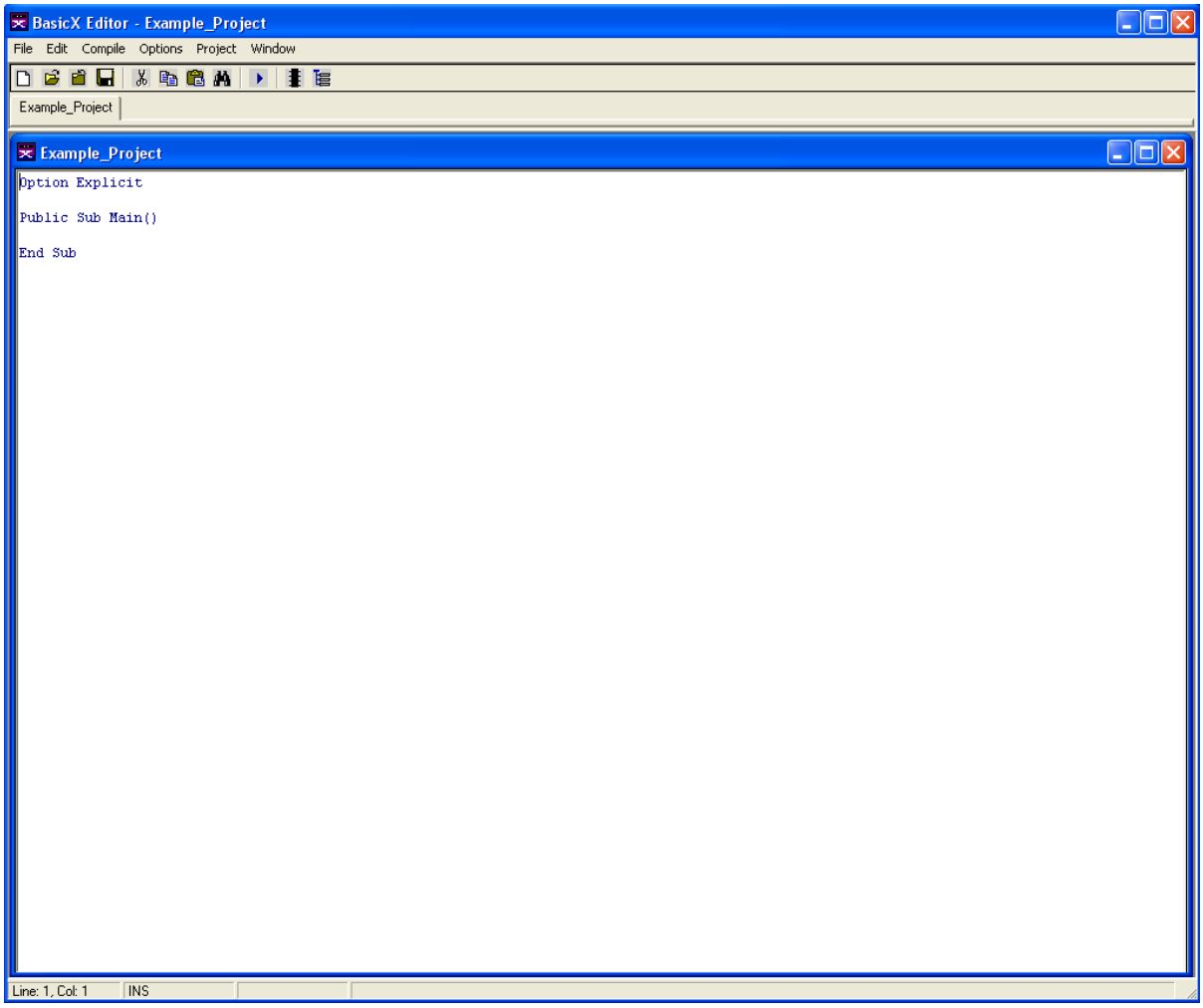
In the Editor Screen – goto FILE – NEW PROGRAM



This is important: <u>Name your Project AND Module THE SAME NAME (with no spaces)</u>

(And, it is suggested you use a unique name. NOT something like: 'Team 1', for example, because if someone else uses that same name, your program could link to the wrong code.)

You will now see the window below:



Select and DELETE the 3 lines show in the window above to clear the programming space and COPY and PASTE the program shown on the next page of this manual into the blank white Editor space.

‘RAW-BOT BASIC CODE

STAGE 1: The goal of this stage is make your motors respond when a touch sensor is hit. We have included a special program specifically for this purpose. THE NAME OF THE PROGRAM IS: ‘RAW-BOT TEST MOTORS CODE’.

NOTE: This program is temporary and is used to test your motors ONLY. (A new program will be downloaded later called ‘RAW-BOT SUMO CODE’)

The program below is for reference only. The actual program will be supplied by your instructor or will come on a CD included with this kit.

RAW-BOT TEST MOTORS CODE IN BLUE BELOW.

DO NOT copy and paste the Program into BasicX FROM THIS DOCUMENT. Use file from your instructor.

```
 Attribute VB_Name = "Module1"

Option Explicit

'Below will create space called Buffers so the processor can print data to the ‘ Download Window in BasicX

        Public OutBuffer(1 To 50) As Byte

        Public InBuffer(1 To 50) As Byte

'Below will create what is called a STACK where the processor can store commands for the servo motors.

Dim MotorStack(1 To 40) As Byte

'Created below are 2 Global Variables to store motor speed info for the servo motors.

        Public ServoLT As Single

        Public ServoRT As Single

Sub Main() 'All programs have ONLY one Sub Main. This is where the IF statement that controls behavior of
‘your RawBot is placed.

' The next three commands established communication with the computer.
' First, it establishes buffers, and then it opens communications
' to transmit the contents to Com 1 at 19200 baud rate.

Call OpenQueue(OutBuffer, 50)
Call OpenQueue(InBuffer, 50)
```

