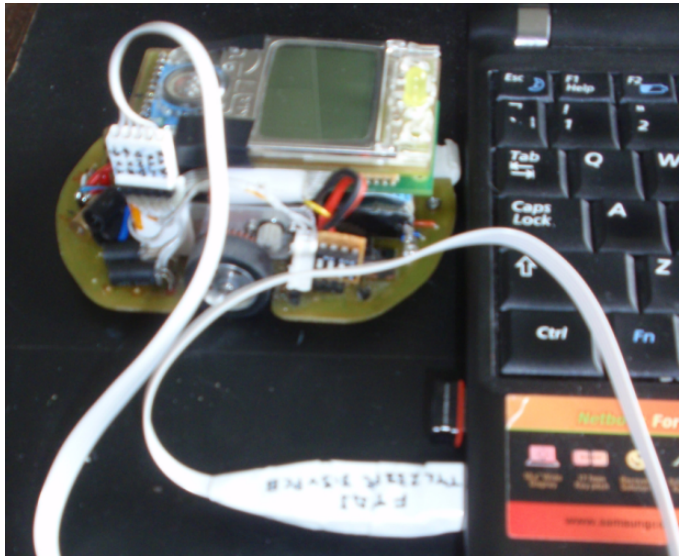
Data logging on the dsPIC

Or how to find out what is really happening in your mouse

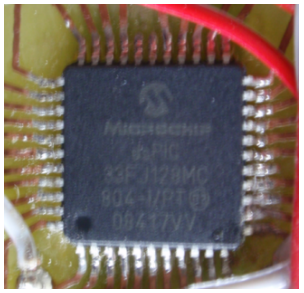


David Hannaford and a

dsPIC33FJ128MC804

For MINOS 2011

What is a dsPIC33FJ128MC804?



A 16 bit 80Mhz Microchip device giving 40MIPS with 8192 16bit words of data storage

44	Pins - surface mount
128k	**Program Flash Memory(Kbyte)**
16k	**RAM (Kbyte)**
26	**Remappable Pins**
5	**16-bit Timers**
4	Input Capture
4	Output Compare Standard PWM
6,2	**Motor Control PWM (Channels)**
2	**Quadrature Encoder Interfaces**
2	**UART**
2	**SPI**

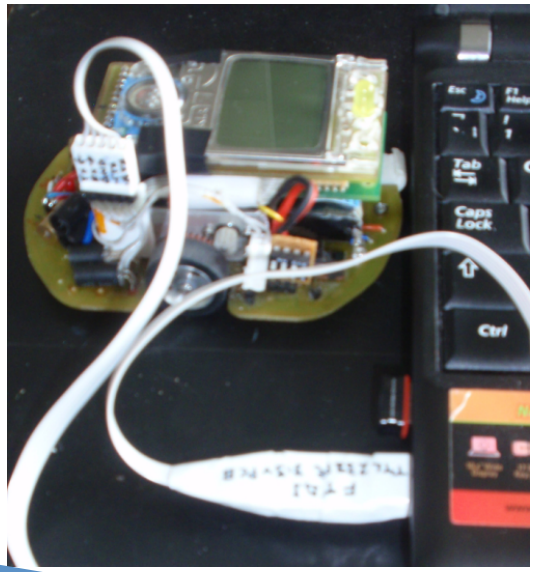
1	ECAN.
3	External Interrupts
1	RTCC
1	I2C.
1	CRC Generator
9	**10-bit/12-bit ADC(Channels)**
1	6-pin 16-bit DAC
1/1	Analogue Comparator (2 Channels/Voltage Regulator)
11	8-bit Parallel Master Port (Address Lines)
35	I/O Pins
QFN / TQFP	Packages

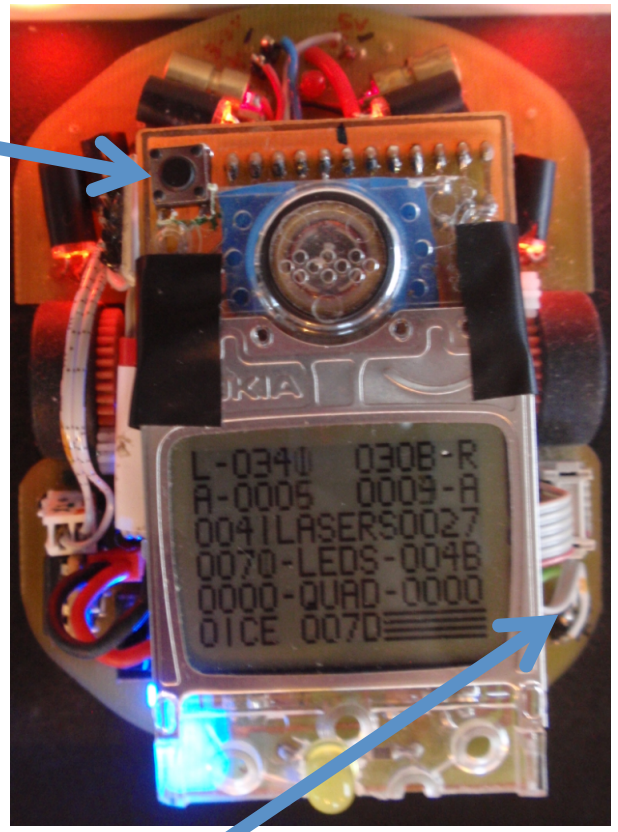
Logging data into RAM

- Limited by 8k words data RAM . Have 1664 words used for fields, mazes, flood, and move lists. Assigned 5504 for log items
- Collect multiple data items – e.g. target & actual speed x 2
- Data most useful if logged every calculation cycle (every millisecond for my mice) so 1000 sets of readings per second
- So for 4 data items I can collect 1.376 seconds worth – enough to cross a cell, or do a turn, or accelerate up to speed, etc.
- Logging routine called every time I start the main loop
- One flag (logena) to start or stop logging, so easy to switch on or off
- All the fields we want to log defined in one place – easy to change
- Automatically stop logging when no more room (for first n readings) or wrap around (for last n readings)

Get ready to display/ transfer data

- Button 1 on mouse to stop it and trigger data display / transfer
- Option to display data on Nokia screen or transfer to PC





- Connect FTDI cable to mouse and USB port on PC – while keeping mouse switched on
- Start terminal program on PC
- Press button 2 to transfer up to 500 sets of logged data

Setting up USB to serial cable

- No serial port on my PC so use an FTDI USB to serial converter
- Get 3.3 or 5v version to match your mouse's CPU supply voltage

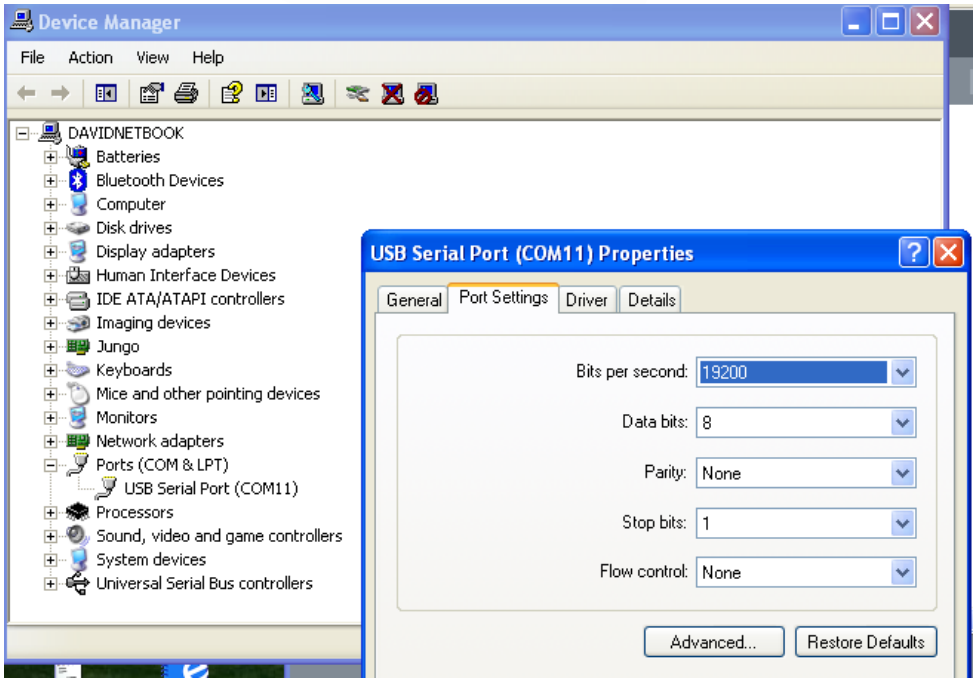
USB-RS232-WE-1800-BT_3.3 or

USB-RS232-WE-1800-BT_5.0

• Load the Virtual Com Port driver for it from FTDI site http://www.ftdichip.com/Drivers/VCP.htm

• Check com port settings in control panel
  - system – device manager

In my case - USB serial Port COM11



Setting up the UART on the dsPIC

- Assign pins to UART: Transmit(Tx), Receive(Rx), Clear To Send(CTS), Ready To Send(RTS)
- Set bits to say all pins being used (U1MODE)
- Set parity and stop bit settings (U1MODE)
- Set baud rate (U1BRG)
- Enable UART module to switch it on (U1MODE)
- Enable Transmit (U1STA)
- Sample dsPIC code for UART set up at end of slides

