

## Excercise Bike Interface as seen in March 2006 of Nuts & Volts Magazine

Pick up an issue at  
<http://www.nutsvolts.com>



As a software and hardware developer I decided to create an interface to one of my exercise bikes. Before I started I set out with a few requirements

- The interface had to work with any bike.
- The interface had to work with a Desktop PC, LapTop or Pocket PC
- The cost of the interface had to be very cheap and use a minimal amount of components.

After a bit of experimenting I think I have a great start towards a really cool interface that will work with any bike.

### Interface

I thought to myself; what is the best way to interface to a PC?

Well, every PC and Laptop has USB. The downside is that the interface can be expensive and Pocket PC's don't have a USB interface that's accessible.

Printer ports are out because they are increasingly being replaced by USB. I could use IRDA but again the interface circuitry would be too complicated.

What about serial? Most of the newer laptops don't have serial ports anymore. Even the laptop I'm using to write this article does not have one. However you can pick up a USB to Serial converter for under \$25 at Wall Mart.

I decided to try the serial port route.

One of my first questions was how to detect the peddle movement on the exercise bike. Easy! I decided to use a Hall Effect Sensor. These can be purchased at several online stores for a few dollars.

There are several Hall Effect types available. We will be using a 6853 chip that runs off of 5v and has a normally low signal on its output lead (Pin-3). When a magnetic field is detected it floats the lead so you can pull it high with a 1K resistor. Some Hall Effect sensors will latch and need to be released. This type will not work for this project.

By connecting the output lead to the receive pin of a serial interface the PC will read a 0 each time the sensor sees a magnetic field. The neat thing is that the result will be the same at just about any baud rate. Why does this work?

The idle state of a RS232 signal is -12 Volts. This is the low state. When the start bit occurs it goes from the low (idle) stat to high. Sound familiar? This is what happens when our sensor detects a magnetic field. Well kind of. In reality we go from a 0 level to a +5v. This will work on 90 % of the newer devices as they are very forgiving. They see anything less than .5v as a low and above 3v as a high. I tested it on every device I had in my lab, which consist of 12 Desktop PC's, 3 Laptops, 4 Pocket PC's , and various serial converters of one type or another.

### The Basic Circuit

Take a look at Schematic 1. This is how you would connect to a Desktop PC or Laptop. Pocket PC's are considered DTE devices so you will need a male connector and will have to connect the sensor output to pin-2 of the DB9 connector.

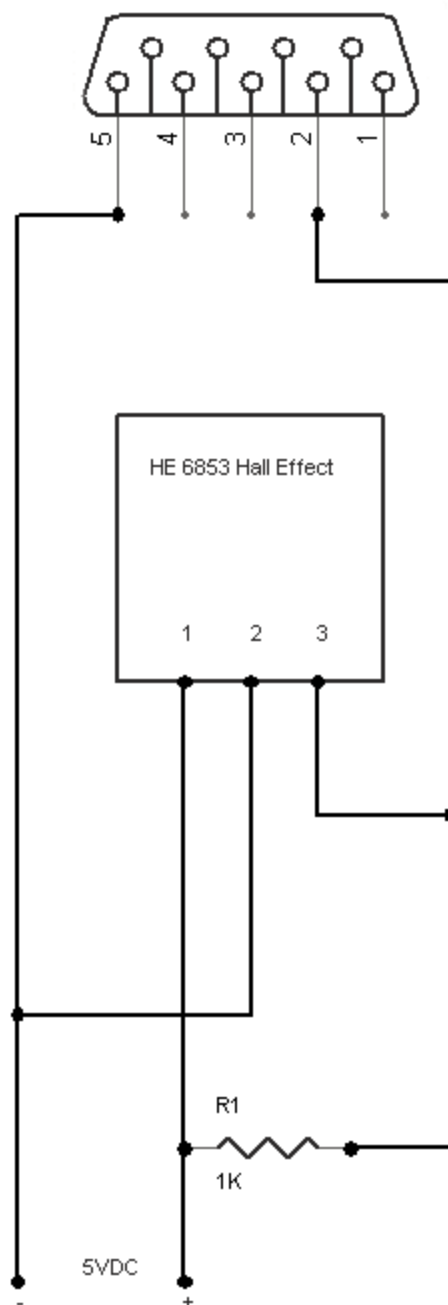
Don't rush out and build this circuit because we are going to do it one better.