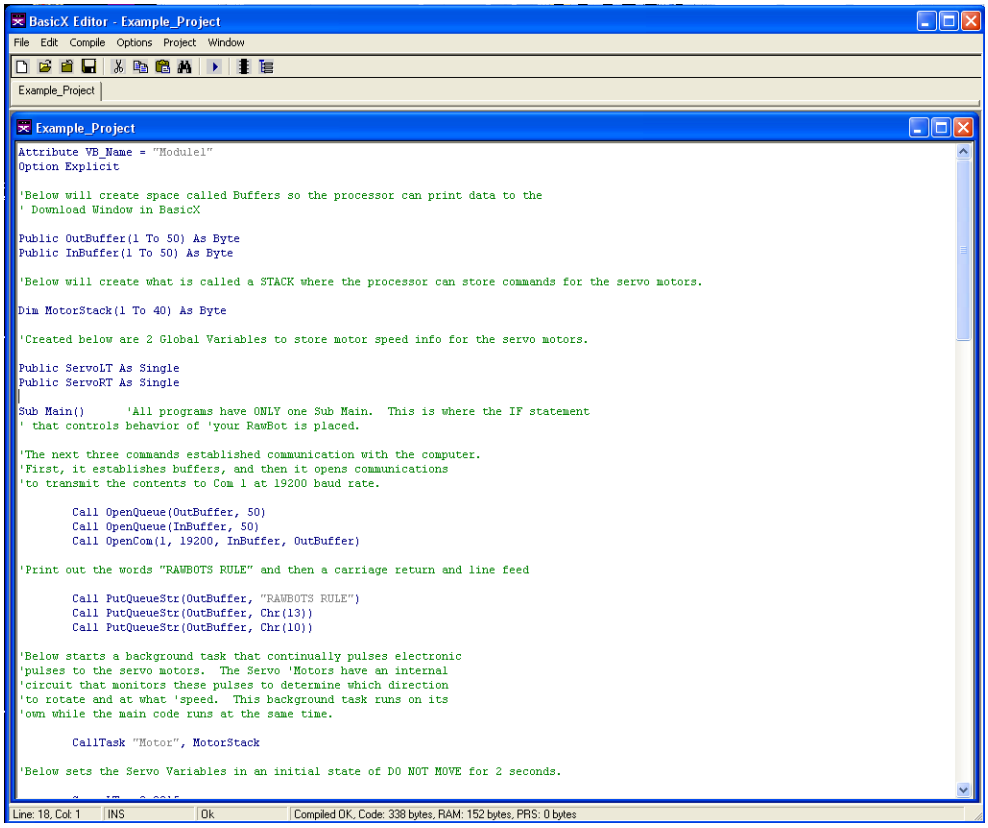
```
Call OpenCom(1, 19200, InBuffer, OutBuffer)

'Print out the words "RAWBOTS RULE" and then a carriage return and line feed

Call PutQueueStr(OutBuffer, "RAWBOTS RULE")
Call PutQueueStr(OutBuffer, Chr(13))
Call PutQueueStr(OutBuffer, Chr(10))

' Below starts a background task that continually pulses electronic
' pulses to the servo motors. The Servo
' Motors have an internal
' circuit that monitors these pulses to determine which direction
' to rotate and at what ‘speed. This background task runs on its
' own while the main code runs at the same time.


CallTask "Motor", MotorStack

'Below sets the Servo Variables in an initial state of DO NOT MOVE for 2 seconds.

        ServoLT = 0.0015
        ServoRT= 0.0015

        call delay(2.0)

' Below calls the subroutine for moving forward.

        call forward

' We now enter the Main DO LOOP with an IF statement inside it.
' The program starts at the DO of DO LOOP and repeats any code inside
' the Loop over and over until the chip is switched off. This means the
' IF statement is repeated over and over. The IF statement is the BEHAVOIR
' of our RawBot. If the line of the IF statement is true, it enters that portion
' of the IF statement and executes the code inside.
' The function GetPin() will look at a pin’s state. Ie if the pin is grounded
' or ZERO, the IF statement is TRUE. In plain English, this means the Touch
' Sensor is pressed and is sending a ground reading to the pin.

DO

‘Below is a basic IF statement to test the functioning of your motors.

If getpin(10) = then  ‘this means one of the touch sensors is hit

        Call left
        Call delay(1.0)
        Call forward

Elseif getpin(15) = 0 then  ‘other touch sensor is hit
```

```
        Call right
        Call delay(1.0)
        Call forward
End if

Loop

End Sub

'Main program is now closed ... below are the Subroutines

Sub Forward() 'Remember the pulse rate to say STOP for the Motors is 0.0015.

' If you add 0.0002 to this rate it moves in one
' direction at full
' speed and if you subtract 0.0002 from this rate, you move in the
' other direction at full speed

        ServoLT = 0.0015 + 0.0002
        ServoRT = 0.0015 - 0.0002

End Sub

Sub Reverse()

        ServoLT = 0.0015 - 0.0002
        ServoRT = 0.0015 + 0.0002

End Sub

Sub Right()

        ServoLT = 0.0015 - 0.0002
        ServoRT = 0.0015 - 0.0002

End Sub

Sub Left()

        ServoLT = 0.0015 + 0.0002
        ServoRT = 0.0015 + 0.0002

End Sub

Sub Stop()

        ServoLT = 0.0015
        ServoRT = 0.0015
```

```
End Sub


Sub Motor()
 'This task runs in the background sending pulses to the left and 'right servos. The global ‘variable ServoLT
‘controls the left 'servo, and the global variable ServoRT controls the right servo. When ServoLT and ServoRT
‘are set to 1.5e-3, then the wheels are stopped. By changing the pulse width greater than or less than that, the
‘motors turn.

Do ' Send a pulse on pin 12 for the left servo.

        Call PulseOut(12, ServoLT, 1) ' This is to produce a pulse rate of about 50 Hz.

        Call Delay(0.01) ' Send a pulse on pin 13 for the right servo.

        Call PulseOut(13, ServoRT, 1) ' This is to produce a pulse rate of about 50 Hz.

        Call Delay(0.01)

Loop

End Sub
```

DO NOT copy and paste the Program into BasicX FROM THIS DOCUMENT.

Your Program should now be in the Editor window with your unique Project and Module Name.



HOW TO SAVE YOUR PROGRAM.

You have now created your Project and Module. Just so you know, each project has a module attached to it. It is MODULE that actually contains your code this is why it is so important you name it the same name as your Project. It is the module that is the important part.

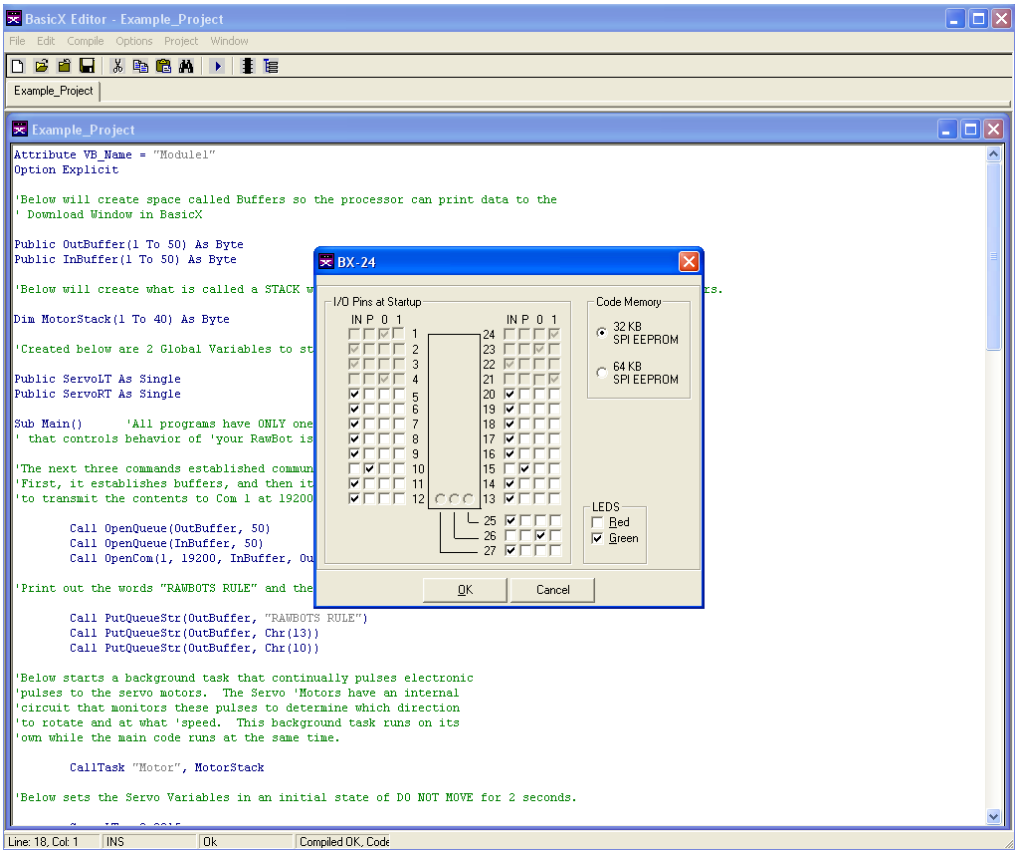
SO, when saving your project…. Be sure to ALSO save your module each time. Ie