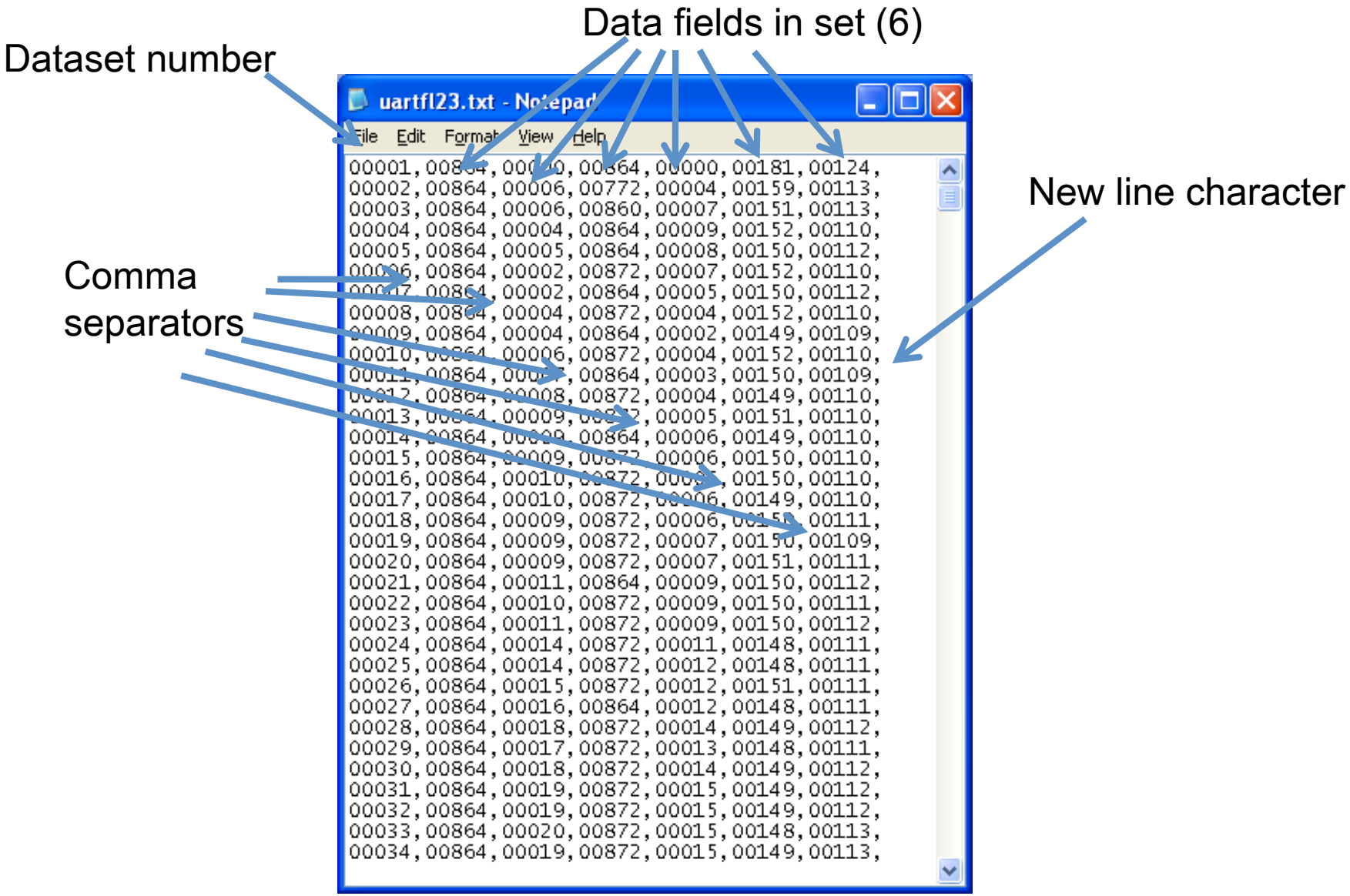
Formatting the logged data

- Before sending the data convert it to CSV format and send it in sets with a set item number at the start
- Routine needed to convert binary logged data to decimal text characters so easily processed at the PC end
- For 16 bit data we need 5 characters plus a leading sign then a comma separator, for 8 bit data we just need 3 characters plus a sign

Binary to character conversion

- Set up 5 binary counts – for the 5 digits
- If bit 0,1 or 2 is set add 1, 2 or 4 to count5
- If bit 3 is set add 8 to count 5
- If bit 4 is set add 1 to count 4 and 6 to count 5
- If bit 5 is set add 3 to count 4 and 2 to count 5
- .....After each addition, check if count >= 10 and carry 1 if needed
- If bit 15 set subtract from zero and set minus sign

Sample data stream



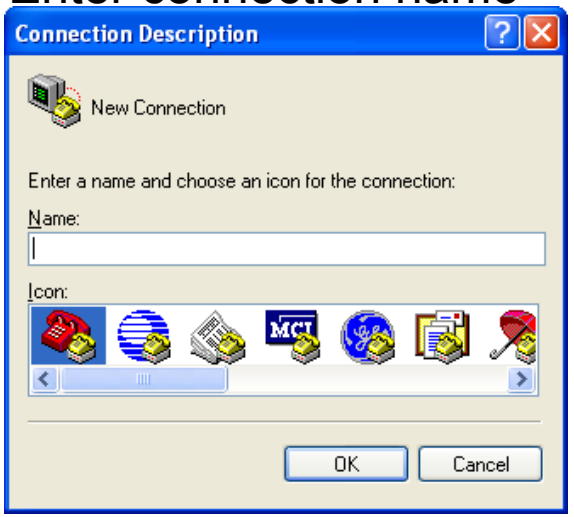
Assembling the character stream

- First send the data set number (0,1,2,3 etc)
- For each number to be sent, call the binary to character routine – you get 5 counts & sign
- Each count will be in range 0 to 9
- For each number 0 to 9 use a routine that sends 8 bits corresponding to the Ascii character for that number E.g. 0 = 01001000 (48), 1 = 01001001 (49)
- Send sign if required (minus = 45) then 5 counts, followed by a comma (44)
- Repeat for each number in the data set then send carriage return (13) and line feed(10)

PC data receive

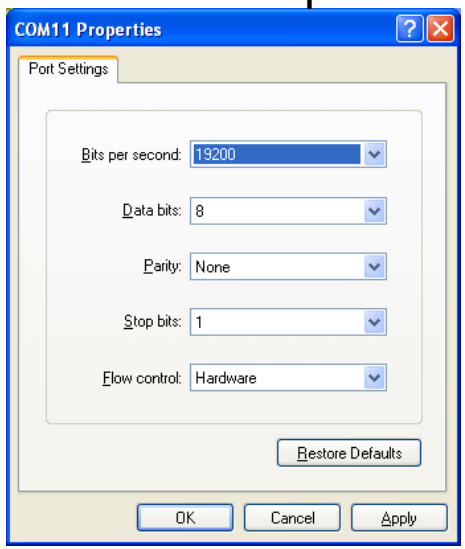
Put the FTDI device in a USB socket and open Hyper Terminal or similar XP terminal emulation program. If you are on Windows 7 get hold of these 2 files **hypertrm.dll** and **hypertrm.exe** from any old XP machine and put them in any folder on your Windows 7 PC

