iCSC 2014 International Cycling Safety Conference
November 18-19, 2014
Gothenburg, Sweden

Braking tests on a variety of urban bicycles

Christopher Morris
Pedal Depot Cooperative
1830 Ontario Street
Vancouver, BC, Canada V5T 2W6
e-mail: cminkey@telus.net

Introduction:
Measurements were made of emergency braking distance on six very different urban bicycles, incorporating a variety of braking systems, wheel sizes and friction pairs. The tests were performed on a defined section of bike route in Vancouver (shared with cars). Conditions were identical (25C, dry and sunny) over the two August days that the instrumented bicycles were tested in dry weather. For the wet tests in October, conditions were 13C and medium rain. The road surface was asphalt with some crazing on a specific 7% downgrade. The downgrade was chosen to draw out differences in stopping distance between the braking systems used, it also made a stable entry speed to the EB easier to achieve. The brakes were mechanical disc, side-pull rim brake, centre-pull rim brake, V brake and coaster brake. The rims were either chromium-plated steel, aluminium or plastic, while wheel sizes varied from 20" to 28" diameter.

Full emergency stops were performed every time, to the limit of rear tire/road adhesion in some cases. Six dry tests were performed for each bike – front brake alone, rear brake alone and both brakes, with this sequence then repeated. With the later wet weather tests added, a total of 72 recorded emergency stops were obtained for the 6 bikes used. All riding/braking was performed by the author.

To encourage other experimenters in bicycle braking safety, some description is also given of the custom-designed circuit and written microcontroller/PC programs to obtain the braking data.

Instrumentation:
A special *Skippy* data-logger designed, built and programmed for the purpose (by the author) was used, taking its wheel rotation count from a front wheel-mounted magnet and reed switch on the fork, the same sensor used for consumer bike speedometers. The name *Skippy* comes from a popular brand of peanut butter marketed in North America – the prototype used an empty *Skippy* Jar to hold the circuit board and 3 AA batteries, so as to mount in a standard bike water bottle holder. An audio DIN socket on the *Skippy* Jar lid allowed an interface cable to connect to either the reed switch on-bike or via a special cable harness to the PC's parallel port for data downloading, off-bike.

The Atmel AVR Tiny11 microcontroller[1] used on-bike, whose advanced Harvard RISC architecture was developed in the late 90's in Trondheim, Norway by Nordic VLSI, has a simple but efficient instruction set well suited to direct Assembler programming. About a hundred instructions were used for the final *Skippy Pedal* program, well within the 512 instruction on-chip flash memory limit, illustrating the power of RISC architecture in tiny microcontrollers.

Skippy peanut butter jar in Supermarket

Skippy prototype logger and final version logger

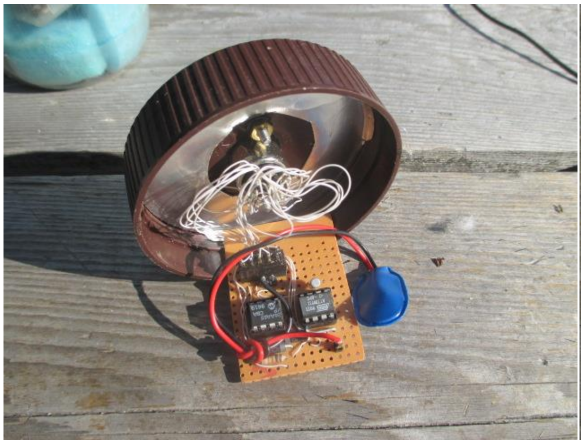
The AVR Tiny11 µC has just 8 pins in a DIP package and a suite of 90 x 16 bit powerful RISC instructions available to choose from. The *Skippy Pedal* program stores the bicycle timing/distance data recorded from the front fork-mounted reed switch to a serial (I^2C 2 wire bus) 8 pin EEPROM memory, using a simple 32Khz watch crystal for accurate timing. This 'clock' also feeds a crystal-locked accurate 128Hz signal to a counter register in the Tiny11 - that is stopped and summed whenever the spoke magnet passes the reed switch, hence measuring the speed of the bike. All compensation for wheel size and conversion to engineering units is done in the PC by the *Skippy Bike* GUI program.

On the *Skippy* Jar prototype, a Microchip 24AA65 EEPROM was used to store data points (8Kb of storage, 1 speed record per byte). The second *Skippy* version, used in the August 2014 brake tests, uses a smaller plastic case with a built-in parallel port plug for the PC interface. It uses the larger 64Kb Atmel 24C512 serial EEPROM, and runs off a 9v PP3 battery via a low power 5v series regulator. The simple circuit is constructed on perf-board, and is completed by a resistor, 2 tantalum capacitors, 2 Schottky diodes and the LP2950 low power regulator (the circuit diagram is in the Appendix). Unlike the prototype, which used the DIN plug insertion to power up, it has a TPST toggle switch on the case exterior.

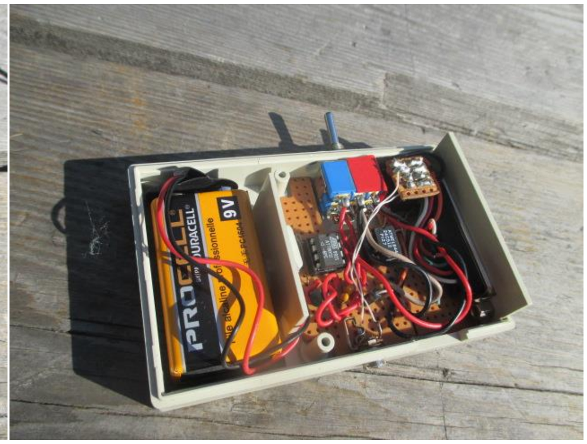
Instead of multi-second wasteful bulk-erasing of the EEPROM every time *Skippy* is used for a ride, the later version of *Pedal* writes a '1' to the EEPROM after every 16 words of bike data written. This '1' represents an impossible Mach 1 bike speed - a marker (when downloading the data from EEPROM to PC) that the end-of-file for the ride has indeed been reached.

The Tiny11 Assembler listing for *Skippy Pedal* is provided in the Appendix to this paper, as is the custom written Liberty Basic GUI program *Bike* for the PC - for those researchers who may wish to experiment further in this area. *Bike* will find the PC port that *Skippy* is plugged into, then downloads the most recent ride as a .csv Excel file, using the *Autograf* button. The Excel file is automatically named, using the time and date that the PC read Skippy's EEPROM.

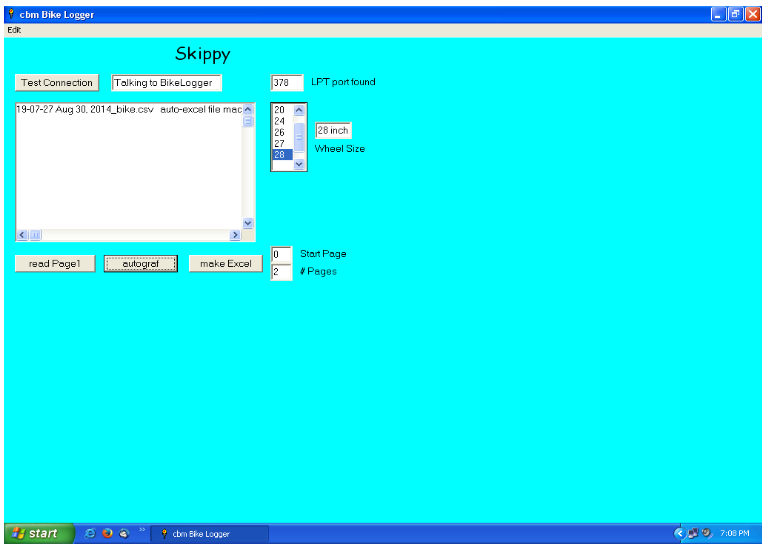
Finally the Chart Wizard feature in Excel is used to produce a graph of speed against wheel rotations (ie distance) from the .csv file. The *Bike* program allows wheel diameter selection before downloading from *Skippy*, then calculates the real speed in MPH for the Excel file.