Save Project AND Save Module each time. (it is recommended you back up your code on an external jump drive as well.)

HOW TO SET YOUR PINS. (IMPORTANT!)

When you are using sensors with the BX-24, often you have to 'Set the Pins' for the Microprocessor to correctly recognize with pins do what for the attached sensor. At this time, the TOUCH SENSOR will need to have the 'P' column checked inorder for the Chip to see the sensor. **IF YOU DO NOT SET THE PIN FOR THE TOUCH SENSOR, THE CHIP WILL THINK THE SENSOR IS BEING HIT ALL THE TIME**.

GO TO – PROJECT – CHIP –To open up the screen to set the pins. SEE BELOW for the Pin Setting Window.



Notice the 'P' column is checked for pins 10, and 15. DO THIS.

This is where the TOUCH SENSORS are located in the circuit. This also matches the GETPIN commands in the IF statement in the code. (by the way, P stands for Pull-UP resistor. It activates an internal resistor in the chip to see the TOUCH SENSOR. (notice the GREEN light is checked in this example)
YOU WILL HAVE TO SET MORE PINS FOR YOUR Sonar Sensor and IR Sensors LATER IN THIS MANUAL.
HOW TO DOWNLOAD YOUR CODE

- Before downloading, check the following is complete:

1) Double check that your circuit is correct.
2) Double check your batteries are new and plugged into the circuit correctly.
3) Your Program and Module is created and saved with the SAME NAME.
4) YOUR PINS ARE SET for the Touch Sensors.
5) Your Motors are plugged in correctly. (The white wire of the motor should interface with the resistor going to the pins 12 or 13.)

- If all of the above 5 are correct, located the COM port on your computer and plug in your RS-232 cable.

- Turn on the Switch connected to the breadboard.

- Now…. With the EDITOR SCREEN Active (not the Download Screen) GoTo – Compile – Complie-Run.

Your program should download to the Chip and whatever color light you chose in the PIN SETTING step should turn on. If not, check your batteries and connections. 99% of the download problems is from incorrect wiring.

NOTE: YOUR SWITCH IS A TWO-WAY SWITCH. ONE DIRECTION TURNS ON ONLY THE CHIP, THE OTHER DIRECTION TURNS ON BOTH THE CHIP AND THE MOTORS.

LOOK BELOW TO TEST YOUR MOTORS AND PROGRAM TO THIS POINT!!!

With your POWER SWITCH in the correct direction to power the motors, your motors should not move for 2 seconds and then spin after that time.

IF YOU DID EVERYTHING CORRECT, hitting the Touch Sensors should make the motors reverse direction.

- Hit each Touch Sensor and watch the motors reverse direction.

CONGRATULATIONS! If you got this far, you have a simple functioning RAW-BOT!

IF YOUR MOTORS RESPOND TO THE TOUCH SENSORS, MOVE ON TO THE SENSORS.
IF NOT

1) CHECK YOUR CIRCUIT
2) CHECK YOUR BATTERY VOLTAGES
3) MAKE SURE ALL OF YOUR COMPONENTS ARE PLUGGED INTO THE CORRECT SPOTS.

It is now time to hook up your DISTANCE and IR sensors.

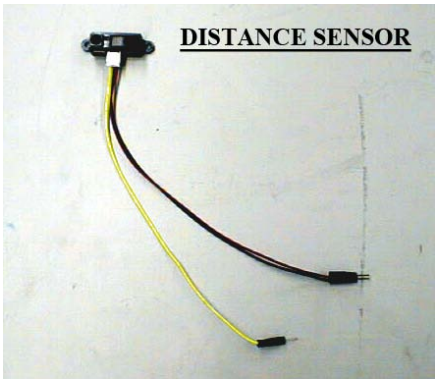
BUT FIRST, you have to build the 5 VDC Circuit To power the Distance Sensor.

HOW TO BUILD THE 5 VDC Circuit

5 Volt Regulator

NOTE: In your kit, locate the circuit called '5 VDC Circuit'. (It is in a small plastic bag labeled 5 VDC Circuit)

By adding the DISTANCE sensor, you will need to build a separate +5 Volt DC Circuit to power the sensor. (The 2 Downward looking IR's do not need a separate power source.)



This new 5 Volt Regulator circuit will power your DISTANCE SENSOR and any other added 5 Volt DC device up to 1 amp. In other words, you could theoretically add more than one sonar sensor. This 5V Reg circuit is powered from the same 9V as the chip and will turn on at the same time as the rest of the circuit.

Below is the 5 Volt Reg circuit draw out in real electrical language.

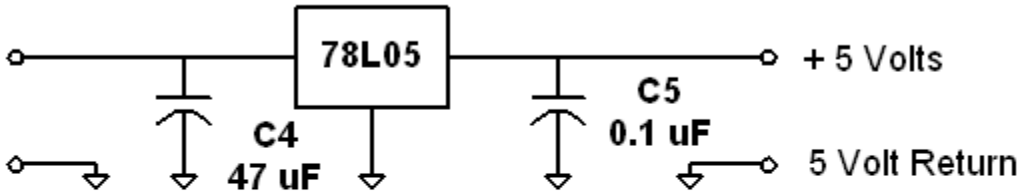
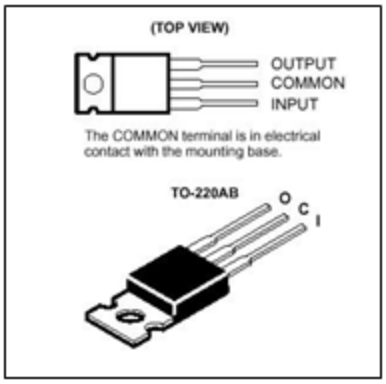
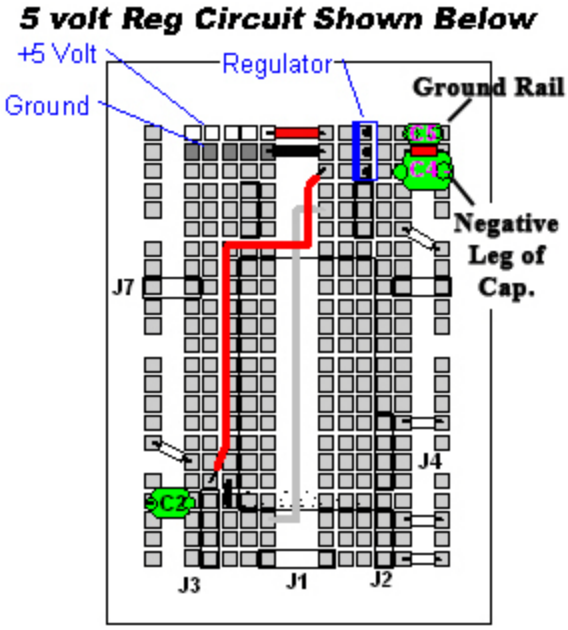


Figure 1. Circuit of Voltage Regulator





Build your circuit as shown in the diagram above.