Enter connection name

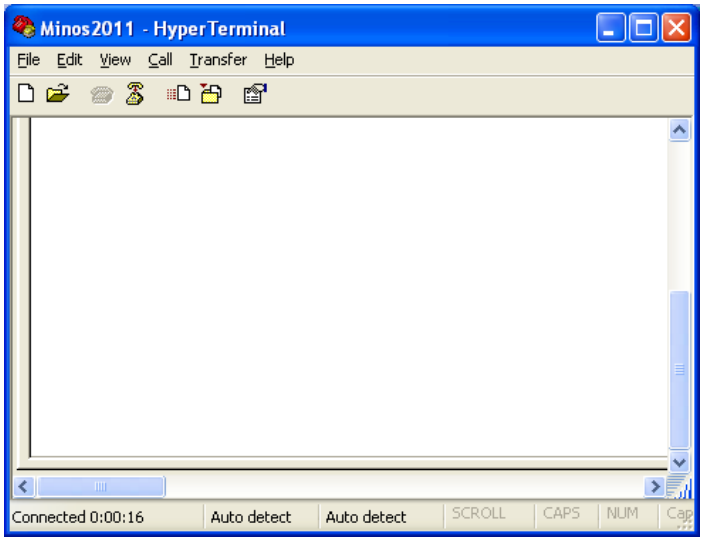


Check COM port

Amend properties for correct speed, data and stop bits



Ready to receive data

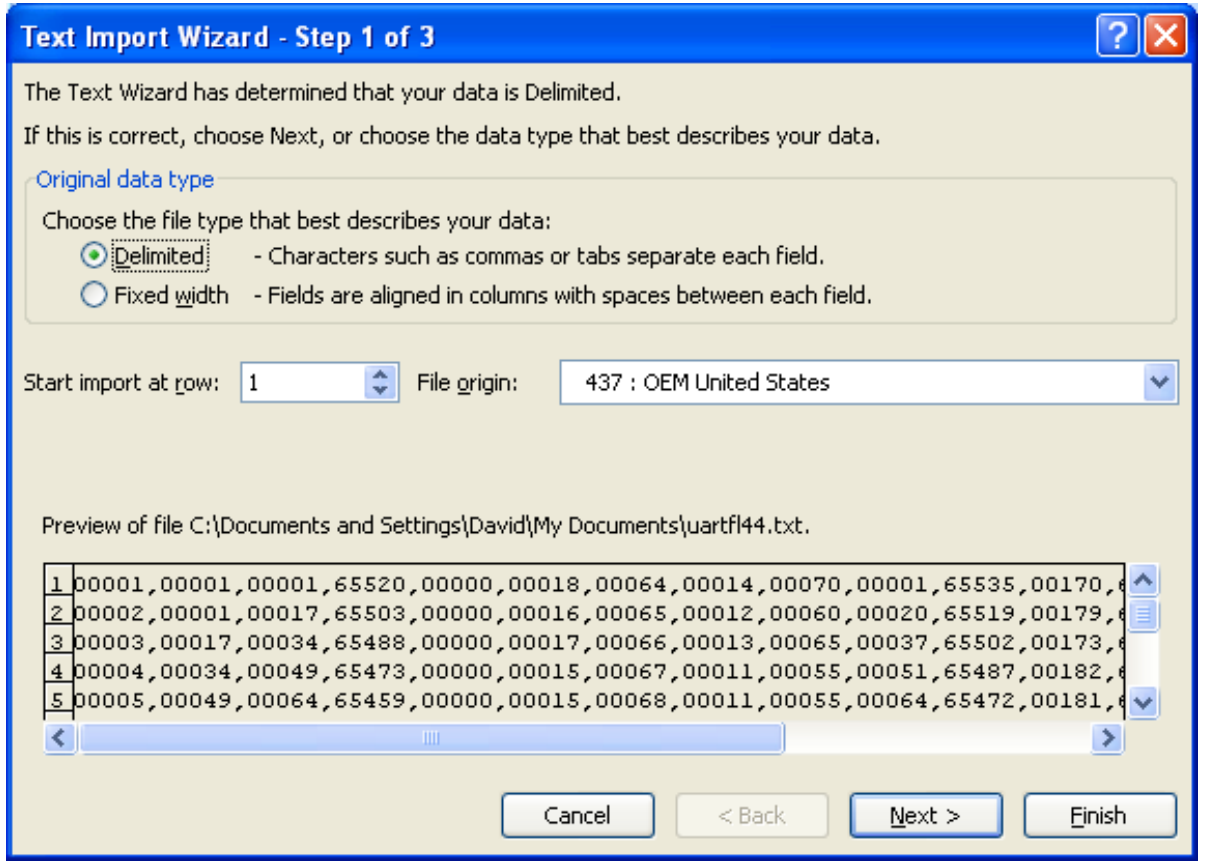


Transfer data from mouse to PC

- Start transfer from mouse
- Data goes into Hyper Terminal buffer
- Can only send a maximum of 500 data sets at a time, so if more we need to send as separately initiated batches of 500
- Open Word. Highlight data in buffer and copy to clipboard. Paste into Word. Repeat if more than 500 data sets.
- Save data from Word as a TXT format file

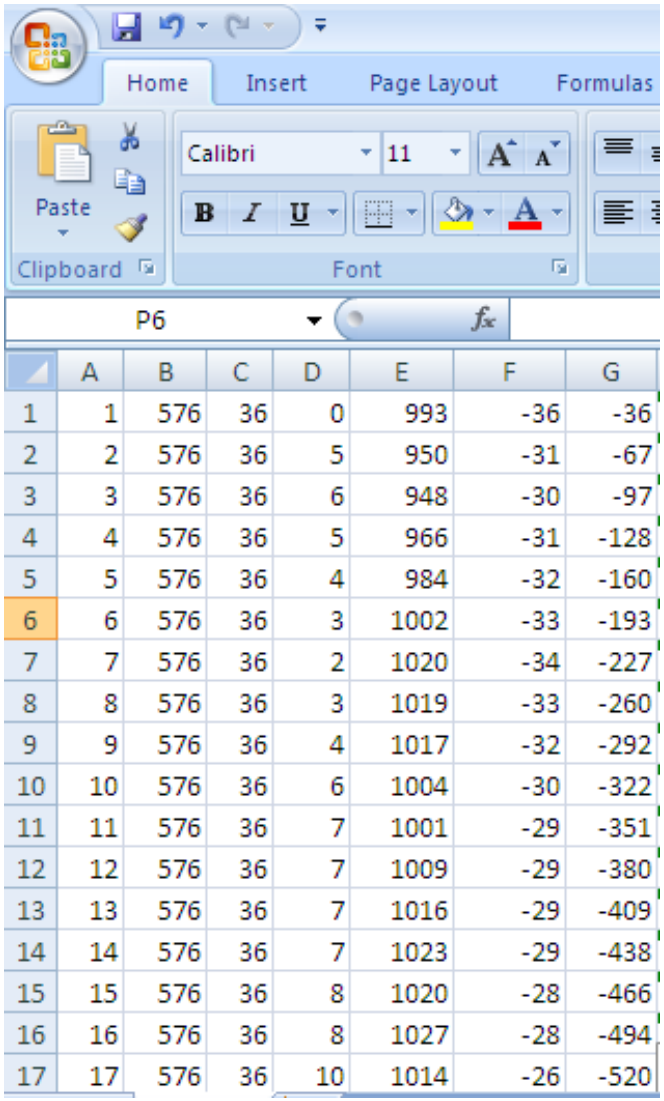
Open Microsoft Excel

- From **within** Excel Open the saved TXT file
- Text import wizard will launch and ask what delimiters you are using
- Select: Delimited, Comma, General , Finish



Imported data

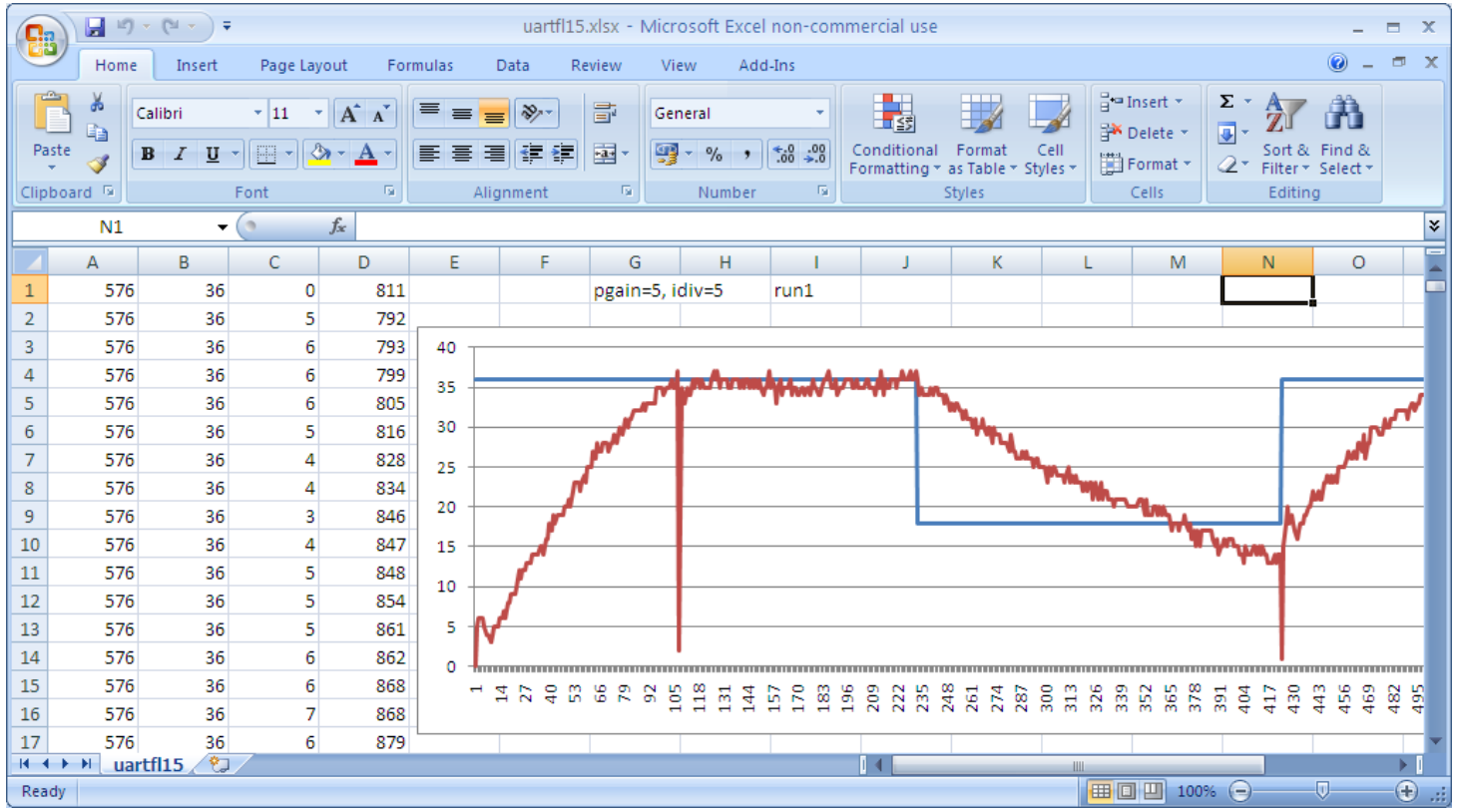
- Data will appear as sets in Excel.
- 6 items being logged each millisecond in this example
- Col A is the dataset number
- Add a header line with names of fields



	A	B	C	D	E	F	G
1	1	576	36	0	993	-36	-36
2	2	576	36	5	950	-31	-67
3	3	576	36	6	948	-30	-97
4	4	576	36	5	966	-31	-128
5	5	576	36	4	984	-32	-160
6	6	576	36	3	1002	-33	-193
7	7	576	36	2	1020	-34	-227
8	8	576	36	3	1019	-33	-260
9	9	576	36	4	1017	-32	-292
10	10	576	36	6	1004	-30	-322
11	11	576	36	7	1001	-29	-351
12	12	576	36	7	1009	-29	-380
13	13	576	36	7	1016	-29	-409
14	14	576	36	7	1023	-29	-438
15	15	576	36	8	1020	-28	-466
16	16	576	36	8	1027	-28	-494
17	17	576	36	10	1014	-26	-520