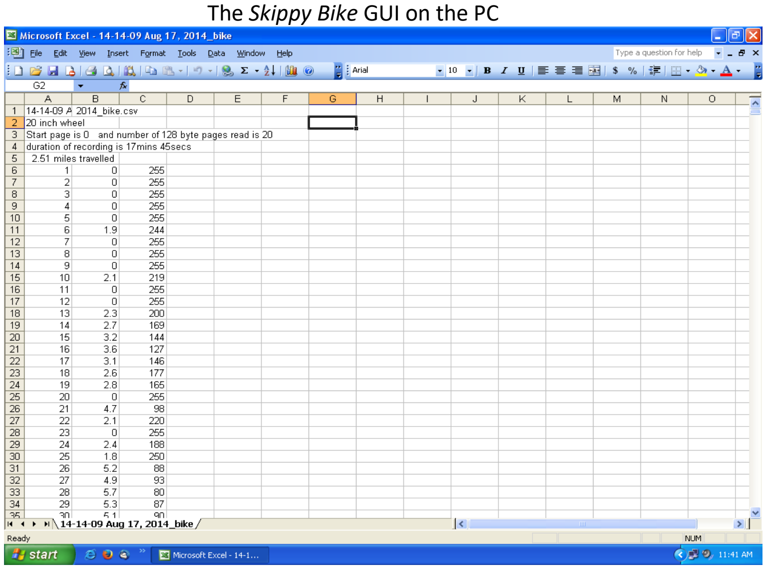
Skippy Jar prototype circuit board



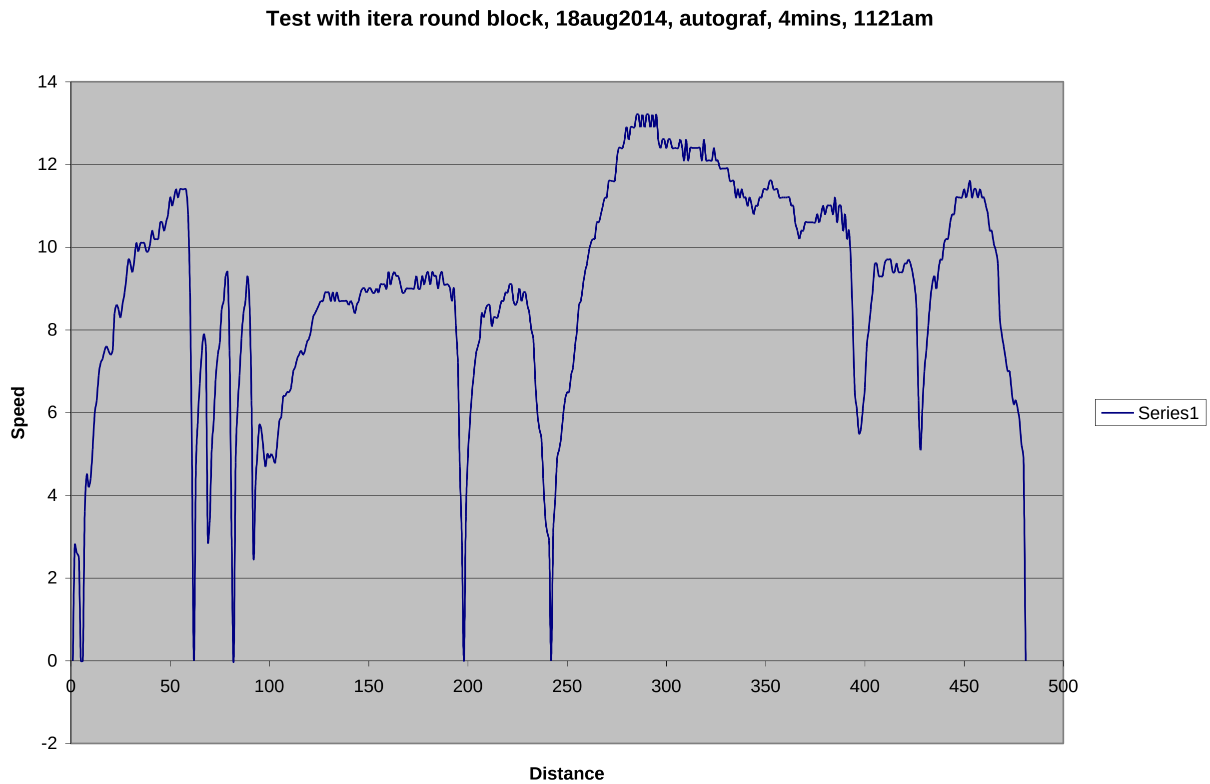
Skippy final version – note on/off switch added



The *Skippy Bike* GUI on the PC



Resulting .csv Excel file – here from Bike 1, Victoria, 1414hrs



Excel graph above - proving out *Skippy* instrumentation on-bike

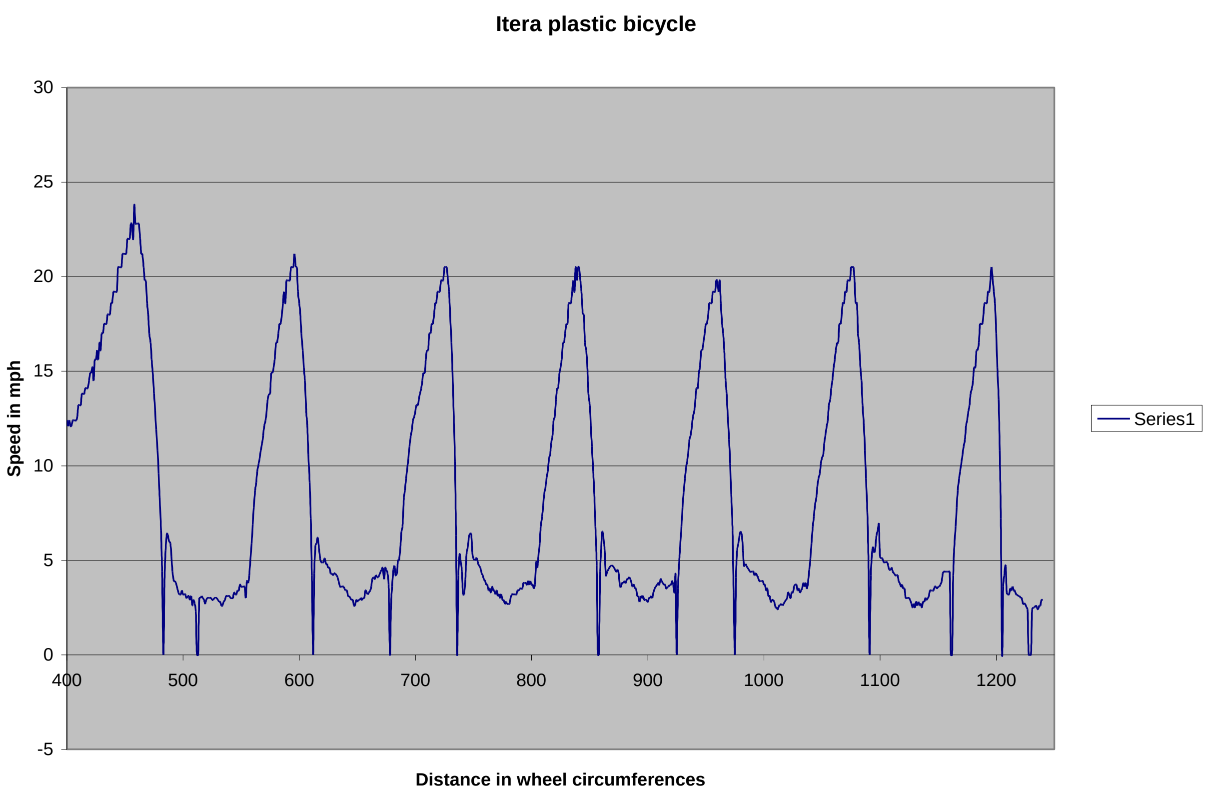
In addition, the test bikes were each equipped in turn with a compact *Mio MiVue*[3] GPS wide-angle camera/video recorder, mounted on the handlebars and looking forward. The camera is made in Taiwan largely for the domestic car/scooter insurance market there (accident claim adjudication). Hence it is available at a very reasonable direct import cost (~$70 including shipping), due to mass-production. Included is GPS map-reading software with speed and time added, in addition to the video record, all of which can be simultaneously displayed on a PC screen. It comes with all necessary mounts for bicycle handlebars as well as car sun visors, plus a waterproof cover for rainy weather. Video is written to a built-in SD micro-card. The only drawback, common to many bicycle cameras, is the short battery life of less than an hour from the built-in twin NiMh rechargeable AAA cells.