CIRCUIT PARTS: ***5 Volt Regulator, 1 Yellow Jumper, 2 Orange Jumpers, 1 LONG Red Jumper, 1 -120 uF Cap, and 1 – 0.1uf Cap.***

FOLLOW the instructions below while referring to the figure above to plug in circuit components.

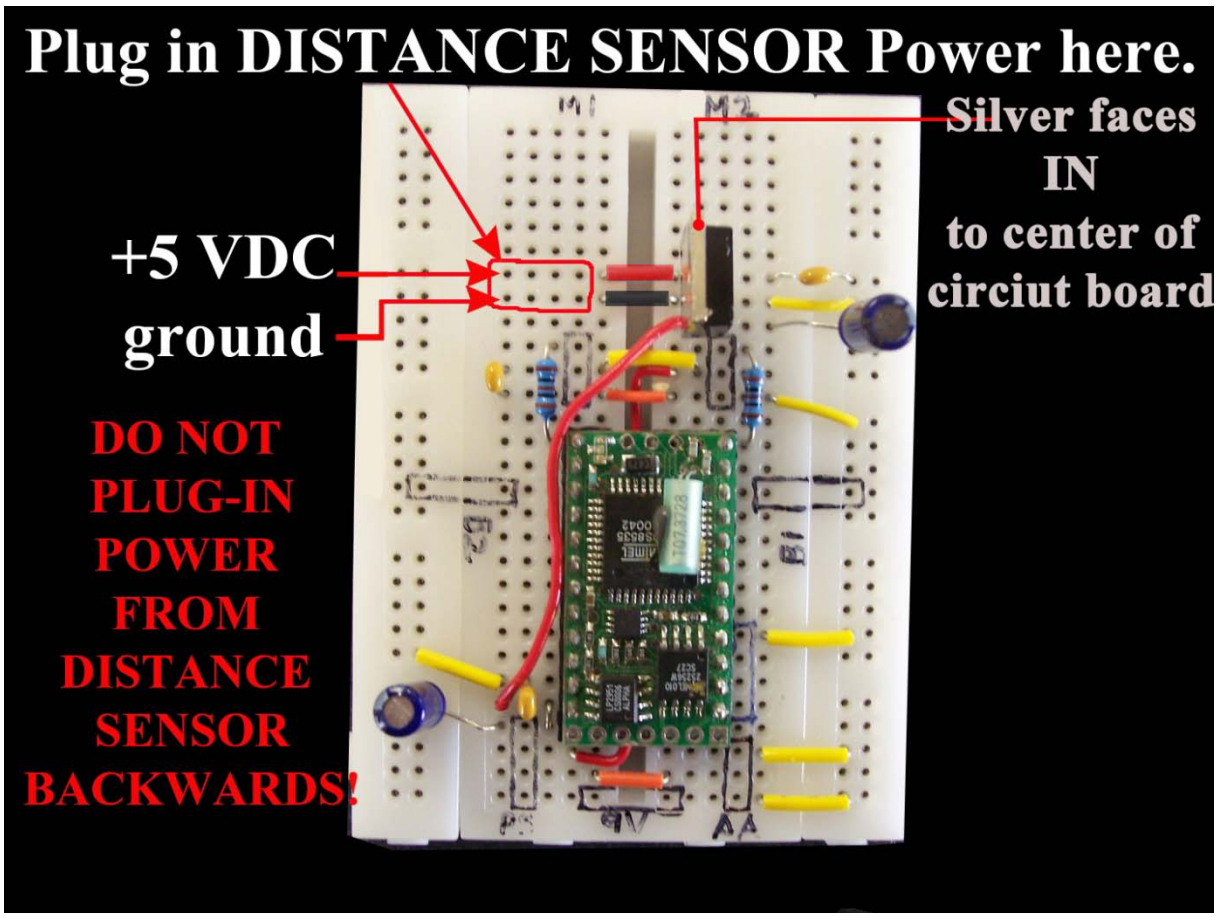
- Please locate your circuit components to build the 5V Reg circuit on the next 3 empty rows of the breadboard after where your motors plug-in.

- INSTALL long red jumper from pin 24 on your chip up to the first leg of the Regulator show in blue.

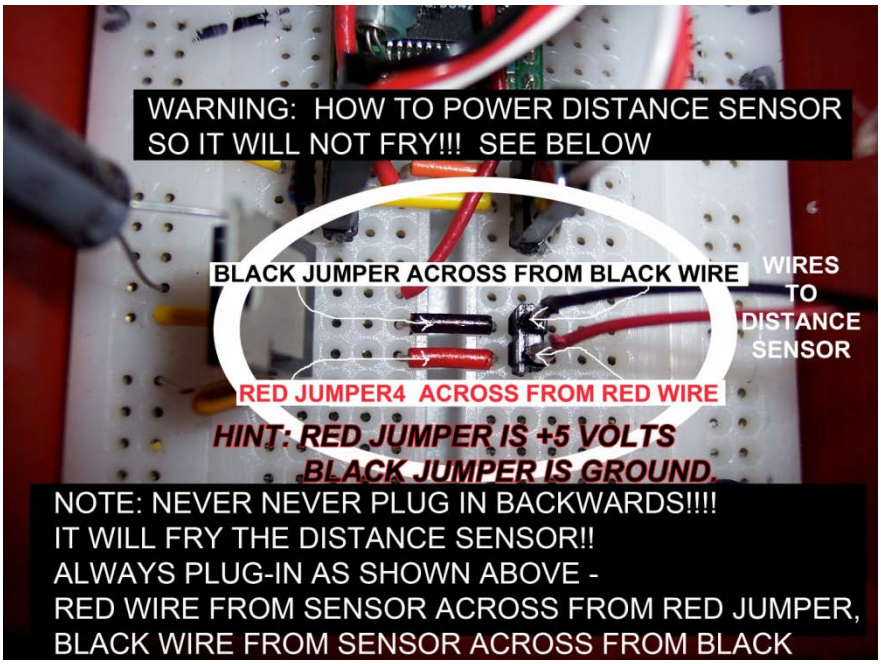
- INSTALL REGULATOR .(the metal, silver back of the regulator faces IN to the center of the circuit board.)
- INSTALL 120 uF Capacitor (take special note to put the NEGAVTIVE leg of the capacitor on the outside rail of the circuit board)
- INSTALL small red jumper from middle leg of Regulator to the outside, ground rail of circuit board.
- INSTALL small RED jumper and BLACK jumper across legs of Regulator as shown above.
- INSTALL small, yellow, 0.1uf CAP as shown above (starts at first leg of Regulator to ground

CON'T

Your finished 5 volt regulator circuit should look as shown below.