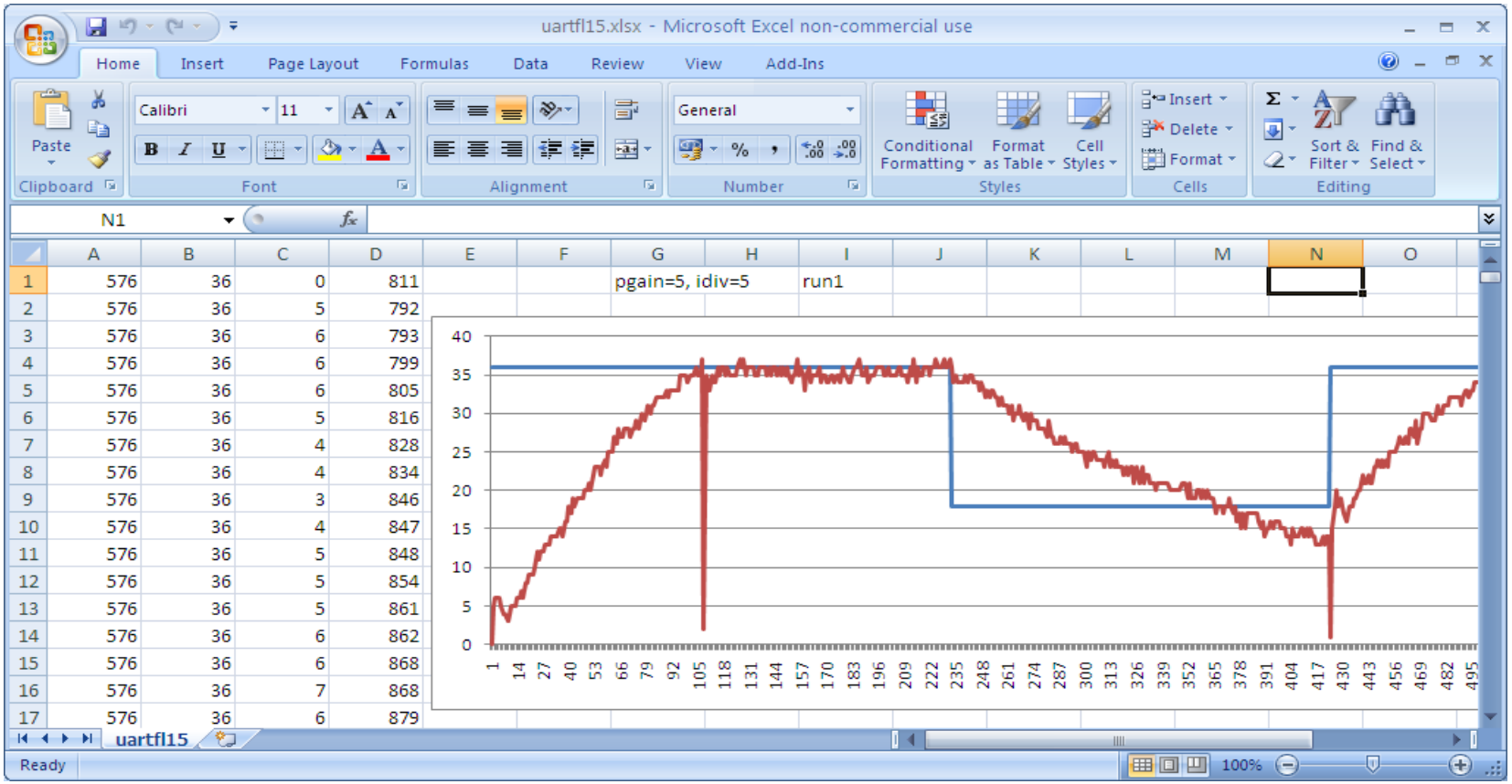
Turn data into a graph

- Select data and insert - line chart - 2D
- Blue = target speed, red = actual (encoder pulses)



Analyse data – spot problems

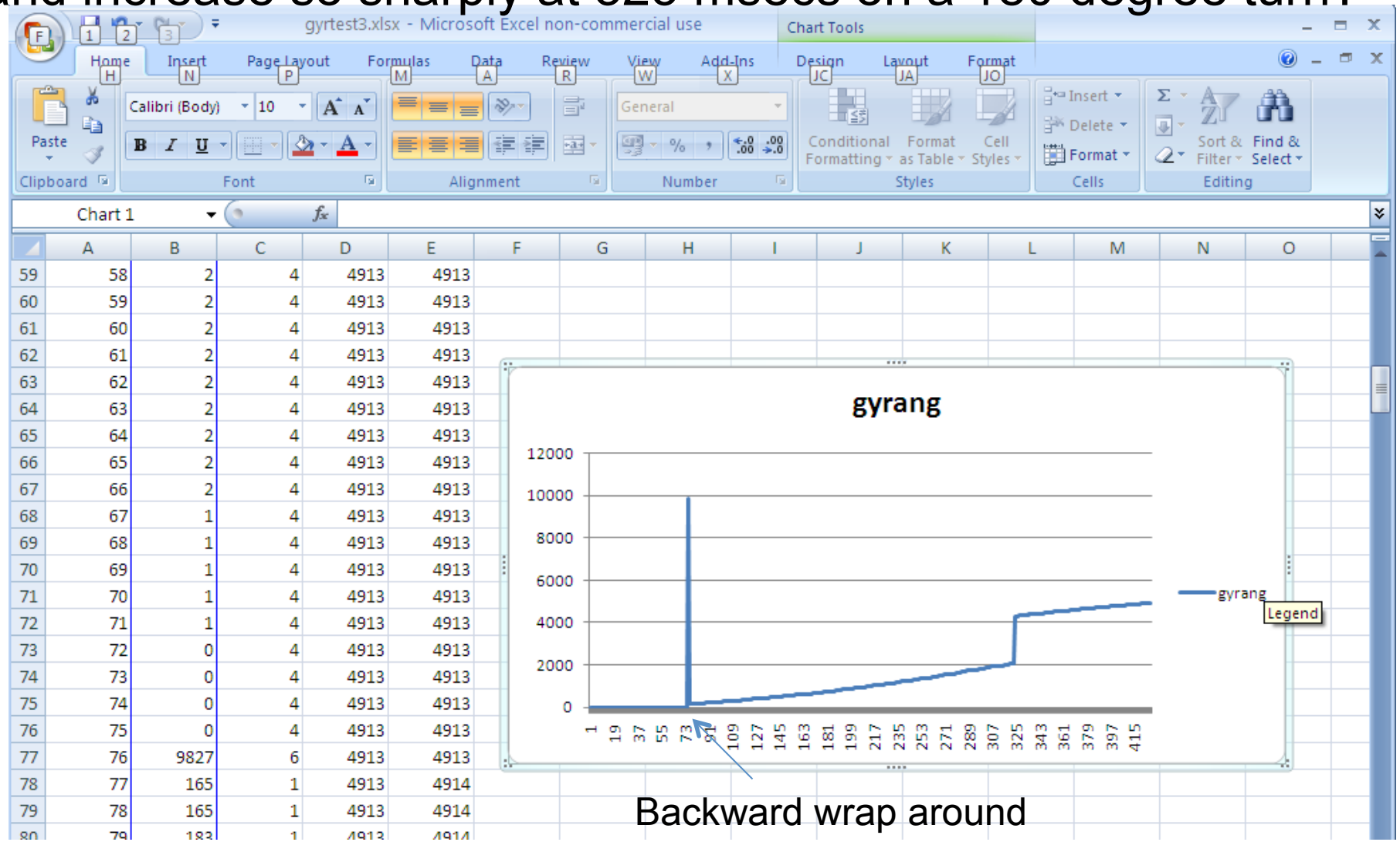
- E.g. Why do the encoder counts drop to zero at 105 and 420 milliseconds?



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	576	36	0	811			pgain=5, idiv=5		run1						
2	576	36	5	792											
3	576	36	6	793											
4	576	36	6	799											
5	576	36	6	805											
6	576	36	5	816											
7	576	36	4	828											
8	576	36	4	834											
9	576	36	3	846											
10	576	36	4	847											
11	576	36	5	848											
12	576	36	5	854											
13	576	36	5	861											
14	576	36	6	862											
15	576	36	6	868											
16	576	36	7	868											
17	576	36	6	879											

Analyse data – spot problems

Why does gyro angle jump to value 9827 after 77 msecs and increase so sharply at 325 msecs on a 180 degree turn?

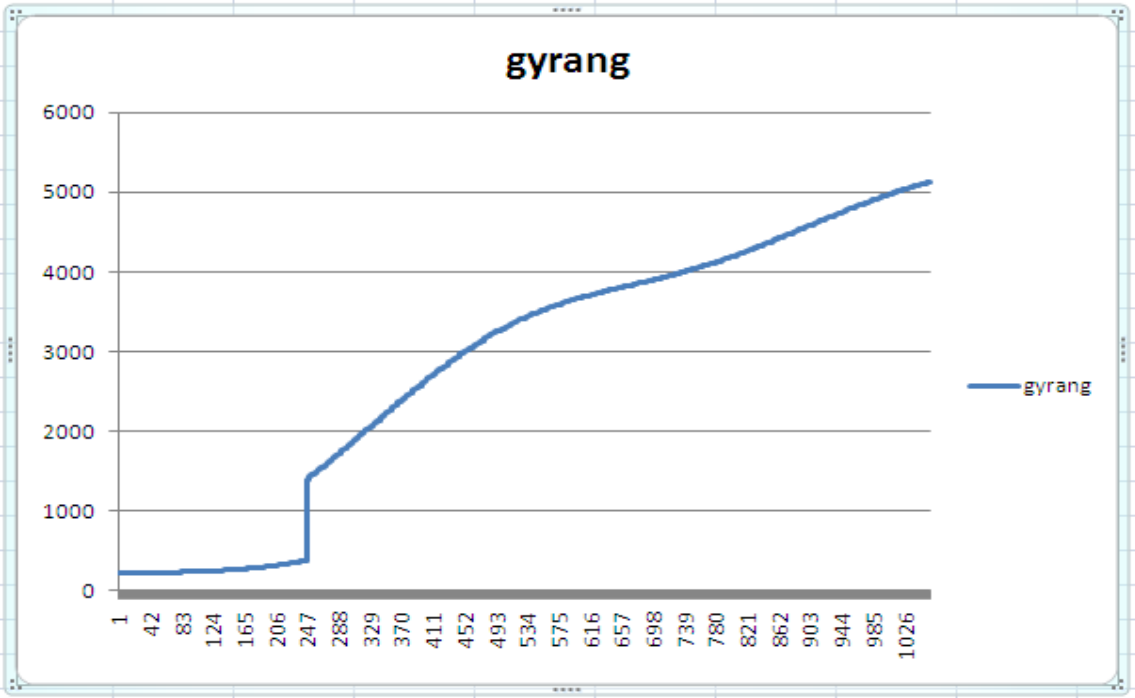


	A	B	C	D	E
59	58	2	4	4913	4913
60	59	2	4	4913	4913
61	60	2	4	4913	4913
62	61	2	4	4913	4913
63	62	2	4	4913	4913
64	63	2	4	4913	4913
65	64	2	4	4913	4913
66	65	2	4	4913	4913
67	66	2	4	4913	4913
68	67	1	4	4913	4913
69	68	1	4	4913	4913
70	69	1	4	4913	4913
71	70	1	4	4913	4913
72	71	1	4	4913	4913
73	72	0	4	4913	4913
74	73	0	4	4913	4913
75	74	0	4	4913	4913
76	75	0	4	4913	4913
77	76	9827	6	4913	4913
78	77	165	1	4913	4914
79	78	165	1	4913	4914

Analyse & spot problems

Why the sudden jump in gyro angle at 250 msec ?