I don't want to have to use a battery or AC adapter so I decided to power the interface. By adding a 78L05 and a diode, you can create a self-powered version as shown in Schematic 2.

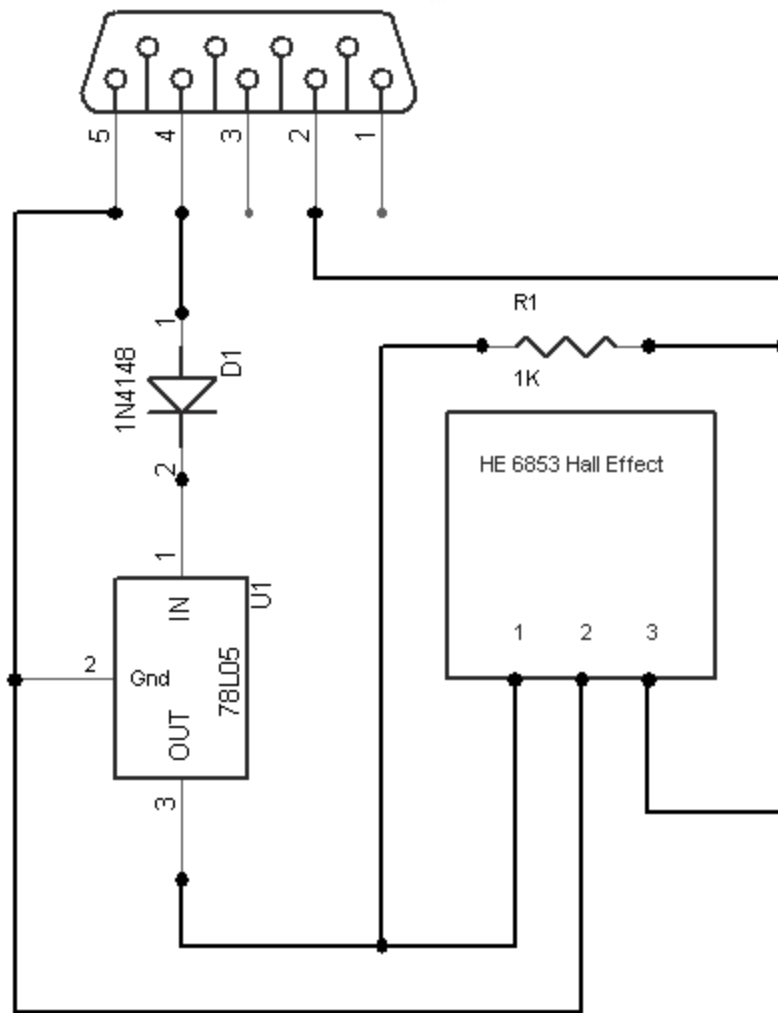
We are using pin-4, DTR to power the circuit. First we run it through a diode so we don't blow anything up if the DTR is set low. When low, the Cathode side of the diode will float, effectively turning off the circuit. Once DTR is raised the positive voltage will drive the 78L05 and get regulated down to 5v. You don't need much since the whole circuit only pulls a few milliamps.

### Sensor Hookup

How you connect the Sensor depends upon your bike. All bikes have some sort of rotating part that the peddles are connected to. The bike I have has an onboard computer so it already has a sensor. Figure 2 shows how I just hot glued my sensor to the top of the existing sensor.