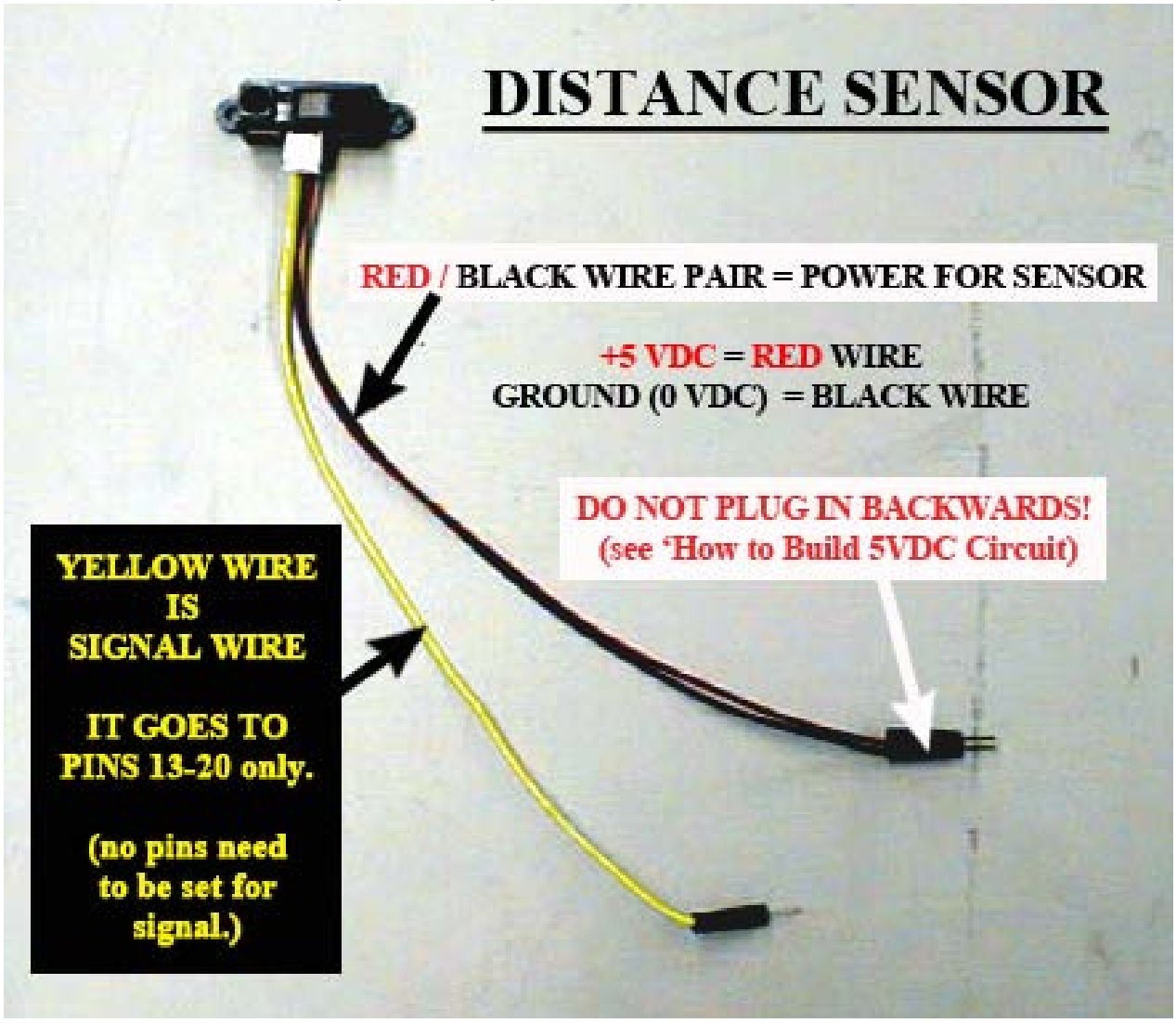


HOW TO PLUG IN DISTANCE SENSOR POWER BELOW:



Now, it is time to locate your DISTANCE SENSOR

DISTANCE (INFARED) SENSOR or DISTANCE SENSOR



DO NOT CONFUSE THE"*DISTANCE infared SENSOR (DIR) "* WITH THE "*DOWNWARD LOOKING IR"* USED TO DETECT THE WHITE EDGES OF THE SUMO RING. THEY ARE ENTIRELY DIFFERENT SENSORS.

Description: The Distance IR (DIR) sensor is an analog sensor that returns a voltage reading from 0.3VDC to 2.6VDC (depending on the distance to an object). The sensor shoots out a thin laser beam that reflects off an object, bounces back and is detected by the sensor. The sensor has a range of 10cm to 80cm. (Place the sensor in the center of your robot since your smallest reading is 10cm = 4")

Wiring of Distance Sensor.

Red = 5VDC (this wire plugs into the +5VDC of the 5V Regulator circuit built on your Bread-board.)

Black = 0VDC (this wire plugs into the 0 VDC of the 5V Regulator circuit built on your Bread-board.)

Yellow = 0-2.6VDC SIGNAL WIRE to BX-24. (this wire plugs into an ANALOG PIN on the BX24. This pin must be 13-20 since these are the analog pins on the BX-24.)

THIS DOCUMENT USES PIN 14 FOR THIS SENSOR.

Setting the Chip for the IR

GREAT NEWS! THERE ARE NO PINS TO SET FOR THIS SENSOR!