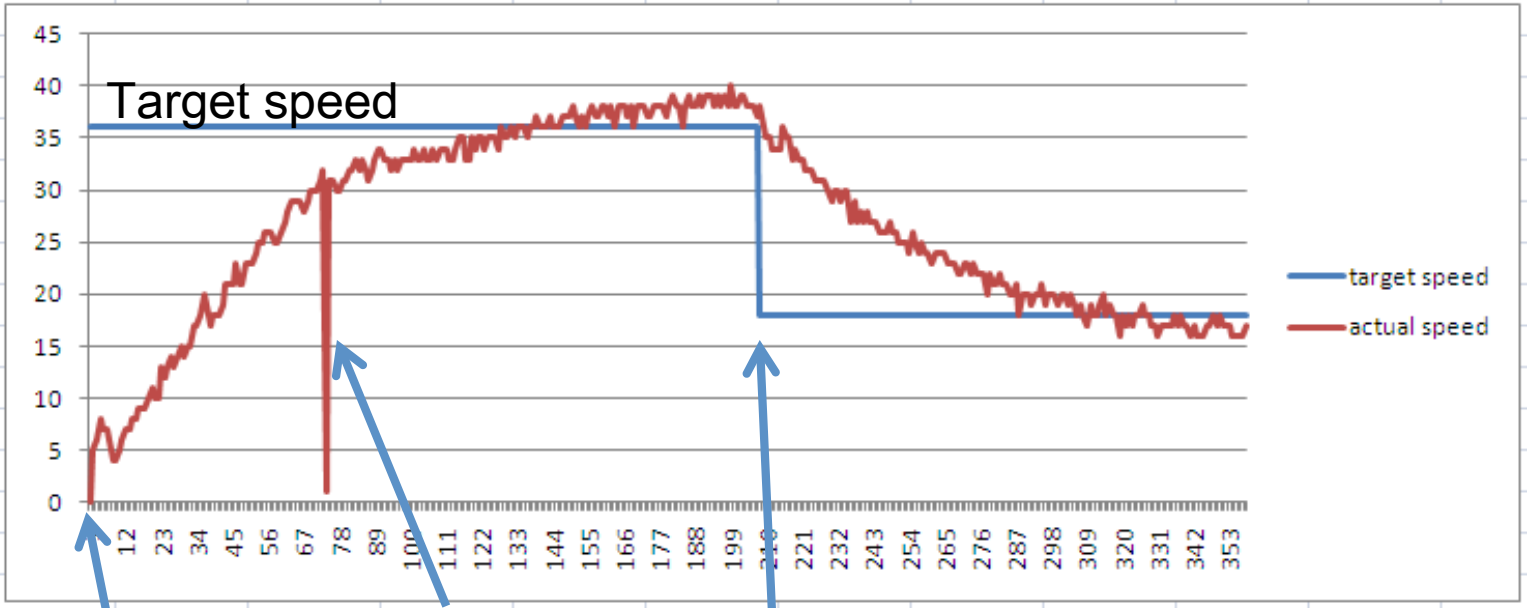
	A	B	C	D	E
236	235	362	4	4913	5131
237	236	362	4	4913	5131
238	237	362	4	4913	5131
239	238	369	4	4913	5131
240	239	369	4	4913	5131
241	240	369	4	4913	5131
242	241	369	4	4913	5131
243	242	377	4	4913	5131
244	243	377	4	4913	5131
245	244	377	4	4913	5131
246	245	377	4	4913	5131
247	246	377	4	4913	5131
248	247	385	4	4913	5131
249	248	1401	4	4913	5131
250	249	1401	4	4913	5131
251	250	1430	4	4913	5131
252	251	1430	4	4913	5131
253	252	1430	4	4913	5131
254	253	1459	4	4913	5131
255	254	1459	4	4913	5131
256	255	1459	4	4913	5131
257	256	1459	4	4913	5131
258	257	1488	4	4913	5131
259	258	1488	4	4913	5131
260	259	1488	4	4913	5131
261	260	1488	4	4913	5131
262	261	1518	4	4913	5131
263	262	1518	4	4913	5131