I am indebted to Professor Sadao Horino[4] of Kanagawa University in Yokohama, Japan (presenter at ICSC 2013 in Helmond) for his suggestion and demonstration to me of this largely unknown (in the West) family of tiny cameras for naturalistic cycling studies at a reasonable cost.

Instrumentation example:

Most bikes were straightforward to instrument with camera and reed switch/magnet pair. The Itera proved a challenge, as it uses 8 large compression spokes on its plastic wheels, instead of steel tension spokes. Its plastic wide-section handlebar also gave some difficulty mounting the Mio GPS camera. The good news was that it was invisible to police speed radar during the EB tests.

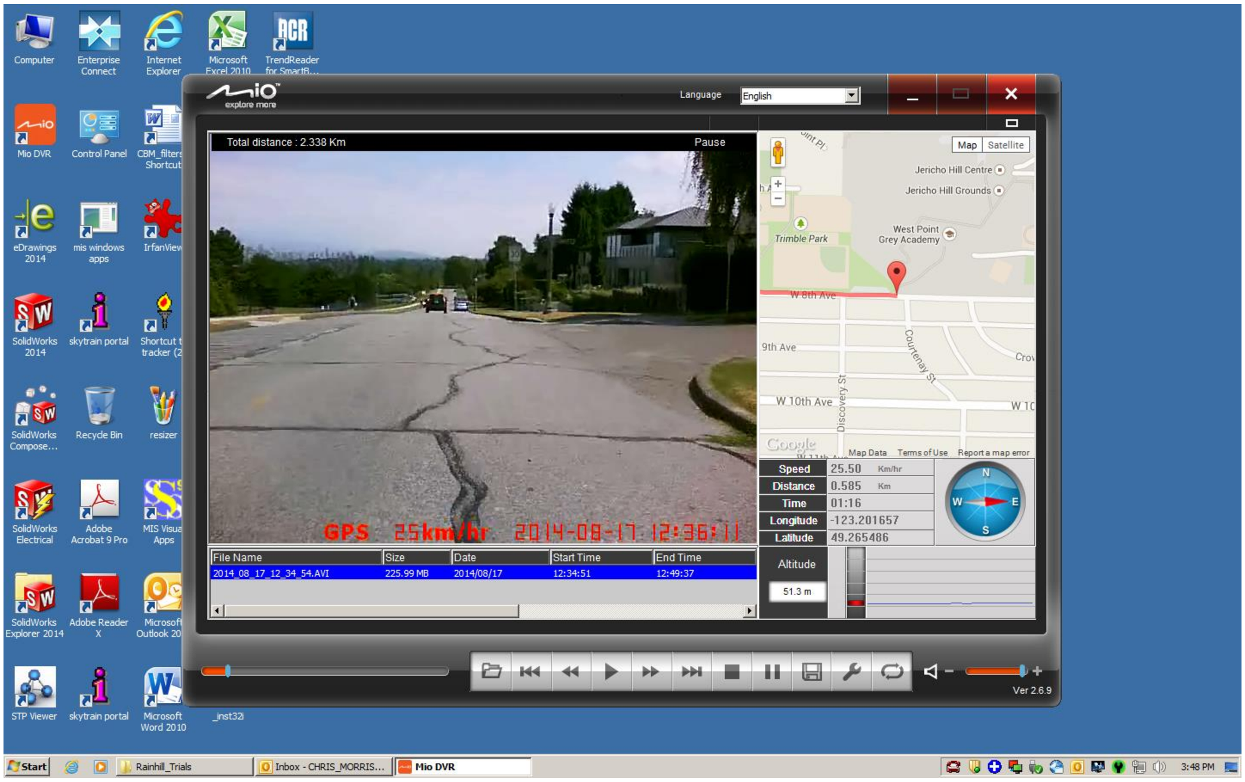
The Itera plastic bike proved a challenge to mount magnet.. ..as there are no tension spokes – see above for solution.



Typical EB stops on test section

<u>Mio MiVue GPS camera</u>

Here is an example of the Mio GUI readout of the bike mounted camera with .avi video and GPS Google map readout recorded for 71* of the 72 tests (*the very last test was in pitch darkness in driving rain..). The example below is the first dry Front Brake EB with the Victoria folder (bike 1). It also illustrates the road surface and downgrade of the test section. Audio is recorded as well, making this a very useful device for cataloging multiple bike test times, comments and skid sounds.



Mio MiVue Video/Audio/GPS mapping playback on the PC from a test run


The *Mio Mivue* GPS camera mounted on the Cadillac handlebars

<u>Test Protocol:</u>

The normal sequence of brake tests was **front EB/rear EB/full EB/front EB/rear EB/full EB** for each bike in turn. Bikes 1-3 (dry) were on 17 August afternoon, bikes 4-6 on 18 August afternoon. Weather was an identical 25C, dry and sunny. The test rider was the same on each run (the author). Hand and pedal brakes were applied as fully as possible each time (Emergency Braking or EB). The rider weight was 182 lbs. West 8th Avenue in Vancouver eastbound between Courtenay and Crown Streets was the EB section, in the area of West Point Grey. A similar protocol was used for the wet tests on 11 October (bike 1) and 17 October (bikes 2-6): weather was raining and 13C for both those days.

The Bikes:

1968 Victoria 20" detachable from West Germany

2010 Cadillac 28" NuVinci from China

1973 CCM Elan 26" from Canada

1982 Itera 27" plastic bicycle from Sweden

2013 Moulton TSR2 (4) 20" detachable from England

2010 Asama Euro7 28" from Taiwan

1 **Victoria** monocoque 20” wheel folder, Nurnberg, West Germany 1968.
Weinmann alloy sidepull brakes on chromed steel rim on front, Sachs Duomatic 2 speed hub coaster brake on rear. Tires 20” x 1.75”. Weight as tested 34lbs.
Tested dry 17 August, *1414* Excel file. Tested wet 11 October, *0946* Excel file.

2 **Cadillac** aluminium frame and rims, urban men’s bike, China 2010. Promax mechanical discs front & rear, NuVinci continuously variable hub drive. Tires 28 x **1.125”**. Weight as tested **40 lbs**. Tested dry 17 August, *1518* Excel file, 6 EB’s. tested wet 17 October, *1819* Excel file.

3 **CCM Elan** mixte steel frame, Weston, Ontario, Canada 1973. Centre pull alloy CLB rim brakes front, no-name steel sidepull rim brakes rear, all on chromed steel rims. Weight as tested 40lbs. Tires 26”x 1.375”. Tested dry 17 August, *1609* Excel file, 6 EB’s - last 2 EB’s reversed order (full then rear). Tested wet 17 October, *1257* Excel file.