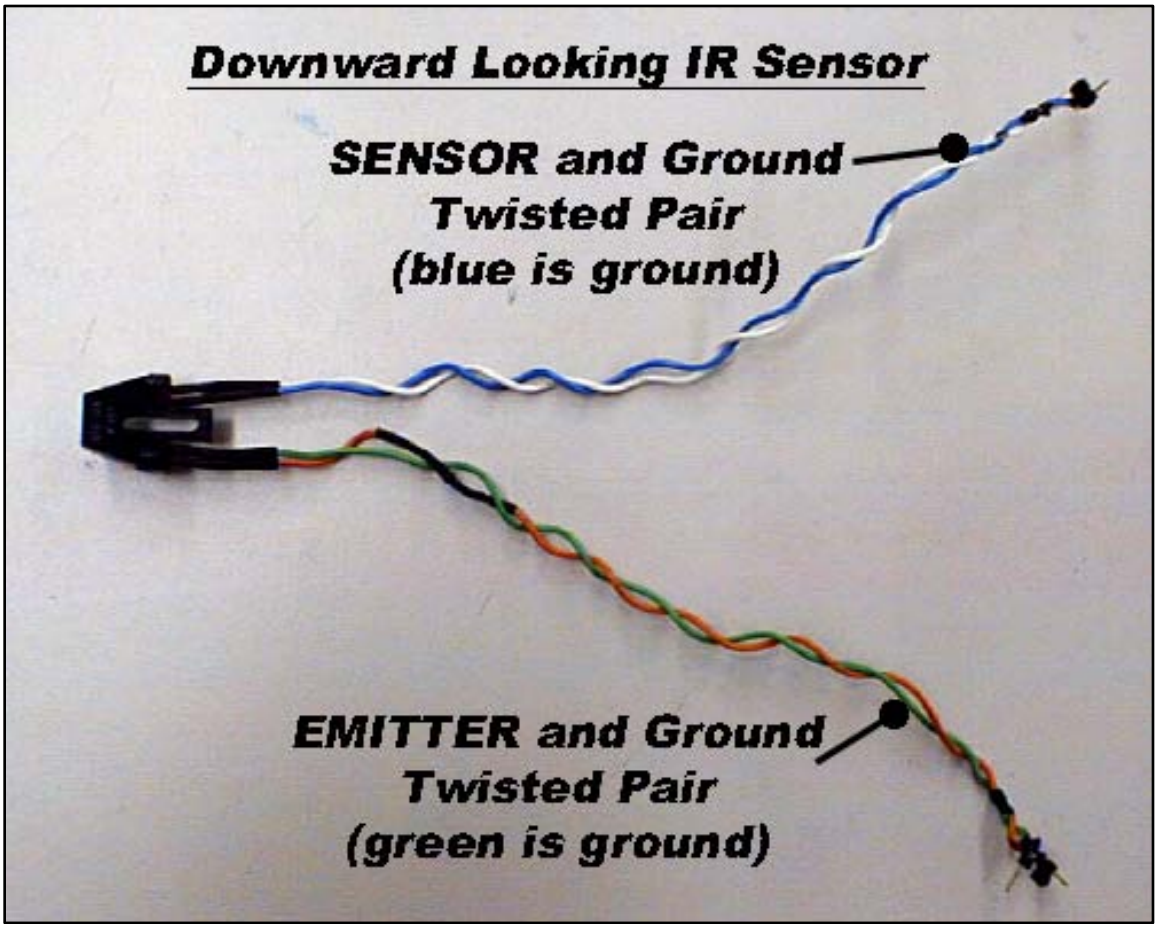
PLUG IN YOUR DISTANCE SENSOR WIRES

RED-BLACK wires go to 5 VDC Circuit.
Yellow Signal wire goes to a pin between 13-20, it is up to you.

Your Distance Sensor is now powered from the 5 Volt Circuit and the Signal wire Is connected to the BX-24.

It is now time to located and plug-in your Downward Looking IR Sensors.

DOWNWARD LOOKING IR SENSORS



*There are 2 parts to the IR sensor. The **EMITTER** and the **SENSOR.***

The EMITTER is an output and it sends out an infrared-beam which reflects off a surface.

The SENSOR detects the reflected beam from the Emitter and assigns a distance reading to the chip.

Wiring of the IR's (remember, there are 2 of these)

Orange = EMITTER + (Plug into the BX-24 Pin that YOU declare in your program.)

Green = EMITTER Ground (Plug into the 'Ground Rail' across from the pin chosen above.)

White = SENSOR + (Plug into the BX-24 Pin that YOU declare in your program.)

Blue = SENSOR Ground. (Plug into the 'Ground Rail' across from the pin chosen above.)

Setting the Chip for the IR in the BX-24 Editor.

EMITTER check the '1' column in the BX-24 Chip Dialog.

SENSOR check the 'P' column in the BX-24 Chip Dialog.

RECORD THIS INFORMATION ON THE SAME PIECE OF PAPER AS THE SONAR PINS.

Example of what to record on paper:

_____

DW IR #1

EMITTER #1 – Orange Wire – Pin 7 – Set the Pin as a '1'

Green Wire goes to the Ground Rail on the Bread Board

SENSOR #1 – White Wire – Pin 8 – Set the Pin as a 'P'

Blue Wire goes to the Ground Rail on the Bread Board.

DW IR #2

EMITTER #2 – Orange Wire – Pin 16 – Set the Pin as a '1'

Green Wire goes to the Ground Rail on the Bread Board

SENSOR #2 – White Wire – Pin 17 – Set the Pin as a 'P'

Blue Wire goes to the Ground Rail on the Bread Board.

Both of your DW IR's are plugged into your BX-24 and ready for use.

AT THIS TIME, ALL COMPONENTS ARE PLUGGED IN AND READY TO USE

IT IS NOW TIME TO DOWNLOAD THE "SUMO CODE" TO YOUR BX-24

EXAMPLE OF FINAL SUMO CODE BELOW: (IN GREEN)