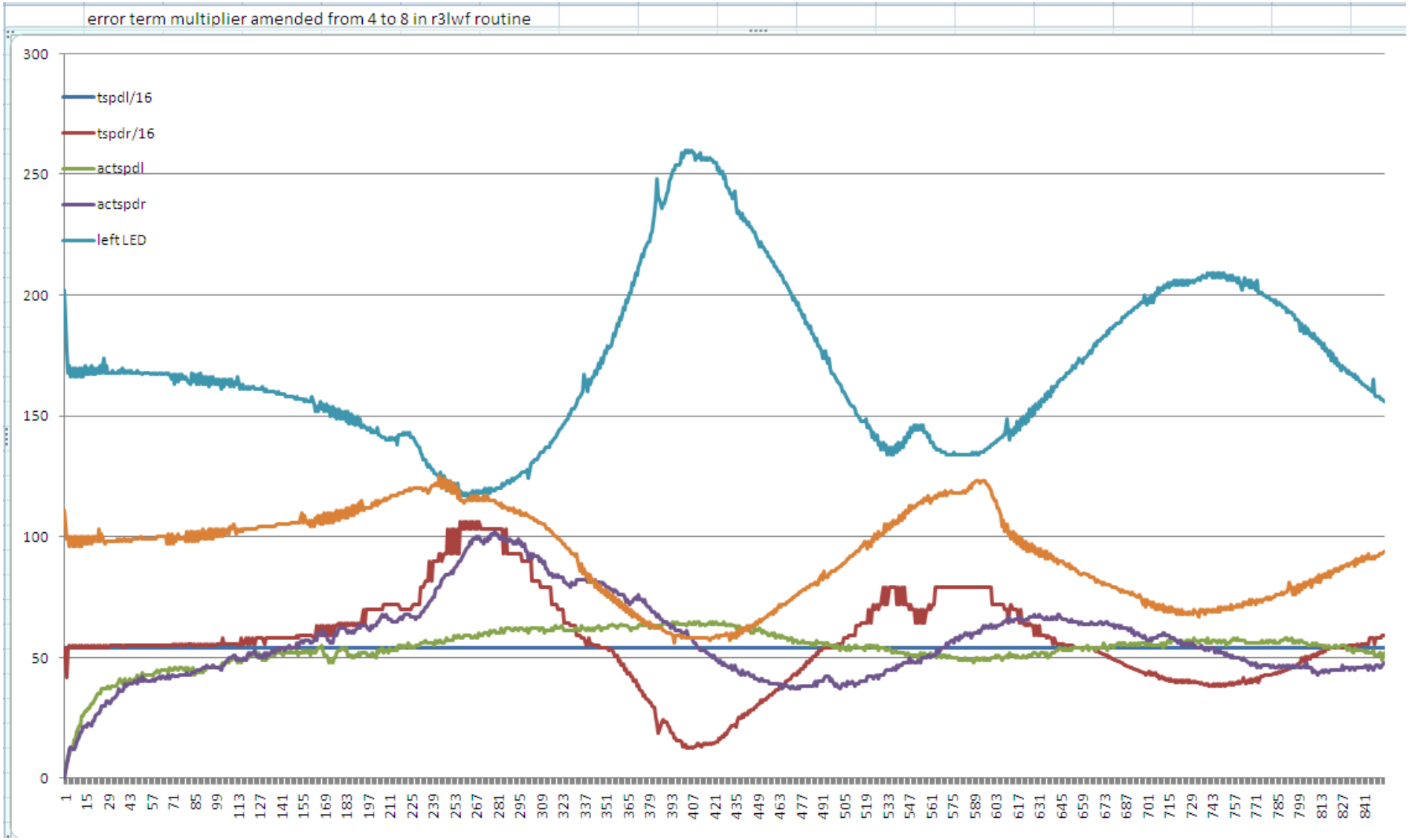
Assess speed change reactivity

With given PID parameters

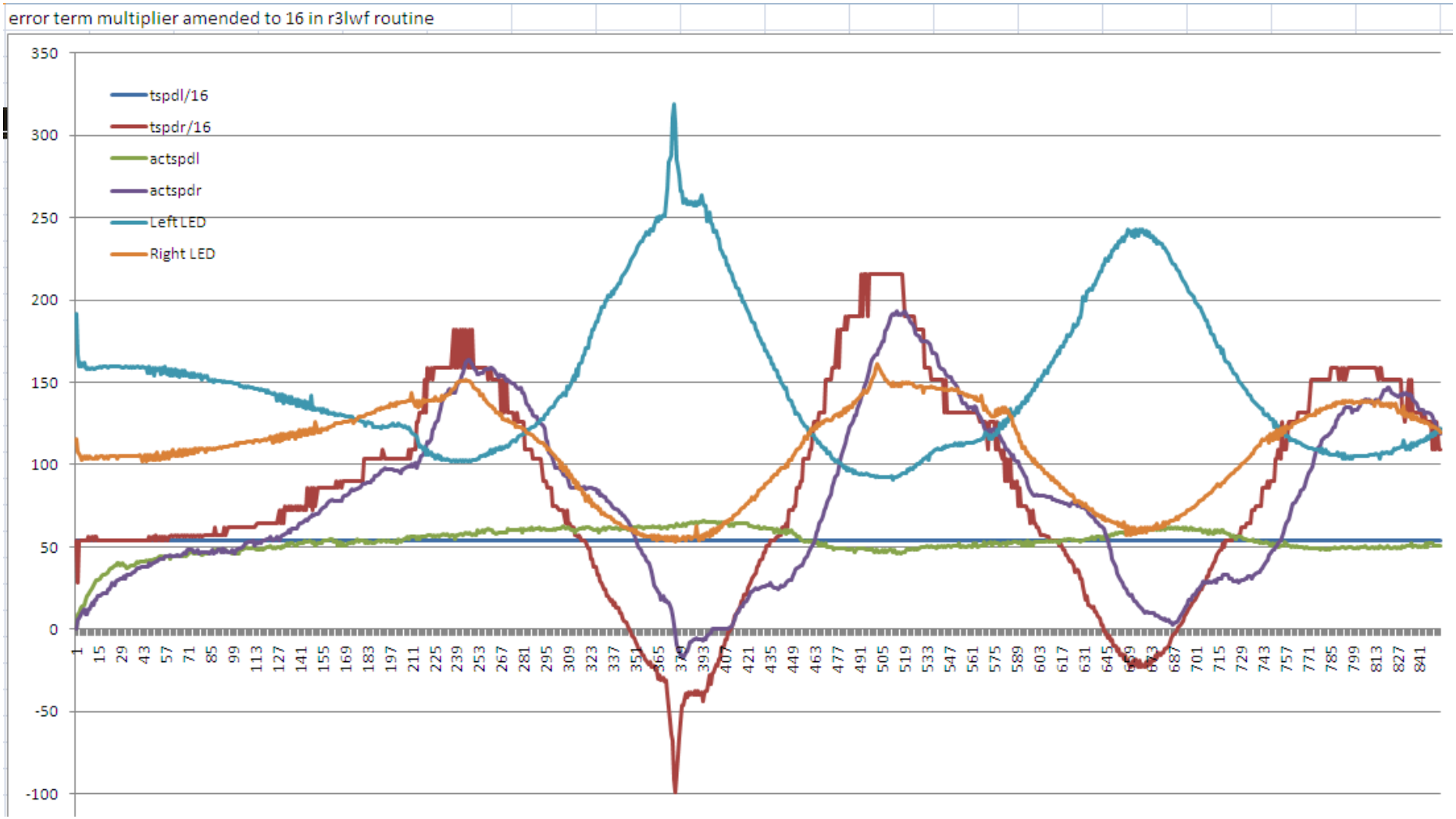
	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	target speed	actual speed		parameters		pgain = 5, idiv =4		run1						
2	36	0												
3	36	5												
4	36	6												
5	36	8												
6	36	7												
7	36	7												
8	36	6												
9	36	4												
10	36	4												
11	36	5												
12	36	6												
13	36	7												
14	36	7												
15	36	8												
16	36	8												
17	36	9												
18	36	9												
19	36	9												