4 **Itera** all-plastic bike, Vilhelmina, Sweden, 1982. Weinmann alloy sidepull rim brakes on plastic rims. Weight as tested 40 lbs. Tires 27” x 1.25”. Tested dry 18 August, *1227* Excel file. Extra front EB added after second front EB, with a tightened brake cable (made no difference). Tested wet 17 October, *1029* Excel file. The all-plastic injection-moulded Itera was developed in 1980 by ex-Volvo engineers in Goteborg, then put into mass-production at Vilhelmina in Lappland, just below the Arctic Circle. By the time production ended in 1985, 30,000 had been produced.

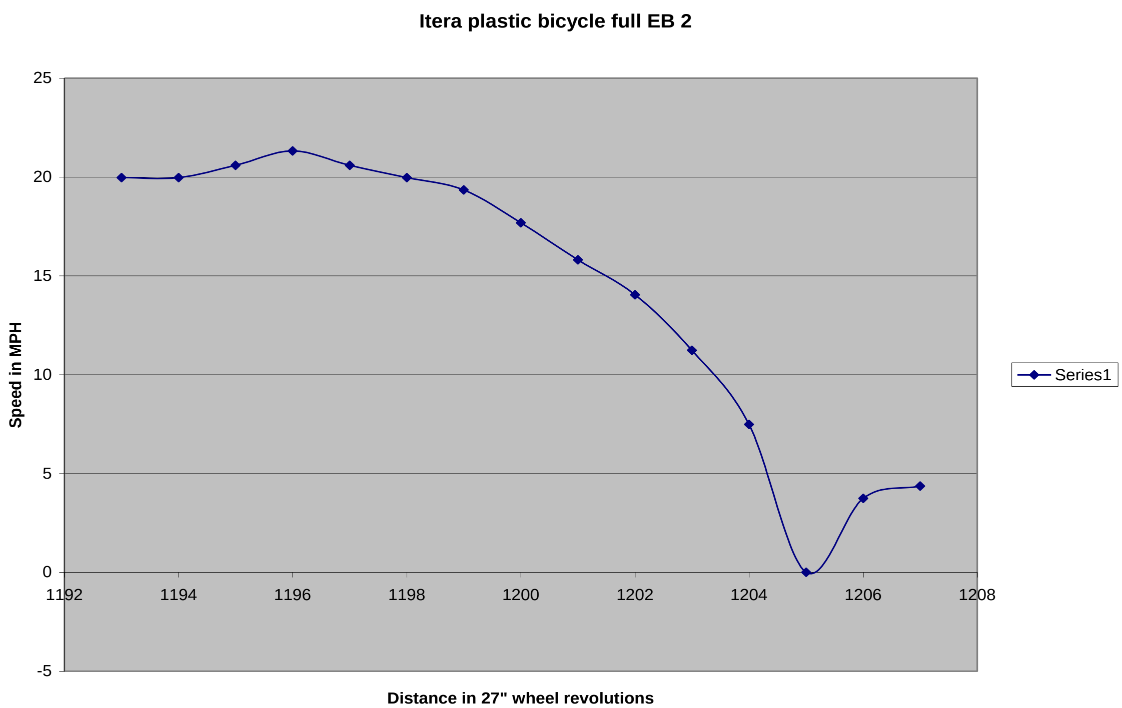
5 **Moulton TSR2 (4)** 20” folder, ChroMoly steel space frame, Stratford-on-Avon, England 2013. Avid *single digit* V brakes on aluminium front rim, Shimano **4 speed** hub coaster brake on rear. Weight as tested 32 lbs. Tires 20” x 1.375”. Tested dry 18 August, *1319* Excel file. Tested wet 17 October, *2207* Excel file.

6 **Asama Euro7** ‘Dutch-style’ lady’s bike, steel frame, Taiwan 2010. Alhamo V brake on the front aluminium rim, Shimano 7 speed hub coaster brake on the rear. Weight as tested 37 lbs. Tires 28” x 1.5”. Tested dry 18 August, *1411* Excel file. First front EB repeated due to ‘lamp post’ uncertainty for onset of EB. Tested wet 17 October, *1629* Excel file.

Use this hyperlink to view all Excel files and .avi videos: http://1drv.ms/1taIw2a

Worked Example:
Using the kinematic equation $v^2 = u^2 + 2as$ where v is initial speed, u the final speed (0 for a stop), a the acceleration and s the distance, the average bicycle deceleration from 20mph (32km/hr) to rest can be computed, knowing the stopping distance s. The effect of the 7% constant downgrade can be added after. Here is a worked example from the *Skippy* log and Excel spreadsheet:

#4 - **Itera** plastic bike, second full EB (both front and rear brakes fully applied).
Wheel diameter 27”, so circumference (one revolution) is 27” x pi = 84.82”.
From the Excel graph below, 7 full wheel revolutions were counted in the braking curve from 20mph to rest.
s = 7 x 84.82”/12 = 49.5’ stopping distance in feet.
v = 20mph = 20 x 88/60 = 29.33 ft/sec., u = 0 ft/sec.
$a = (v^2 - u^2)/(2 \times s)$ = 29.3^2/ 2 x 49.5 = 8.69 ft/sec^2
g (acceleration due to gravity) = 32.17 ft/sec^{2,} hence a = 8.69/32.17 = ***0.27g deceleration***



Overall Results for all six bikes:

Table 1: Deceleration measured from 20mph, ignoring downgrade. **Dry**, 25C, sunny, asphalt surface. **August 17/18 2014.** No wind.

	Front EB 1	Rear EB 1	Full EB 1	Front EB 2	Rear EB 2	Full EB 2
1.Victoria 20”	0.20g	0.28g	0.42g	0.21g	0.25g	0.41g
2. Cadillac 28”	0.36g	0.24g	0.36g	0.46g	0.24g	0.36g
3. CCM Elan 26”	0.28g	0.25g	0.39g	0.28g	0.25g	0.33g
4. Itera 27”	0.13g	0.15g	0.27g	0.15g	0.15g	0.27g
5. Moulton 20”	0.39g	0.36g	0.46g	0.42g	0.22g	0.42g
6. Asama 28”	0.48g	0.24g	0.48g	0.40g	0.21g	0.51g

Conclusions – dry braking

All bicycles had the Bowden cable and hand levers adjusted and in good condition, lubricated and moving freely, where applicable.

The best front brakes were the V brakes on aluminium wheel rims (Moulton and Asama). The mechanical disc on the Cadillac came next, followed by the centre pull rim brake on the CCM Elan. The Victoria sidepull rim on a small chromed steel wheel was poor, the sidepull rim brake on the Itera plastic wheel rim was atrocious. This latter phenomenon is likely due either to the friction pair (standard brake block rubber on a plastic rim) or the plastic rim yielding elastically as the brake shoes squeeze it. The same effect occurred on the identical Itera rear brake.

Coaster brakes were by far the best brakes on the rear. Other than the Itera, all other rear brakes grouped in the quarter ***g*** range, regardless of whether they were sidepull or mechanical disc. Light skids occurred at the rear in several cases, particularly on the Cadillac which had excessively skinny tires (for styling presumably) despite its heavy weight (partly due to the 8 lb NuVinci rear transmission hub). It is clear that the propensity to rear skid with undersized tire widths (1.125" to support a 40lb bike!) on dry pavement dominated the Cadillac results, from an otherwise good disc brake. During a hard EB stop with dual brakes, the rear wheel is naturally unloaded so that often tire/asphalt adhesion at a light normal (ie vertical) load is the skid limiting factor, rather than the brake type or capability.

Full EBs were best on the three coaster brake bikes (Victoria, Moulton and Asama). The Cadillac (mechanical discs) and the CCM Elan (rim brakes) were similar to each other; which is interesting - given that this was 1973 rim brakes up against modern discs. The Itera was in a class of its own..