DO NOT copy and paste the Program into BasicX FROM THIS DOCUMENT. Use file from your instructor.

```
Attribute VB_Name = "Module1"

Option Explicit

'Below will create space called Buffers so the processor can print data to the Download Window in BasicX

Public OutBuffer(1 To 50) As Byte

Public InBuffer(1 To 50) As Byte

'BLUE TEXT MEANS ADDED SUMO SENSOR CODE.

Dim VoltageReading As Single
Public RANGE as Single
Public VRFive as Single

Const DIRPIN As Byte = 14        '(you can use 13-20, this is the pin where the white wire is Plugged)

Const LT_DOWN_LED as Byte = 7
Const LT_DOWN_SENSOR as Byte = 9
Const RT_DOWN_LED as Byte = 15
Const RT_DOWN_SENSOR as Byte = 18

Const BLACK as Byte = 1
Const WHITE as Byte = 0
```

```
Dim MotorStack(1 To 40) As Byte
Dim SensorStack(1 To 40) As Byte

'Created below are 2 Global Variables to store motor speed info for the servo motors.

        Public ServoLT As Single

        Public ServoRT As Single

        Public StatusRight as byte

        Public StatusLeft as byte


Sub Main() 'All programs have ONLY one Sub Main. This is where the IF statement that controls behavior of
'your RawBot is placed.

' The next three commands established communication with the computer.
' First, it establishes buffers, and then it opens communications
' to transmit the contents to Com 1 at 19200 baud rate.

Call OpenQueue(OutBuffer, 50)
Call OpenQueue(InBuffer, 50)
Call OpenCom(1, 19200, InBuffer, OutBuffer)

'Print out the words "RAWBOTS RULE" and then a carriage return and line feed

Call PutQueueStr(OutBuffer, "RAWBOTS RULE")
Call PutQueueStr(OutBuffer, Chr(13))
Call PutQueueStr(OutBuffer, Chr(10))

' Below starts a background task that continually pulses electronic
' pulses to the servo motors. The Servo
' Motors have an internal
' circuit that monitors these pulses to determine which direction
' to rotate and at what 'speed. This background task runs on its
' own while the main code runs at the same time.

CallTask "Motor", MotorStack
Calltask "sensor", Sensorstack 'starts the back ground task for sonar reading

'Below sets the Servo Variables in an initial state of DO NOT MOVE for 2 seconds.

ServoLT = 0.0015
ServoRT= 0.0015

statusright = 0
```

```
statusleft = 0


        call delay(5.0)   'THIS USE TO BE 2.0

' Below calls the subroutine to start your robot spinning to look for other robot.

        call LEFT  'THIS USE TO BE FORWARD

' We now enter the Main DO LOOP with an IF statement inside it.
' The program starts at the DO of DO LOOP and repeats any code inside
' the Loop over and over until the chip is switched off. This means the
' IF statement is repeated over and over. The IF statement is the BEHAVOIR
' of our RawBot. If the line of the IF statement is true, it enters that portion
' of the IF statement and executes the code inside.

DO

'NOTE  BE SURE TO COMMENT OUT THE DEBUG PRINT LINES BELOW WHEN IN ACTAUL SUMO 'COMPETITION
SO AS NOT TO SLOW DOWN THE RUNNING OF THE CODE WHEN IN BATTLE.

Debug.print " DIR = "; cstr(RANGE)  'prints the value of the Distance Sensor to the screen

        delay(0.01)

debug.print " status right = "; cstr(statusright)

        call delay(0.01)

debug.print " status left = "; cstr(statusleft)

        call delay(0.1)


if (statusright = 1) then  'white line is found, stop, backup and spin again to find other robot

        call halt
        call delay(0.2)
        call reverse
        call delay(0.5)
        call left

elseif (statusleft = 1) then  ''white line is found, stop, backup and spin again to find other robot

        call halt
        call delay(0.2)
        call reverse
        call delay(0.5)
        call right
```

```
elseif range > 10.0 Then 'Turns Left looking for other robots.

        Call left


ElseIf range < 10.0 Then ' the other robot is found, go forward and push it!

        Call forward

End If

Loop

End Sub

'Main program is now closed ... below are the Subroutines

Sub Forward() 'Remember the pulse rate to say STOP for the Motors is 0.0015.

' If you add 0.0002 to this rate it moves in one
' direction at full
' speed and if you subtract 0.0002 from this rate, you move in the
' other direction at full speed

        ServoLT = 0.0015 + 0.0002
        ServoRT = 0.0015 - 0.0002

End Sub

Sub Reverse()

        ServoLT = 0.0015 - 0.0002
        ServoRT = 0.0015 + 0.0002

End Sub

Sub Right()

        ServoLT = 0.0015 - 0.0002
        ServoRT = 0.0015 - 0.0002

End Sub

Sub Left()

        ServoLT = 0.0015 + 0.0002
        ServoRT = 0.0015 + 0.0002
```

```
End Sub

Sub halt()

        ServoLT = 0.0015
        ServoRT = 0.0015

End Sub



Sub Motor()
 'This task runs in the background sending pulses to the left and right servos. The global variable ServoLT
'controls the left servo, and the global variable ServoRT controls the right servo. When ServoLT and ServoRT are
'set to 1.5e-3, then the wheels are stopped. By changing the pulse width greater than or less , the motors turn.

Do ' Send a pulse on pin 12 for the left servo.


        Call PulseOut(12, ServoLT, 1) ' This is to produce a pulse rate of about 50 Hz.

        Call Delay(0.01) ' Send a pulse on pin 13 for the right servo.

        Call PulseOut(13, ServoRT, 1) ' This is to produce a pulse rate of about 50 Hz.

        Call Delay(0.01)

Loop

End Sub



Sub sensor()

Do

            Call GetADC(DIRPIN, VoltageReading)

   VRFive = voltagereading*5.0      'multiplies voltage reading by 5
        RANGE = ((34.633/VRFive) - 5.162)/2.54      'gives distance in inches

        Call Delay(0.1)


'START INFRARED ******************************

IF GetPin(RT_DOWN_SENSOR) = 0 THEN
```

```
        Statusright = black

        call delay(0.01)

Elseif getpin(RT_DOWN_SENSOR) = 1 THEN

        Statusright = white
        call delay(0.01)

end if

if getpin(LT_DOWN_SENSOR) = 0 THEN

        Statusleft = black
        call delay(0.01)

Elseif getpin(LT_DOWN_SENSOR) = 1 THEN

        Statusleft = white
        call delay(0.01)

end if

loop

end sub
```

END OF COMPLETE SUMO CODE

At this time, you should have enough knowledge and experience to be able to bring your Raw-Bot SUMO to life. Be creative with your Robot Frame and have fun.

Time to SUMO!

HOW TO CHECK YOUR DISTANCE SENSOR (DIR) AND IR'S WITHOUT RUNNING THE MOTORS.

Before running the motors, it is a great idea to test your DIR and 2 IR sensors by PRINTING their values to the screen. To do this, simply comment out the entire IF statement in the Main Program DO-LOOP and only run the Debug.Print commands.

This WILL ensure only the values of the DIR and IR are printed to the screen and no other code is run. Below shows how to comment out the IF statement.

(note, to comment out code, put a ‘ infront of the line of code. That way it is there but not run by the BX-24

```
DO
   debug.print " range value  = "; cstr(range)   ' This prints the range value to the screen
   delay(0.01)

        debug.print " status right = "; cstr(statusright)
        delay(0.1)

        debug.print " status left = "; cstr(statusleft)
        delay(0.1)

        ‘          If statusright = 1 then
        ‘                  call halt
        ‘                  call delay(0.5)
        ‘                  call reverse
        ‘                  call delay(1.75)
        ‘          elseif statusleft = 1 then
        ‘                  call halt
        ‘                  call delay(0.5)
        ‘                  call reverse
        ‘                  call delay (1.75)

       ‘ elseIf ((statusright = 0) and (statusleft = 0) and (range > 5.0)) Then
           'Back up for .3 seconds
         ‘  Call left
      'When the left bump switch is closed,  the GetPin(14)
      'will return a 0
                ‘elseif ((statusright = 0) and (statusleft = 0) and (range < 5.0)) then
                ‘          if (statusright = 1) or (statusleft =1)  then
                           ‘          call halt
                           ‘          call delay(1.0)
                           ‘          call left
                           ‘end if
                           ‘call forward
   ‘  End If
LOOP
```

Notice, now the only code that will be executed in the Main Do-Loop is the printing of the DIR and IR values.
Once you have the IF statement commented-out, download and see the printing of the DIR and IR sensors on the DownLoad Screen of Basic X software similar to below:

EXAMPLE PRINTING RESULTS:

Range Value = 25.545
Status Right = 0
Status Left = 1

Remember, Range Value should change as you move it closer and farther from an object. Status Right = '0' means it is on black and Status Left = '1' means it is 1/8" away from a white surface. If you do not get these results, check your batteries and wiring and code.

NOW IT IS TIME TO SUMO!

APPENDIX

HOW TO TUNE YOUR MOTORS

It is possible that your motors come 'out of tune' as you use them over a period of time.
How do you know if your motors are 'out of tune'? It is very simple.