Interface Schematic 1



Interface Schematic 2



Figure 2

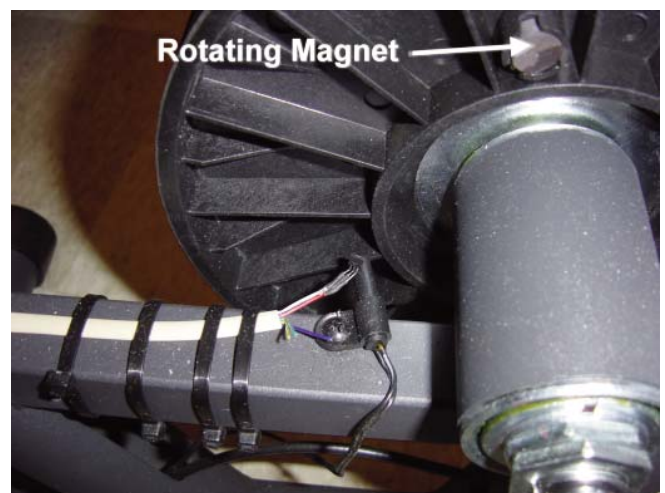


Figure 3

It already has a magnet mounted on the main drive so I simply oriented my sensor so that the front of the sensor faced the magnet as shown in Figure 3.

I strapped the cable down with tie wraps to help secure the sensor. If your bike does not already have a magnet you will need to hot glue one to the drive mechanism somewhere. Keep in mind that if you don't want to open up your bike it is possible to mount the sensor on the exterior of the bike and a magnet on one of the peddle arms. This is what I plan on doing to one of my other bikes.

That's it on the interface. It will cost you around \$5 for the parts depending on what you have in your junk box. As far as assembly goes I soldered the 78L05, Diode and Resistor directly to the connector.

## Software

Now for the fun part. We used a development program called Zeus. There is even a special Free Nuts and Volts version available to Nuts and Volts readers.

That's what we are going to use here.

## Sensor Test

I'm going to start with a simple program just to test if the sensor is working properly.

```
'Exercise Bike Sensor Test
func main()

  gconst chBike 1
  dim Rot as integer
  clearall

  if ComOpen(chBike,baud=9600,port = 4) = 0 then
    MsgBox("Error Opening Com Port")
  end
  endif

  'Power up the connector
  ComDTR chBike,1

Loop:
  if ComBuff(chBike) > 0 then
    Rot = Rot + 1
    Print val(ComInput(chBike)),"Count="+Rot
  endif

  DoEvents
  goto Loop

endfunc
```

Program 1 is fairly straight forward. We **Program 1 Sensor Test**

open up the com port. If the ComOpen returns a 0 we have an error so we display an error message and exit the program.

Once we have an open port we raise DTR with the ComDTR command. This provides power to the interface.

We then run a very tight loop that checks the status of the com buffer with the ComBuff command. If it returns anything greater than 0 we know we have a sensor event.

We increment the Rot variable then print the received value as well as the Rot value.

Note that you have to access the ComInput command to pull the data out of the buffer or the ComBuff command will just keep getting larger and won't ever return a 0.

## Race Program 1

```
'Bike Race Program 1
func Main()

'-----
' Setup Section
'-----

dim Tots(5) as integer 'Total of all rotations
dim Rots(5) as integer 'Current lap rotations
dim Pos(5) as integer 'Current race position
global Laps(5) as integer 'Total completed laps

dim indat as string
ClearAll()

initBike(1) 'Init Com Port and Power up connector. Pass Com Port
FormFont("-",9,1)

'Work Variables
dim ticks as integer
dim x as integer
dim y as integer
dim startticks as integer
dim curticks as integer

'Set Starting point
startticks = getms()/100
FormUpdateAutoOff()

'-----
' Main Loop
'-----

loop:
  DoEvents()
  ticks= (getms() / 100) -startticks
  'print "Ticks ",Ticks

  if ticks > 3 then
```

```

startticks = getms()/100
if Random(1,10) > 4 then
    rots(2) = rots(2) + 1
endif
if Random(1,10) > 4 then
    rots(3) = rots(3) + 1
endif
if Random(1,10) > 4 then
    rots(4) = rots(4) + 1
endif

DrawTrack()
for x = 1 to 4
    Rots(x)=PlotPlayers(x,Rots(x))
next