Two results of the thirty six EB tests performed were anomalous and deserve a retest (Cadillac Front EB2 and Moulton Front EB2).

Including the effect of the 7% downgrade (0.07g) gave full EB's on the level as good as 0.49-0.58g from the top 3 bikes. This well exceeds the bicycle braking regulations of the EU, USA and Japan.

Overall, the best braking combination in these dry tests was front V Brake/alloy rim and rear coaster. The V brake was invented by Shimano[5] in 1996, the coaster brake was invented well over a century ago[6] and is clearly still going strong.

Table 2: Deceleration measured from 20mph, ignoring downgrade. **Wet, raining,** 13C, overcast, asphalt surface. **October 11 and 17, 2014**. No wind.

	Front EB 1	Rear EB 1	Full EB 1	Front EB 2	Rear EB 2	Full EB 2
1.Victoria 20"	0.16g	0.19g	0.31g	0.15g	0.18g	0.28g
2. Cadillac 28"	0.23g	0.16g	0.27g	0.25g	0.17g	0.24g
3. CCM Elan 26"	0.07g	0.02g	0.20g	0.13g	0.18g	0.28g
4. Itera 27"	0.17g	0.13g	0.27g	0.19g	0.15g	0.25g
5. Moulton 20"	0.24g	0.19g	0.32g	0.24g	0.16g	0.30g
6. Asama 28"	0.36g	0.22g	0.31g	0.33g	0.20g	0.39g

Conclusions - wet braking

The wet results turned some dry conclusions on their head. In general there was an unsurprising drop in braking of about 0.1g due to the wet road/tire adhesion and wet friction pairs on-bike. However, there was **complete fade in the wet** for the CCM Elan on the first two EB's – dragging one's heels on the ground would have been more effective. The exact same result was obtained within an hour of this test, again on the first two EB's until the rim had temporarily dried with heavy use. This appears bizarre, but has been reported elsewhere in the literature for steel rim brakes, for example by Professor D. Gordon Wilson[7] of MIT.

What is even more bizarre is that this combination (sidepull brakes on chromium-plated steel rims) was used on millions of bikes manufactured in the last half of the previous century: effective in the dry, dubious in the wet (always remembering that that first EB in the rain may be the important one).

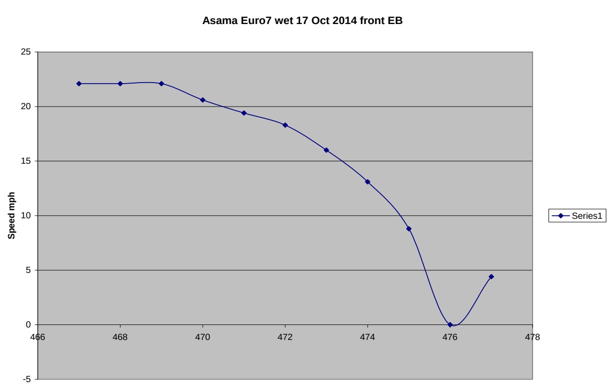
The Victoria did not suffer the same problem with its steel rims on test, it is possible that the sipes cut into the rim by the German manufacturer helped. The general 'solution' in the closing decades of the last century was to switch to aluminum rims, which do not fade to the same extent in the wet. However the aluminum wears sacrificially with rim brakes, so the more responsible manufacturers now provide a wear groove (to prevent the rim exploding in use as it gets thinner).

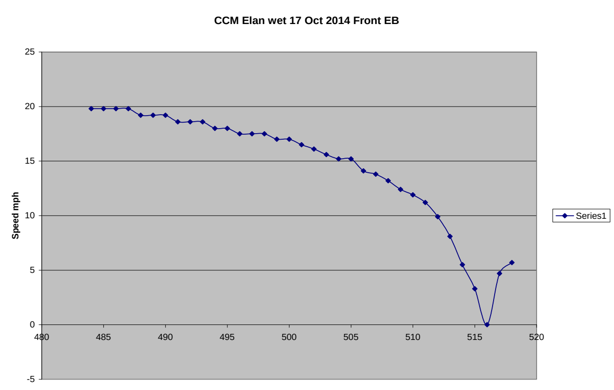
Sipes on rim of 20" Victoria folder, for wet braking

Condemning groove on aluminium rim of Asama Euro7

The Swedish Itera redeemed itself during the wet trials. Apparently the plastic rim, while low on overall braking force, does not fade in the wet with conventional rim brakes, a happy result. Alone among the rest of the bikes tested, the Itera had identical results dry to wet.

Often on the various bike tests, skidding at the rear wheel limited overall braking, a consequence of wet adhesion and tire pattern. This was especially noticeable with the heavy Cadillac with its tiny tire widths, despite having disc brakes.





The Good (Asama first Front EB wet, V Brake)…(each dot is a wheel rev)…and **the Ugly** (CCM Elan first Front EB wet, Sidepull)

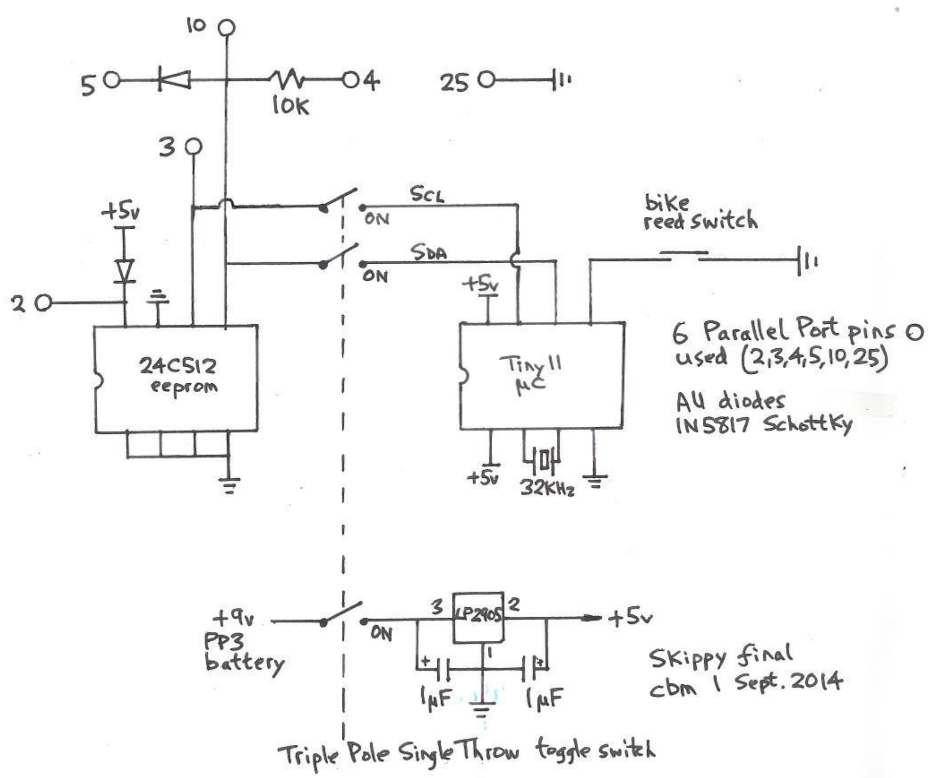
As in the dry, the most effective braking combination was V Brake on aluminum rims at front, coaster brake at rear. Again, the best braking was on the Asama Euro7, followed by the small-wheel Moulton. It was interesting to see the 46 year-old Victoria folder come in third. Progress is not always linear..