Since your motors are designed NOT to move when you send a pulse rate of 0.0015, IF they move or 'creep' when you send this pulse rate, your motors are out of tune. (NOTE, when you first turn on power to your motors, they will SURGE and then stop… this is normal and not necessarily indicating your motors are out of tune. Your motors are out of tune ONLY if they continually move when being sent a 0.0015 pulse.)

Here is what to do if your motors are out of tune.

Locate the code below just above the Main Do Loop.

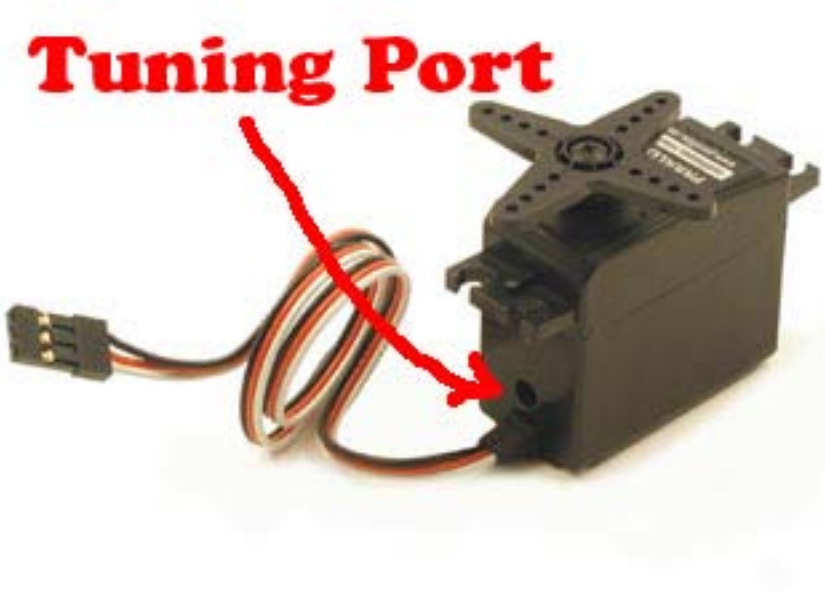
```
ServoLT = 0.0015
ServoRT = 0.0015

        call delay(2.0)
```

Of course the code continues on after this point but the trick to tuning your motors is done with the code shown above.

The Trick: CHANGE THE DELAY TO (200.0) and then re-download.

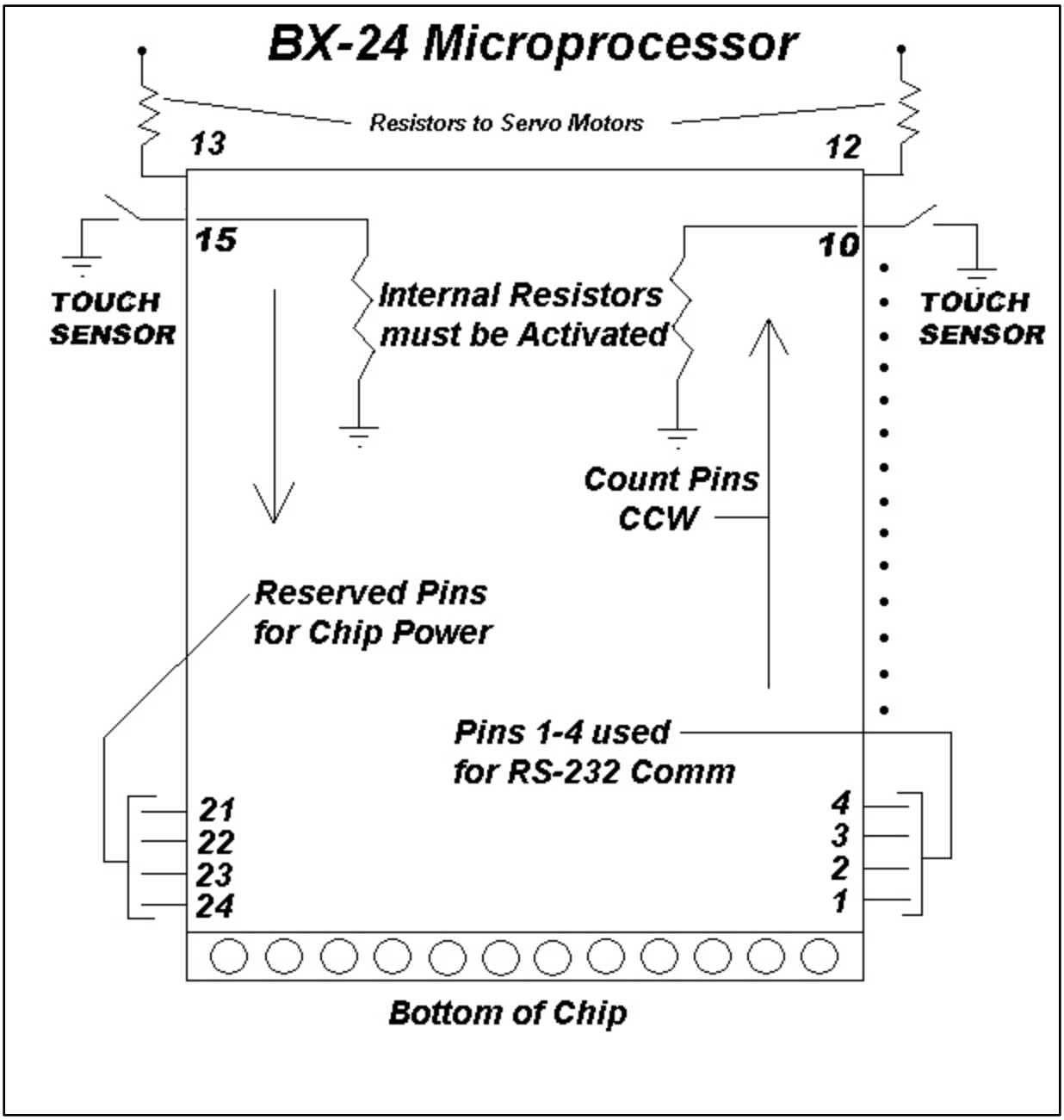
NOW YOU HAVE 200 SECONDS TO TUNE YOUR MOTORS.

After downloading this new code with the long 200 second delay saying don't move, LOCATE the Tuning-Hole located just above the wires on the motors.



Now, with the 200 second delay code running and the motor 'creeping' out of tune, take a small, flat head screw driver and insert it into the Tuning Port.

Turn the potentiometer inside the port with the screwdriver UNTIL the motor stops moving. This takes some practice and patience but as long as you take your time, you will be able to TUNE your motor. Now, go back to your code and change the Delay back to its original value and re-download. Your motors should now be in tune.



Below is the Pulsing Graph for the Operation of the Servo Motors.

1.5 ms Pulse to the motors is STOP
1.5 ms + 0.2 ms = FULL SPEED CW
1.5 ms – 0.2ms = FULL SPEED CCW

Anything in between these 2 points is less than full speed.

Below is the pulsing values sent in the variables ServoLT and ServoRT in the code.