```

'Calculate Position

```

-----
for x = 1 to 4
    Tots(x) = Laps(x) * 407 + Rots(x)
    Pos(x) = 0
next

for x = 1 to 4
    for y = 1 to 4
        if Tots(x) > Tots(y) then Pos(x) = Pos(X)+1
    next
next

for x = 1 to 4
    Pos(x) = 4 - Pos(x)
next

```

'Display Data

```

-----
FormPrint(10,130,"Mike ")
FormPrint(10,150,"Bot1")
FormPrint(10,170,"Bot2")
FormPrint(10,190,"Bot3")

FormPrint(60,130,Laps(1))
FormPrint(60,150,Laps(2))
FormPrint(60,170,Laps(3))
FormPrint(60,190,Laps(4))

FormPrint(100,130,Pos(1))
FormPrint(100,150,Pos(2))
FormPrint(100,170,Pos(3))
FormPrint(100,190,Pos(4))

formupdate()

```

```

endif

'Check Com Buffer for activity
'-----
if ComBuff(chBike) > 0 then
    indat=ComInput(chBike)
    Rots(1) = Rots(1) +1
    print Rots(1)

endif 'End check tick statement

goto loop
endfunc

'-----
'Draw a Player Pip
'-----
func PlotPlayers(Player,Rots as integer) as integer
    dim Xpos,Ypos
    dim PlayMod

    if player = 1 then
        FormBrush(0,0,255)
        PlayMod = 0
    endif
    if player = 2 then
        FormBrush(255,0,0)
        PlayMod = 6
    endif
    if player = 3 then
        FormBrush(200,0,255)
        PlayMod = 12
    endif
    if player = 4 then
        FormBrush(200,200,155)
        PlayMod = 18
    endif

    if Rots < 92 then
        Xpos = Rots+111
        Ypos = 31 - PlayMod
    endif

    if Rots > 91 and Rots < 122 then
        Xpos = 203 + PlayMod
        Ypos = Rots - 59
    endif

    if Rots > 121 and Rots < 294 then
        Xpos = 324 - Rots
        Ypos = 63 + PlayMod

```

```

endif

if Rots > 293 and Rots < 326 then
  Xpos = 31 - PlayMod

  Ypos = 356 - Rots
endif

if Rots > 325 and Rots < 407 then
  Xpos = Rots - 291
  Ypos = 31 - PlayMod
endif

if Rots >= 407 then
  Rots = 0
  Xpos = Rots+111
  Ypos = 31 - PlayMod
  Laps(Player) = Laps(Player) + 1
endif

FormFillEllipse(Xpos,Ypos,6,6)
FormEllipse(Xpos,Ypos,6,6)

exit(Rots)

endfunc

'-----
' Draw the track onto the screen
'-----

func DrawTrack()

'Track
FormBrush(0,255,0) 'Green
FormFillRectangle(0,0,240,100)
FormRectangle(0,0,239,99)

FormBrush(248,240,112) 'Track Color
FormFillRectangle(10,10,220,80)

FormBrush(0,255,0) 'Green
FormFillRectangle(40,40,160,20)

'Start Finisth Line
FormLine(115,10,115,40)

endfunc

'-----
' Open Com Port and Turn on Interface
'-----

```

```

func initBike(tport as integer)
  gconst chBike 1

  dim stat as integer

  if ComOpen(chBike,baud=9600,port = tport) = 0 then
    stat=MsgBox("Error Opening Com Port")
  end
  endif

  'Power up the connector
  ComDTR(chBike,1)

  Pause(20)

endfunc

```

There are 4 functions used to make up Race Program 1.

#### **initBike(ComPort)**

This function opens the indicated port and sets DTR to power the interface.

#### **DrawTrack()**

This function draws the track. You can change the FormBrush values to change the track colors.

#### **PlotPlayers(Player, Rotations)**

This function plots a player pip at a particular position on the track based on its Rotation value. Note that player 1 is the sensor and players 2-4 are the bot players.

#### **Main()**

This is the entry point of the program. This is where most of the work gets done and due to its complexity it is broken down into its own sub sections.

#### **Main Setup**

Here we set up the program by creating a few arrays.

Tots Total of all rotations. (Sensor Events)

Rots Current rotations of current lap.