References:

1 http://www.atmel.com/images/1006s.pdf
2 http://www.libertybasic.com/
3 http://www.mio.com/sea/products-GPS-Drive-Recorder-MiVue128-overview.htm
4 http://www.icsc2013.com/papers/horino2013ergonomics-abstract.pdf
5 http://en.wikipedia.org/wiki/Bicycle_brake#V-brakes
6 *A New Departure or Two*, Christopher Morris, ICHC 2013, Lisbon, page 2.
http://ichc2013.cies.iscte-iul.pt/images/ICHC2013_BookAbstracts2.pdf
Paper published in *Cycle History 24*, pp 67-81, by Cycling History (Publishing) Ltd. 2014, ISBN 978-0-9573427-3-6
7 *Human Power* magazine vol. 53, Spring 2002, D.G.Wilson pages 10-18
http://www.ihpva.org/HParchive/PDF/hp53-2002.pdf

Appendix

Skippy microcontroller final schematic, bike mounted, using reed switch sensor/spoke magnet from front wheel as input.



Pedal assembler program for the Atmel AVR Tiny11 uC in *Skippy*


```
;
;**************************************************************************
;
;   Pedal1.asm - cbmorris - for ICSC III - Goteborg, Sweden, November 2014
;
;**************************************************************************
;
;v1.1    5A/A5 eeprom write checker-board memory test, 32KHz xtal, Tiny 11
;v1.2    reed switch read, eeprom page write of count
;v1.2    working with magnet/reed !!  29 bytes written per page.
;v1.3    multipage write (16 bytes x 3)
;v1.4    unlimited page write, overflow detection for when bike stopped (255)
;v1.5    full 64k bytes vs. 256 bytes before (pagebig)
;v1.6    bulk erase added.
;v1.7    modified to erase first 32 pages only,takes 23 seconds.
;        - working on 1951 Humber bike, trip to 711 Store and back !!
;v1.8    moved counter clear to before page write, page erase as
;        was taking enough time to give false reading in pagelet byte 0
;        -also added SLEEP shutdown after 32 pages written.
;v1.8A   modified for Skippy 24AA65 datalogger in a jar (8K bytes/8 byte page write)
;v1.9    64 pages vs. 32 (allows 1 hour recording ~11 miles)
;v1.10   added Marker to write # pages used to top byte
;v1.11   got Marker working (added 10mS delay at top)
;v1.12   removed redundant bulk erase code, saving eeprom life
;v1.13   changed to Marker every 16 bytes (pagelet write), Mach 1 speed for eof

;v1.14   fixed page boundary problem.
;        FIN 29 days evolution in all.
;
;Atmel AVR Tiny11 flash microcontroller & 24C512 I2C eeprom with 64K bytes storage
;records bike speed from speedo-style reed switch and writes to eeprom..
;written in AVR assembler code for WAVRASM compile to hex.
;
;24C512 can use up to 128 byte write pages, Tiny11 has 16 registers spare for
;bike speed data - therefore 16 byte pagelet writes used by 'Pedal' pgm
;
;Program uses 96 words of AVR Tiny11 out of 512 word program space available (v1.12)
;
;Flash Fuse on Tiny11 CKSEL=110 at programming time set for 4 sec.
;(startup time with slow 32KHz watch xtals.)
;
;
;
.include "Tn11def.inc"
;
.def Temp       =R29
.def Counter    =R31
.def Chareg     =R29
.def pagestart  =R28
.def pagebig    =R27            ;pagebig is # 256 byte double pages
;
.equ Scl        =2
.equ Sda        =1
.equ Reed       =0
.equ eeprom     =0xA0           ;I2C 24C256 address
.equ page       =16             ;small Tiny11 RAM=16 byte data pagelet
.equ marker     =1
;
Initialise:
        ldi Temp,0b00000110     ;SCL & SDA outputs, Reed switch input port.
        out DDRB,Temp
        ldi temp,0b00000011     ;SCL lo, SDA hi, Reed pullup,i.e. cldh
        out PORTB,Temp
```

```
        clr pagestart            ;start at page 0
        clr pagebig
        ldi Temp,0b00000100
        out TCCR0,Temp
        clr r30
;
Magnet_arrives:
        sbic    PINB,reed        ;Reed closed?
        rjmp    Magnet_arrives   ;No
        in      Temp,TIFR
        sbrc    Temp,TOV0        ;timer/counter overflow?
        rjmp    Over_flow        ;Yes
        in      Temp,TCNT0       ;Reed count
        rjmp    Store_reading
Over_flow:
        ldi Temp,0b00000010      ;reset overflow bit
        out     TIFR,Temp
        ldi     Temp,$ff         ;255 signifies overflow ~stationary wheel.
Store_reading:
        st      z,Temp           ;store to reg,file (r0-r15 used for pagelet)
        clr     Temp
        out     TCNT0,temp       ;clear counter before any page write
        inc     r30              ;incr z
        cpi     r30,page         ;pagelet full?
        brne    Magnet_passes    ;No
;
Page_write:
        clr     r30
        rcall   _START
        ldi     Chareg,eeprom
        rcall   _WRITE
        mov     Chareg,pagebig   ;msb page address for 24C512
        rcall   _WRITE
        mov     Chareg,pagestart;lsb page address
        rcall   _WRITE
LoopR:
        ld      Chareg,z         ;indirect register addressing
        rcall   _WRITE
        inc     r30
        cpi     r30,page         ;pagelet written?
        brne    LoopR            ;No
        rcall   _STOP            ;10mS eeprom write time covered by 6.2Hz max reed
        clr     r30              ;at 30mph, =  161mS update time.
        subi    pagestart,-page ;increment to next page address
        cpi     pagestart,0      ;256 bytes written?
        brne    Mark_Sub         ;No
        inc     pagebig          ;Yes
        cpi     pagebig,254      ;next-to-last 'bigpage' of memory (256 byte page)
        breq    Stop_Recording
;
Mark_Sub:
        rcall   _MARKER
;
Magnet_passes:
        sbis    PINB,Reed        ;Reed open? (normally open Reed)
        rjmp    Magnet_passes    ;No
        rjmp    Magnet_arrives   ;Yes
;
Stop_recording:
        ldi     Temp,0b00110000 ;set SE & SM bits for power-down -
                                 ;unlikely-memory space now huge for Ride file
        out     MCUCR,Temp
        sleep                    ;only 2uA drawn in theory
```

```
;
_START:                         ;I2C start
        sbi     PORTB,Scl
        cbi     PORTB,Sda
        cbi     PORTB,Scl
        sbi     PORTB,Sda       ;restore default
ret
;
_STOP:                          ;I2C stop
        cbi     PORTB,Sda
        sbi     PORTB,Scl
        sbi     PORTB,Sda
        cbi     PORTB,Scl
ret
;
_WRITE:                         ;I2C write
        ldi     Counter,0x08
_LOOPA:
        sbi     PORTB,Sda
        sbrs    Chareg,7
        cbi     PORTB,Sda
        rcall   _STROBE
        lsl     Chareg
        dec     Counter
        brne    _LOOPA
        sbi     PORTB,Sda
        cbi     DDRB,Sda
        rcall   _STROBE
        sbi     DDRB,Sda
ret
;
_STROBE:                        ;I2C strobe = ACK
        sbi     PORTB,Scl
        cbi     PORTB,Scl
ret
;
_10msDELAY:                     ;EEPROM write time 10mS
        clr     Temp            ;30.5uS instrn.cycle - 32KHz xtal
LoopD:
        inc     Temp
        cpi     Temp,100        ;82 x 4 cycle loop = 10uS, use 100 x 4
        brne    LoopD
ret
;
                                ;MARKER is the subroutine that writes a 'Mach 1'
                                ;Bike speed after every 16 byte pagelet write,
                                ;eventually indicating the end of Ride file
                                ;when Skippy Pedal is switched off
_MARKER:
        rcall   _10msDELAY      ;needed because just did a page write to slow EEPROM
        rcall   _START
        ldi     Chareg,eeprom
        rcall   _WRITE
        mov     Chareg,pagebig  ;current high order address
        rcall   _WRITE
        mov     Chareg,pagestart;current low order address
        rcall   _WRITE
        ldi     Chareg,marker
        rcall   _WRITE
        rcall   _STOP
ret
;
```