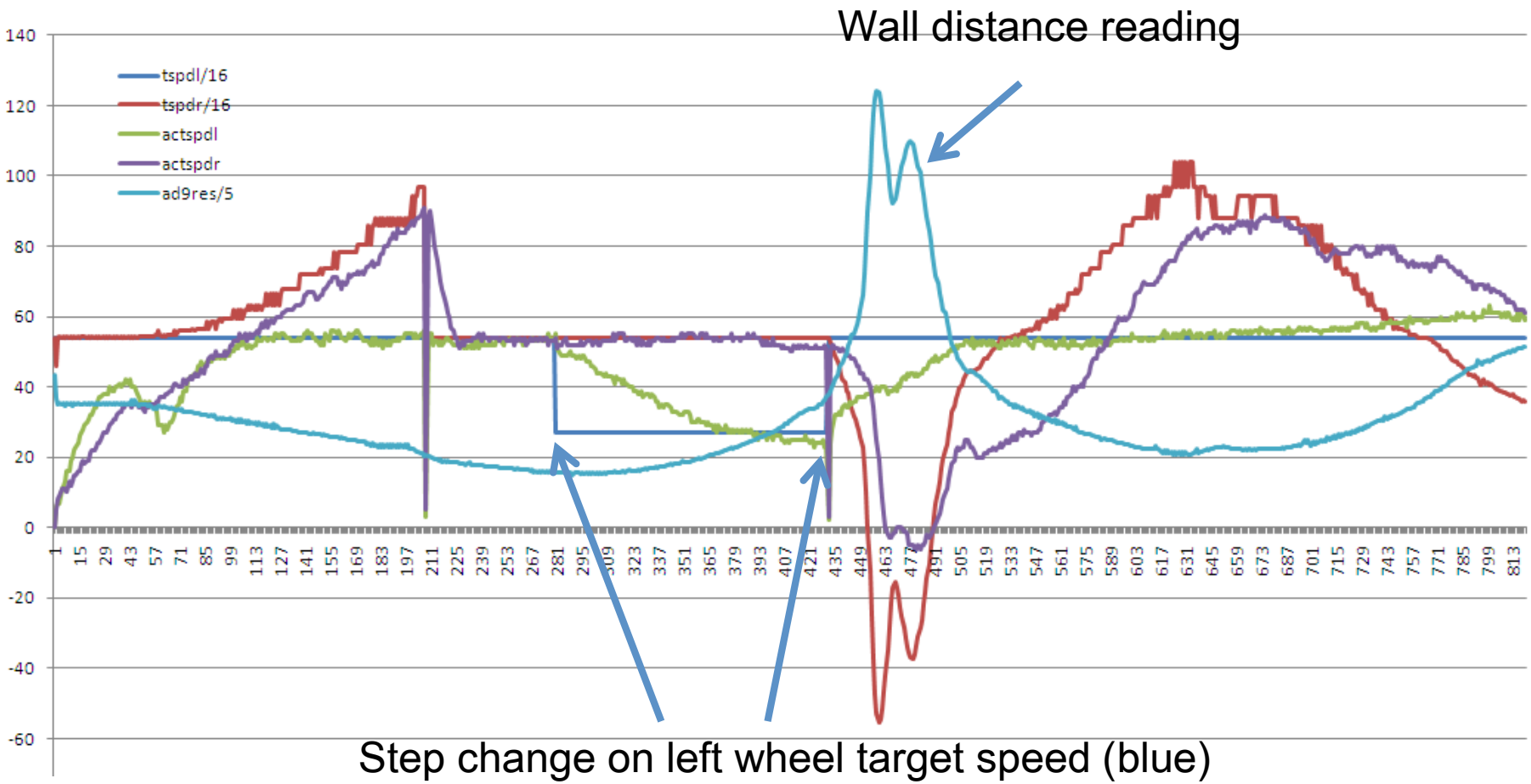
Check impact of parameter changes



Error term multiplier increased



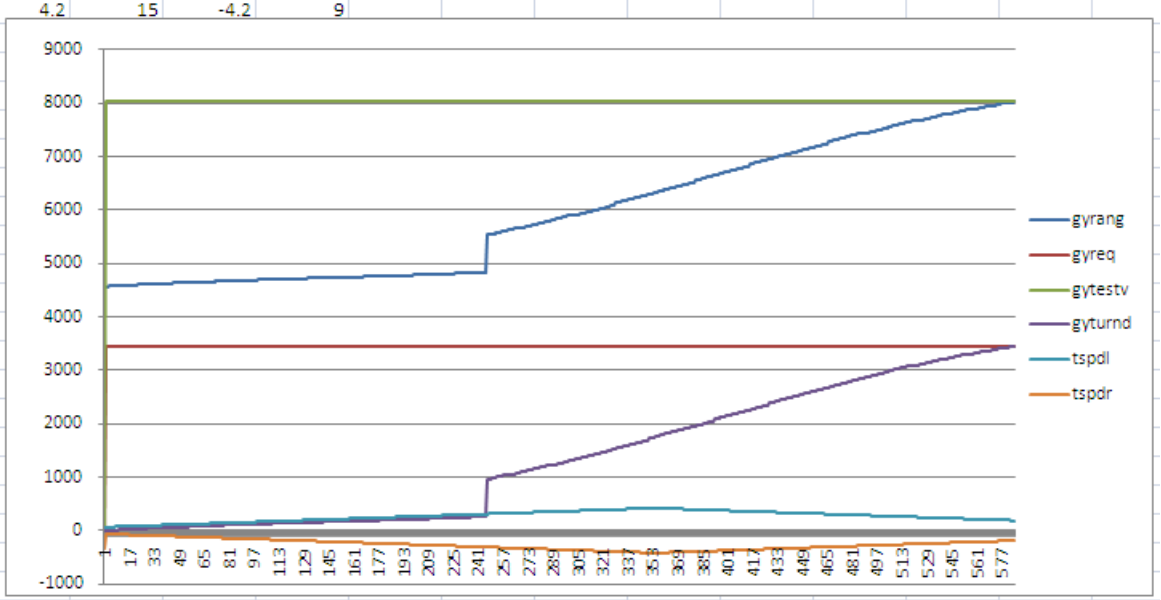
How does it perform

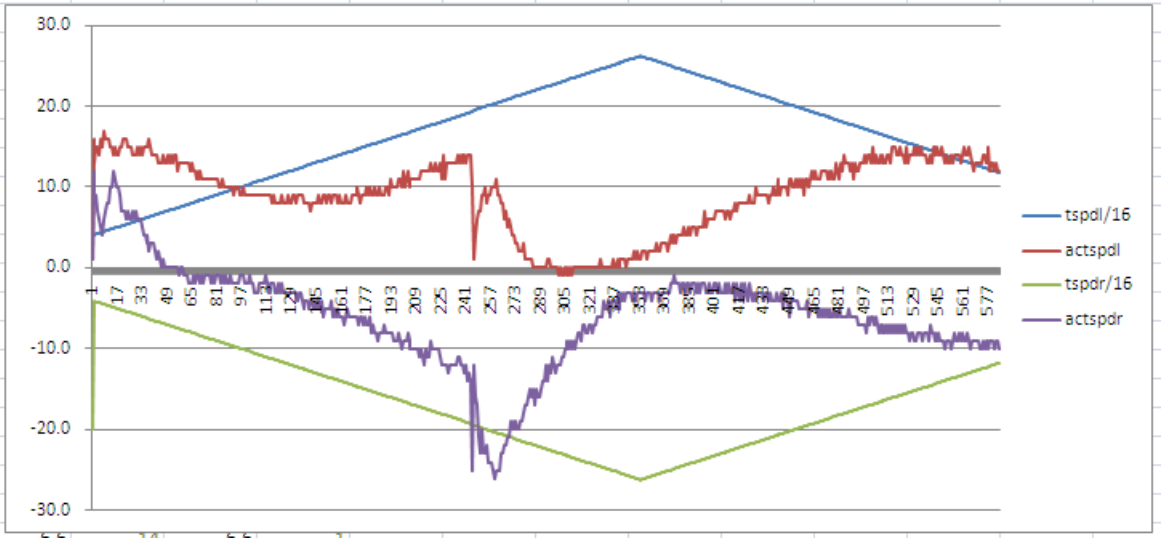


Step change on left wheel target speed (blue)
Actual speed follows (pale green)

The more fields you log & graph the better the chance of seeing what is happening

	dataset	gyrang	gyreq	gytestv	gyturnd	tspdl	tspdr	tspdl/16	actspdl	tspdr/16	actspdr
3	2	4567	3456	8023	0	65	-65	4.1	16	-4.1	12
4	3	4567	3456	8023	0	66	-66	4.1	13	-4.1	9
5	4	4574	3456	8023	0	67	-67	4.2	15	-4.2	9
6	5	4574	3456	8023	7	68	-68				
7	6	4574	3456	8023	7	69	-69				
8	7	4574	3456	8023	7	70	-70				
9	8	4574	3456	8023	7	71	-71				
10	9	4581	3456	8023	7	72	-72				
11	10	4581	3456	8023	14	73	-73				
12	11	4581	3456	8023	14	74	-74				
13	12	4581	3456	8023	14	75	-75				
14	13	4588	3456	8023	14	76	-76				
15	14	4588	3456	8023	21	77	-77				
16	15	4588	3456	8023	21	78	-78				
17	16	4588	3456	8023	21	79	-79				
18	17	4588	3456	8023	21	80	-80				
19	18	4595	3456	8023	21	81	-81				
20	19	4595	3456	8023	28	82	-82				
21	20	4595	3456	8023	28	83	-83				
22	21	4595	3456	8023	28	84	-84				
23	22	4601	3456	8023	28	85	-85				
24	23	4601	3456	8023	34	86	-86				
25	24	4601	3456	8023	34	87	-87				
26	25	4601	3456	8023	34	88	-88				
27	26	4601	3456	8023	34	89	-89				
28	27	4608	3456	8023	34	90	-90				
29	28	4608	3456	8023	41	91	-91				
30	29	4608	3456	8023	41	92	-92				
31	30	4608	3456	8023	41	93	-93				
32	31	4614	3456	8023	41	94	-94				
33	32	4614	3456	8023	47	95	-95				
34	33	4614	3456	8023	47	96	-96				
35	34	4614	3456	8023	47	97	-97				
36	35	4614	3456	8023	47	98	-98				
37	36	4620	3456	8023	47	99	-99				
38	37	4620	3456	8023	53	100	-100				
39	38	4620	3456	8023	53	101	-101				
40	39	4620	3456	8023	53	102	-102				
41	40	4626	3456	8023	53	103	-103				
42	41	4626	3456	8023	59	104	-104				
43	42	4626	3456	8023	59	105	-105				





End of main presentation

Questions ?

For reference:
dsPIC assembler logging and data transfer code examples follow

Code – every msec in loop

```
        ;Code called every millisecond at start of loop
        call        psechk1         ; check if pause button on display board is hit (RC8)
        ; only log data if in state 5 – doing a 180 degree turn
        mov         #0x0005, w0
        cp          state
        bra         Z, psd1
        goto        psret

psd1:   mov         prevdisl, w0    ;
        call        datalog         ; log data to area in RAM
        mov         actdisl, w0     ;
        call        datalog         ; log data to area in RAM
        mov         POS2CNTL, w0    ;
        call        datalog         ; log data to area in RAM
        mov         hactdisl, w0    ;
        call        datalog         ; log data to area in RAM
Psret   return
```

Code – check for button hit

```
; Routine to check if the press button on the display board has been hit in run 1
; this should pause the motors so that the info on the display can be read.
psechk1:    btsc    PORTA, #10      ; check status of Port C8 (push button)
            return
            btsc    sim,#0          ; don't stop if in simulation
            return
            mov     #0x0000, w0     ; set motor duty cycle value to stop
            mov     w0, dutycl      ; save for left motor
            mov     w0, dutycr      ; save for right motor
            call    mdriveb         ; update duty cycles
            ; don't do anything with data until main board push button hit
psewt1:     btsc    PORTB, #10      ; skip if push button pressed
            goto    psewt1          ; otherwise wait for it
;           Uncomment relevant line below for number of data items in set
;           mov     #0x0009, w0     ; group log data into 9 fields per line
;           mov     #0x0004, w0     ; group log data into 4 fields per line
;           mov     #0x0006, w0     ; group log data into 6 fields per line
            mov     #0x000F, w0     ; group log data into 15 fields per line
            mov     w0, noinset
            call    sendlog
            call    logdisp
```

Code – Data logging to RAM

```
; logs data in W0 into an area of RAM and moves pointer to next word
datalog:    btss    logena, #0      ; see if data logging is enabled
            return                  ; return if logging not enabled
            mov     w1, w1save      ; save W1 so not destroyed by routine
            mov     w0, w0save      ; save W0 so not destroyed by routine
            mov     logptr, w1      ; put next logging location into W1
            mov     w0, [w1]        ; put data into log area current position
            inc2    w1, w1          ; point w1 at next log location
            mov     w1, logptr      ; save current pointer location
            ; test for stop logging or wrap around
            mov     #dlogend, w0    ; get address of end of data logging area
            cp      w1, w0          ; check for end of log area reached
            bra     Z, dlogde       ; go to action for data space end
dlogret:    mov     w1save, w1      ; restore W1 register
            mov     w0save, w0      ; restore W0 register
            return
            ; action to do when reached end of data log area
dlogde:     nop
      ;     mov     #dlogarea, w0   ; get address of start of log area
      ;     mov     w0, logptr      ; save it in field - i.e. wrap round to start
            bclr    logena, #0      ; switch off logging
            goto    dlogret         ; restore w0, w1 and return
```

Code – Send log part 1

```
; sends logged data across the UART converted to text format as a CSV file
; Sets of data separated by cr/lfs - no in group determined by input parameter "noinset"
; Sends items of data from start of dlogarea up to item before current logptr, in blocks of 500
sendlog:    call    uartset             ; Switch off SPI & chip selects. Set up UART parameters
            call    wt1ms               ; wait for duration of 1 bit transmit before sending data
            mov     #dlogarea, w0       ; get address of start of log area
            mov     w0, dlogcur         ; save current location pointer
            mov     #0x01F4, w1         ; reset count of max sets to send to 500
            mov     w1, nosent          ; save it in nosent
            clr     nosets              ; counter of no of sets of data being sent
            call    sndsetn             ; send the set number bas the first item in each set
sndlp1:     mov     noinset, w2         ; w2 used as a counter for no of items in a set
sndlp2:     mov     dlogcur, w0         ; load current pointer
            cp      logptr              ; are we at the end of the logged data?
            bra     Z, sndret           ; branch out if we are
            mov     [w0], w1            ; get data item into w1
            call    sndhex              ; send hex number that is in w1
            ; word now sent
            inc2    dlogcur             ; point to next datalog item
            ; send a comma to separate the words
            call    sndcomma            ; send a comma text character
            dec     w2, w2              ; reduce set items counter
            bra     Z, sentset
            goto    sndlp2              ; process next item in set
            return
```

Code – Send log part 2

```
                ; send a carriage return / line feed to indicate end of record
sentset:        call    sndcrlf                 ; send carriage return / line feed text chars
                call    sndsetn                 ; send the set number
                goto    sndlp1                  ; go round loop again for next set of data to be sent
sndret:         return
sndcomma:       mov     #0x002C, w0     ; comma ASCII character
                call    sndchar                 ; transmit the character in low byte of w0
                return
sndcrlf:        mov     #0x000D, w0     ; carriage return ASCII character
                call    sndchar                 ; transmit the character in low byte of w0
                mov     #0x000A, w0     ; line feed return ASCII character
                call    sndchar                 ; transmit the character in low byte of w0

                return
sndsetn:        inc     nosets                  ; increment the set number
                dec     nosent                  ; decrement counter of sets to send in one go
                bra     Z, sndwt                ; pause after sending maximum of 500 sets in one go
sndrst:         mov     nosets, w1
                call    sndhex                  ; and send it
                call    sndcomma                ; followed by a comma
sndwt:          btsc    PORTB, #10      ; skip if push button pressed
                goto    sndwt                   ; otherwise wait for it
                mov     #0x01F4, w1     ; reset count of max sets to send to 500
                mov     w1, nosent
                goto    sndrst                  ; send the next group of up to 500 sets
```