Pos Race position. First, Second, Third and Fourth.

Laps Completed laps.

A call to initBike is also made. Notice that I am using com port 4. You will need to pass the com port number your interface is connected to.

#### **Main Loop**

Here we set a variable called ticks. This variable will increment every 10 milliseconds. Once we get a tick value greater than 3 we reset the counter and using a random number generator increment the position of each of the bot pips. For instance take the following snippet of code:

```
if Random(1,10) > 4 then
    rots(2) = rots(2) + 1
endif
```

Here a random number between 1 and 10 is generated. If it is greater than 4 we will increment the bots rotation count. This equates to a 40% chance of this particular pip advancing one position every 40ms.

By changing the value from 4 to say 8 you effectively double the skill level of this bot. This would give the appearance of that bot running faster on average.

After the rotations are calculated we then draw the track and plot all 4 player pips.

#### Main Calculate Position

Here we do a quick bubble sort to calculate each players position in the race.

#### Main Display Lap Data

Here we display the text data indicating the player name, his position and number of laps completed.

#### Main Check Com Buffer

Here we check for activity on the com buffer. If we have activity we increment your (player 1) rotation counter.

## Going Further

I consider this a starting point as there is plenty of enhancements you can make to the program. Here are a few Ideas.

- You could keep track of track of lap times using the *Getms* command and creating a LapTime Array.
- You could calculate other data points like speed, calories and distance. If you have a computer on your bike this will be easy as you have something to help you calibrate your calculations.
- You could keep track of all your laps for so that you could replay them as one of the bot players to race against your own times.
- You could dynamically change the skill levels of the bots thus simulating the fact that players will start to slow down as they get tired.
- You could add sound effects or voice commands to your program using the *PlaySound* command.
- You could upgrade to ZeusPro and add bitmaps and double buffering for smoother graphics.
- Add missiles so you can blow up any of the bot pips that get in your way.

The ideas are limitless.

## Final Thoughts

If you plan on building a Pocket PC version of this software I recommend building and testing it first with a PC or Laptop. You can always use a Null adapter to connect to the Pocket PC if you build the circuit to interface to the PC. You will need to purchase Zeus Pro or Pocket Zeus to compile a pocket PC application.

Feel free to experiment with the code to create the ultimate motivator. Contact me in care of the KronosRobotics

web site and I will post your code up on the website. You can also post your code up on the forums listed at the end of this article.

I successfully used a Bluetooth to RS232 adapter to connect my Pocket PC with out a cable. This type of converter is a valuable one to add to your workbench. I keep one in my Laptop bag as well.

## Parts Source

None of the parts are critical; for instance any diode will work. You will need a DB9 female if you are connecting to a PC or Laptop and a DB9 male if connecting to a Pocket PC. Other Hall Effect sensors will work. Just make sure you use the non-latching type.

78L05 Regulator	Kronos Robotics #16207
6853 Sensor	Kronos Robotics #16211
1K Resistor	Kronos Robotics #16178
1N4148 Diode	Kronos Robotics #16135
DB9 Male	Kronos Robotics #16254

ZeusNV	Nuts and Volts website
ZeusPro	KRMicros website under development section

## Web Links

Nuts and Volts website: [www.nutsvolts.com](http://www.nutsvolts.com)  
KronosRobotics website: [www.kronosrobotics.com](http://www.kronosrobotics.com)  
KRMicros website: [www.krmicros.com](http://www.krmicros.com)  
KronosRobotics forums: [www.kronosrobotics.com/forums](http://www.kronosrobotics.com/forums)

**EZServo1** .....<http://kronosrobotics.com/xcart/customer/product.php?productid=16273>  
**Athena WorkBoard PCB** .....<http://kronosrobotics.com/xcart/customer/product.php?productid=16460>  
**Athena WorkBoard Deluxe** ....<http://kronosrobotics.com/xcart/customer/product.php?productid=16457>  
**Athena WorkBoard Basic** ....<http://kronosrobotics.com/xcart/customer/product.php?productid=16473>  
**ZeusPro** .....<http://kronosrobotics.com/xcart/customer/product.php?productid=16479>