Liberty Basic program *Bike* that acts as the PC GUI to *Skippy*, downloading from the on-bike I2C serial eeprom and converting data to a .csv Excel file

```
'
'     bike_lib.bas program
'24C512 eeprom reader and Excel data file builder August 2014
'
'This Liberty Basic  program uses the Lpt port on a PC to read the Skippy Pedal
'I2C serial EEPROM and write the Ride data to an Excel file
'NB: busy polling used, hence PC independent with no delays needed.
'     later modified for 24C512 Atmel 64K EEPROM (double word address)
'-added recording time, 2 dec.places for mph in excel file.
'176 lines of code.
'-Liberty 3.0 (32 bit).  Added Lpt finder.
'-redesigned the Form using Freeform 3.
'-Working!! reads EEPROM ok.
'-reliable PC read -, cut Scl to uP line when Skippy 'off'.
'-EXCEL file creation working, added hourglass cursor
'-corrected bug with no default Magic Number
'-cleaned up GUI screen, got p/np input working
'-fixed bug in miles and time calculations - ignore FF bytes (bike stationary)
'-incr.data to full 64K available, fixed bug in excel fmt,add eecount
'-Added AutoGraf trap for end of data = '1' or 611mph, no wasteful bulk erase!
'
'          Connections:
'LPT pin                                  24C512 pin
'  2 (D0)                                 8 (Vcc)
'  3 (D1)                                 6 (Scl)
'  4 (D2)   via 10K                       5 (Sda)
'  5 (D3)   via cathode 1N5817            5 (Sda)
' 10 (ack)                                5 (Sda)
' 25 (gnd)                                4 (Vss),3/2/1 (addr.),7 (wp)
'
DIM Dataexcel(65536)    'was 4096, limiting pages to 32
DIM eedata(65536)
pwroff(1) = 0           '(32 pages x 128 bytes = 4096)
cldl(1) = 5
chdl(1) = 7
cldh(1) = 13
chdh(1) = 15
WriteC = HEXDEC("&HA0")    '24Cxx device code
ReadC = HEXDEC("&HA1")
'
```

```
'*********************************************************
'Form created with the help of Freeform 3

[setupMainWindow]

    '-----Begin main GUI window code

    nomainwin
    WindowWidth = 550
    WindowHeight = 410

    BackgroundColor$ = "cyan"
    ForegroundColor$ = "black"

    '-----Begin GUI objects code

    TextboxColor$ = "white"
    textbox #main.textbox1, 145, 52, 150, 25
    button #main.button2, "Test Connection", [TestConnection], UL, 15, 52, 114, 25
    statictext #main.statictext3, "Skippy", 230, 5, 150, 40
    textbox #main.textbox4, 360, 52, 45, 25
    statictext #main.statictext5, "LPT port found", 415, 55, 87, 20
    texteditor #main.textedit6, 15, 92, 325, 200
    ListboxColor$ = "white"
    listbox #main.listbox7, tiredia$(, [TireSize], 360, 92, 50, 100
    button #main.button8, "read Page1", [ReadPage1], UL, 15, 310, 109, 25
    statictext #main.statictext9, "Wheel Size", 420, 150, 68, 20
    textbox #main.textbox10, 360, 297, 30, 25
    statictext #main.statictext11, "Start Page", 400, 300, 63, 20
    textbox #main.textbox12, 360, 322, 30, 25
    statictext #main.statictext13, "# Pages", 400, 325, 50, 20
    textbox #main.textbox14, 420, 120, 50, 25
    button #main.button15, "make Excel", [MakeExcelFile], UL, 250, 310, 100, 25
    button #main.button16, "autograf", [AutoGraph],UL,135,310,100,25

    '-----End GUI objects code
    DATA "16","20","24","26","27","28"," "'read in tire sizes to ListBox
    i = 1
    WHILE value$ <> " "
       READ value$
       tiredia$(i) = value$
       i = i + 1
    WEND
```

```
   open "cbm Bike Logger" for window as #main
   print #main, "font ms_sans_serif 0 16"
   print #main.statictext3, "!font Comic_Sans_MS 18"   'large curly script
   print #main, "trapclose [quit]"

[FindLptAddr]
   PortAddr(1)=HEXDEC("&H3BC") ' all three choices for Lpt port address
   PortAddr(2)=HEXDEC("&H378") ' - most likely for modern Pentiums
   PortAddr(3)=HEXDEC("&H278")
   Lpta(1) = 0
   FOR i = 1 to 3

    OUT PortAddr(i),HEXDEC("&H55")
      Test1=INP(PortAddr(i))
      OUT PortAddr(i),HEXDEC("&HAA")
      Test2=INP(PortAddr(i))
      IF (Test1=HEXDEC("&H55"))AND(Test2=HEXDEC("&HAA"))THEN Lpta(1) =
PortAddr(i)
      IF (Test1=HEXDEC("&H55"))AND(Test2=HEXDEC("&HAA"))THEN EXIT FOR
   NEXT i
   PRINT #main.textbox4,DECHEX$(Lpta(1))
   IF Lpta(1) = 0 THEN PRINT #main.textbox4,"NONE!"
   OUT Lpta(1),cldh(1)     '  default, power on

   'set up defaults
   tire = 26             ' tire dia. in inches
   p = 0                 ' first page to read (128 byte pages)
   np = 2                ' number of pages to be read
   MagicNumber = (tire * 3.14159 * 32768 * 60) / (12 * 256 * 88)   'used by Excel
   PRINT #main.textbox14,tire;" inch"
   PRINT #main.textbox10,p
   PRINT #main.textbox12,np
  wait

[TestConnection]   'Perform action for the button named 'TestConnection'
      'checking for absent programmer
      bit = 64 AND INP(Lpta(1) + 1)
      bit = INT(bit / 64)
      hi = bit
      OUT Lpta(1), cldl(1)
      bit = 64 AND INP(Lpta(1) + 1)
      bit = INT(bit / 64)
      lo = bit
```

```
      IF hi - lo = 0 THEN PRINT #main.textbox1,"No BikeLogger found"
      IF hi - lo > 0 THEN PRINT #main.textbox1,"Talking to BikeLogger"
      OUT Lpta(1), cldh(1)
    wait

[TireSize]   'Perform action for the listbox named 'TireSize'
      PRINT #main.listbox7, "selection? tire$"
      PRINT #main.textbox14, tire$;" inch"

      tire = VAL(tire$)
      MagicNumber = (tire * 3.14159 * 32768 * 60) / (12 * 256 * 88)
    wait

'***********************************************************

[ReadPage1]   'Perform action for the button named 'Page1'
      CURSOR HOURGLASS
      i = 0
[Repeat]
      i = i + 1
      IF i > 3 THEN notice "Busy"     'give up after 3 tries
      IF i > 3 THEN wait
      CALL Start
      Byte1 = WriteC
      CALL Xmit Byte1
      IF Byte1 < 0 THEN GOTO [Repeat]      'logger is busy or not there
      Byte1 = 0
      CALL Xmit Byte1     'send address 00 00, i.e. Page 1.

     Byte1 = 0
      CALL Xmit Byte1
      CALL Start
      Byte1 = ReadC
      CALL Xmit Byte1
      FOR k = 1 TO 128
        newb = ReadByte()
        newb$ = DECHEX$(newb)
        IF newb < 16 THEN newb$ = "0" + newb$
        print #main.textedit6, newb$;" ";
        IF INT(k/16)*16 = k THEN print #main.textedit6, chr$(13)+ chr$(10);'13=CR,10=LF
        IF k < 128 THEN CALL Ack ELSE bit = ReadBit()    'Ack for new read, NoAck for finish
        NEXT k
      CALL StopC
      CURSOR NORMAL
```

```
  wait

'*****************************************************************

[AutoGraph]      'Perform action for the button named 'autograf'
     CURSOR HOURGLASS
     file1$ = time$()
     timex$ = LEFT$(file1$,2)+"-"+MID$(file1$,4,2)+"-"+ RIGHT$(file1$,2)
     file3$ = timex$ + " " + date$() + "_bike.csv" 'invent EXCEL filename
     p1lower = 0
     p1upper = 0
     i = 0
[Repeat3]
     i = i + 1
     IF i > 3 THEN notice "Busy"     'give up after 3 tries
     IF i > 3 THEN wait
     CALL Start
     Byte = WriteC
     CALL Xmit Byte
     IF Byte < 0 THEN GOTO [Repeat3]
     Byte = p1upper
     CALL Xmit Byte
     Byte =  p1lower
     CALL Xmit Byte
     CALL Start
     Byte = ReadC
     CALL Xmit Byte
     q  = 65024        '(max. 256 x 254, or 508 pages of 128 bytes)
     Etime = 0
     m = 0
     ' read the eeprom memory
     FOR k  = 1 TO q
          newb = ReadByte()
          IF newb = 1 THEN EXIT FOR   'marker for end of data – Mach 1 speed
          eedata(k) = newb
          IF newb <> 255 THEN Etime = Etime + newb
          IF newb <> 255 THEN m = m + 1
          IF newb = 0 THEN newb = 1   'unlikely but avoids /0 error
          Dataexcel(k ) = MagicNumber / newb
          IF newb  = 255 THEN Dataexcel(k ) = 0
          IF k  < q  THEN CALL Ack ELSE dbit = ReadBit()
     NEXT k
     np = k / 128
     CALL StopC
```

```
    ' create the EXCEL file
    OPEN file3$ FOR OUTPUT AS #1
    Etime  = Etime * 256 / (60 * 32768 )
    EMins  = INT(Etime )
    Esecs  = (60 * Etime ) - (60 * INT(Etime))
    PRINT #1, file3$
    PRINT #1, tire;" inch wheel"
    PRINT #1, "Number of 128 byte pages read is "; USING("###.#",np)
    PRINT #1,"duration of recording is ";EMins;"mins ";INT(Esecs);"secs"
    Miles  = m  * tire  * 3.14159 / (12  * 5280 )
    PRINT #1,USING("###.##", Miles );" miles travelled"
    FOR i  = 1 TO k
        PRINT #1, i ; ","; USING("###.#",Dataexcel(i));","; USING("###",eedata(i))
    NEXT i
    PRINT #main.textedit6,file3$; "   auto-excel file made"
    close #1
    CURSOR NORMAL
    'PRINT #main.textedit6,p;" ";np
   wait
'
'*****************************************************************

[MakeExcelFile]   'Perform action for the button named 'Excel'
    CURSOR HOURGLASS
    file1$ = time$()
    timex$ = LEFT$(file1$,2)+"-"+MID$(file1$,4,2)+"-"+ RIGHT$(file1$,2)
    file3$ = timex$ + " " + date$() + "_bike.csv" 'invent EXCEL filename
    PRINT #main.textbox10, "!contents?"
    INPUT #main.textbox10, p$
    p=VAL(p$)
    p1upper  = INT(p/2)              'for 128 byte page 24C512
    IF p1upper * 2 = p  THEN p1lower  = 0 ELSE p1lower  = 128
    i = 0
[Repeat2]
    i = i + 1
    IF i > 3 THEN notice "Busy"     'give up after 3 tries
    IF i > 3 THEN wait
    CALL Start
    Byte = WriteC
    CALL Xmit Byte
    IF Byte < 0 THEN GOTO [Repeat2]
    Byte = p1upper
    CALL Xmit Byte
    Byte =  p1lower
```

```
    CALL Xmit Byte
    CALL Start
    Byte = ReadC
    CALL Xmit Byte
    PRINT #main.textbox12, "!contents?"
    INPUT #main.textbox12, np$
    np = VAL(np$)
    q  = 128 * np       '128 byes/page assumed (as in 24C512 eeprom)
    Etime = 0
    m = 0
    ' read the eeprom memory
    FOR k  = 1 TO q
        newb = ReadByte()
        eedata(k) = newb
        IF newb <> 255 THEN Etime = Etime + newb
        IF newb <> 255 THEN m = m + 1
        IF newb = 0 THEN newb = 1   'unlikely but avoids /0 error
        Dataexcel(k ) = MagicNumber / newb
        IF newb  = 255 THEN Dataexcel(k ) = 0
        IF k  < q  THEN CALL Ack ELSE dbit = ReadBit()
    NEXT k
    CALL StopC
    ' create the EXCEL file
    OPEN file3$ FOR OUTPUT AS #1
    Etime  = Etime * 256 / (60 * 32768 )
    EMins  = INT(Etime )
    Esecs  = (60 * Etime ) - (60 * INT(Etime))
    PRINT #1, file3$
    PRINT #1, tire;" inch wheel"
    PRINT #1, "Start page is ";p;"   and number of 128 byte pages read is ";np
    PRINT #1,"duration of recording is ";EMins;"mins ";INT(Esecs);"secs"
    Miles  = m  * tire  * 3.14159 / (12  * 5280 )
    PRINT #1,USING("###.##", Miles );" miles travelled"
    FOR i  = 1 TO q
        PRINT #1, i ; ","; USING("###.#",Dataexcel(i));","; USING("###",eedata(i))
    NEXT i
    PRINT #main.textedit6,file3$; "   excel file created."
    close #1
    CURSOR NORMAL
    'PRINT #main.textedit6,p;" ";np
   wait

[quit] 'End the program when 'X' clicked
  OUT Lpta(1), pwroff(1)
```

```
close #main
end

'******************************************************

SUB Ack                          'sends ACK pulse, I2C
OUT Lpta(1), cldl(1)
OUT Lpta(1), chdl(1)
OUT Lpta(1), cldl(1)
OUT Lpta(1), cldh(1)
END SUB

FUNCTION ReadBit()                 'I2C strobes clock, reads bit
OUT Lpta(1), chdh(1)               'also used for NO ACK
bit = 64 AND INP(Lpta(1) + 1)        'mask Lpt ack bit 6
bit = INT(bit / 64)                'move to lsb
OUT Lpta(1), cldh(1)
ReadBit = bit
END FUNCTION

FUNCTION ReadByte()                'I2C reads a byte
Byte = 0
FOR k  = 1 TO 8
      Byte = Byte * 2
      bit = ReadBit()
      Byte = Byte + bit
NEXT k
ReadByte = Byte
END FUNCTION

SUB Start                        'sends I2C Start pulse
OUT Lpta(1), chdh(1)
OUT Lpta(1), chdl(1)               'start signal
OUT Lpta(1), cldl(1)
OUT Lpta(1), cldh(1)               'default condition for SDA
END SUB

SUB StopC                         'sends I2C Stop pulse
OUT Lpta(1), cldl(1)
OUT Lpta(1), chdl(1)
OUT Lpta(1), chdh(1)               'stop signal
OUT Lpta(1), cldh(1)               'default
END SUB
```

```
SUB Xmit Byte                         'I2C writes a byte
FOR i  = 1 TO 8
 IF Byte  AND 128 THEN OUT Lpta(1), cldh(1)  ELSE OUT Lpta(1), cldl(1)
 IF Byte  AND 128 THEN OUT Lpta(1), chdh(1)  ELSE OUT Lpta(1), chdl(1)
 IF Byte  AND 128 THEN OUT Lpta(1), cldh(1)  ELSE OUT Lpta(1), cldl(1)
 Byte = Byte * 2
NEXT i
OUT Lpta(1), cldh(1)
bit = ReadBit()                       'sends clk for ACK
IF bit THEN Byte = -1                 '-1 = busy
END SUB
```