Code – Send log part 3

```
                ; routine to send a 4 hex digit number as 5 decimal digits to the UART
sndhex:         clr     digit1          ; clear 1's decimal digit
                clr     digit2          ; clear 10's decimal digit
                clr     digit3          ; clear 100's decimal digit
                clr     digit4          ; clear 1000's decimal digit
                clr     digit5          ; clear 10000's decimal digit
                ; convert hex word into 5 decimal text characters
                btsc    w1, #0          ; check for bit 0
                call    sndad1          ; add 1 to the decimal values
                btsc    w1, #1          ; check for bit 1
                call    sndad2          ; add 2 to the decimal values
                btsc    w1, #2          ; check for bit 2
                call    sndad4          ; add 4 to the decimal values
                btsc    w1, #3          ; check for bit 3
                call    sndad8          ; add 8 to the decimal values
                btsc    w1, #4          ; check for bit 4
                call    sndad16         ; add 16 to the decimal values
                btsc    w1, #5          ; check for bit 5
                call    sndad32         ; add 32 to the decimal values
                btsc    w1, #6          ; check for bit 6
                call    sndad64         ; add 64 to the decimal values
                btsc    w1, #7          ; check for bit 7
                call    sndad128        ; add 128 to the decimal values
                btsc    w1, #8          ; check for bit 8
```

Code – Send log part 4

```
                call    sndad1b         ; add 256 to the decimal values
                btsc    w1, #9          ; check for bit 9
                call    sndad2b         ; add 512 to the decimal values
                btsc    w1, #10         ; check for bit 10
                call    sndad4b         ; add 1024 to the decimal values
                btsc    w1, #11         ; check for bit 11
                call    sndad8b         ; add 2048 to the decimal values
                btsc    w1, #12         ; check for bit 12
                call    sndad16b        ; add 4096 to the decimal values
                btsc    w1, #13         ; check for bit 13
                call    sndad32b        ; add 8192 to the decimal values
                btsc    w1, #14         ; check for bit 14
                call    sndad64b        ; add 16384 to the decimal values
                btsc    w1, #15         ; check for bit 15
                call    sndad128b       ; add 32768 to the decimal values
;               call    sndminus        ; convert from 2's complement minus no
                ; send 5 decimal digits to the UART to transmit 1 converted word
                mov     digit5, w3      ; get 5th decimal digit
                call    sndigit         ; send it to the UART
                mov     digit4, w3      ; get 4th decimal digit
                call    sndigit         ; send it to the UART
                mov     digit3, w3      ; get 3rd decimal digit
                call    sndigit         ; send it to the UART
                mov     digit2, w3      ; get 2nd decimal digit
                call    sndigit         ; send it to the UART
                mov     digit1, w3      ; get 1st decimal digit
                call    sndigit         ; send it to the UART
                return
```

Code – Send log part 5

```
; send a decimal digit (0 to 9) in bottom byte of w3 as an ascii character to the UART
sndigit:    mov     w3, w0          ; save w3 into sndchar parameter
            mov     #0x0030, w3     ; put ascii number offset in w3
            add     w3, w0, w0      ; add it to the value in w0
            goto    sndchar
; sends the character in the low bte of w0 to the UART
sndchar:    mov     #U1TXREGL, w1
            mov.b   w0, [w1]        ; put low byte of data into transmit register
sndtxa:     btsc    U1STA, #9       ; test for transmit buffer full
            goto    sndtxa          ; if full, wait
            bclr    U1STA, #15      ; reset interrupt bits
            bclr    U1STA, #13      ; reset interrupt bits
            return
; add numbers for hex bit positions to the 5 decimal digits
sndad1:     mov     #0x0001, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad2:     mov     #0x0002, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad4:     mov     #0x0004, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad8:     mov     #0x0008, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
```

Code – Send log part 6

```
sndad16:    mov     #0x0006, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0001, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad32:    mov     #0x0002, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0003, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad64:    mov     #0x0004, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0006, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad128:   mov     #0x0008, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0002, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            mov     #0x0001, w0     ; get value to add in wreg
            add     digit3                  ; add to 100's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
```

Code – Send log part 7

```
sndad1b:    mov     #0x0006, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0005, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            mov     #0x0002, w0     ; get value to add in wreg
            add     digit3                  ; add to 100's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad2b:    mov     #0x0002, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0001, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            mov     #0x0005, w0     ; get value to add in wreg
            add     digit3                  ; add to 100's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
sndad4b:    mov     #0x0004, w0     ; get value to add in wreg
            add     digit1                  ; add to 1's digit
            mov     #0x0002, w0     ; get value to add in wreg
            add     digit2                  ; add to 10's digit
            mov     #0x0000, w0     ; get value to add in wreg
            add     digit3                  ; add to 100's digit
            mov     #0x0001, w0     ; get value to add in wreg
            add     digit4                  ; add to 1000's digit
            goto    sndoflo                 ; check for overflow and propogate carry if reqd
```

Code – Send log part 8

```
sndad8b:        mov             #0x0008, w0     ; get value to add in wreg
                add             digit1                  ; add to 1's digit
                mov             #0x0004, w0     ; get value to add in wreg
                add             digit2                  ; add to 10's digit
                mov             #0x0000, w0     ; get value to add in wreg
                add             digit3                  ; add to 100's digit
                mov             #0x0002, w0     ; get value to add in wreg
                add             digit4                  ; add to 1000's digit
                goto            sndoflo                 ; check for overflow and propogate carry if reqd
sndad16b:       mov             #0x0006, w0     ; get value to add in wreg
                add             digit1                  ; add to 1's digit
                mov             #0x0009, w0     ; get value to add in wreg
                add             digit2                  ; add to 10's digit
                mov             #0x0000, w0     ; get value to add in wreg
                add             digit3                  ; add to 100's digit
                mov             #0x0004, w0     ; get value to add in wreg
                add             digit4                  ; add to 1000's digit
                goto            sndoflo                 ; check for overflow and propogate carry if reqd
sndad32b:mov            #0x0002, w0     ; get value to add in wreg
                add             digit1                  ; add to 1's digit
                mov             #0x0009, w0     ; get value to add in wreg
                add             digit2                  ; add to 10's digit
                mov             #0x0001, w0     ; get value to add in wreg
                add             digit3                  ; add to 100's digit
                mov             #0x0008, w0     ; get value to add in wreg
                add             digit4                  ; add to 1000's digit
                goto            sndoflo                 ; check for overflow and propogate carry if reqd
```

Code – Send log part 9

```
sndad64b:       mov                     #0x0004, w0     ; get value to add in wreg
                add                     digit1                  ; add to 1's digit
                mov                     #0x0008, w0     ; get value to add in wreg
                add                     digit2                  ; add to 10's digit
                mov                     #0x0003, w0     ; get value to add in wreg
                add                     digit3                  ; add to 100's digit
                mov                     #0x0006, w0     ; get value to add in wreg
                add                     digit4                  ; add to 1000's digit
                mov                     #0x0001, w0     ; get value to add in wreg
                add                     digit5                  ; add to 10000's digit
                goto                    sndoflo                 ; check for overflow and propogate carry if reqd
sndad128b:mov           #0x0008, w0     ; get value to add in wreg
                add                     digit1                  ; add to 1's digit
                mov                     #0x0006, w0     ; get value to add in wreg
                add                     digit2                  ; add to 10's digit
                mov                     #0x0007, w0     ; get value to add in wreg
                add                     digit3                  ; add to 100's digit
                mov                     #0x0002, w0     ; get value to add in wreg
                add                     digit4                  ; add to 1000's digit
                mov                     #0x0003, w0     ; get value to add in wreg
                add                     digit5                  ; add to 10000's digit
                goto                    sndoflo                 ; check for overflow and propogate carry if reqd
                return
sndminus:       nop
                return
```

Code – Send log part 10

```
; test for overflow above nine in any digit and carry it forward to next digit
sndoflo:    mov     #0x000A, w0     ; lowest overflow value
sndof1:     cp      digit1                  ; check 1st digit for overflow
            bra     LT, sndof2      ; if 9 or less skip next instructions
            inc     digit2                  ; carry 1 to next digit
            sub     digit1                  ; subract 10 from digit1
            goto    sndof1                  ; check if still above 9
sndof2:     cp      digit2                  ; check 2nd digit for overflow
            bra     LT, sndof3      ; if 9 or less skip next instructions
            inc     digit3                  ; carry 1 to next digit
            sub     digit2                  ; subract 10 from digit2
            goto    sndof2                  ; check if still above 9
sndof3:     cp      digit3                  ; check 3rd digit for overflow
            bra     LT, sndof4      ; if 9 or less skip next instructions
            inc     digit4                  ; carry 1 to next digit
            sub     digit3                  ; subract 10 from digit3
            goto    sndof3                  ; check if still above 9
sndof4:     cp      digit4                  ; check 4th digit for overflow
            bra     LT, sndof5      ; if 9 or less skip next instructions
            inc     digit5                  ; carry 1 to next digit
            sub     digit4                  ; subract 10 from digit4
            goto    sndof4                  ; check if still above 9
sndof5:     mov     #0x0006, w0     ; highest valid value for 5th digit (65535)
            cp      digit5                  ; check 5th digit for overflow
            bra     LE, sndor               ; if 6 or less return
            goto    snderr
sndor: return                               ; return with overflows and carry handled
snderr:     goto    snderr                  ; error loop
```

Code - UART set up 1

```
; Set up UART for communication with PC via FTDI USB TTL232R device
uartset:        bset                    LATA, #9        ; switch off the gyro SPI chip select
                bset                    LATA, #7        ; switch off the Nokia SPI chip select
                bclr                    SPI1STAT, #15   ; disable SPI1 module to switch it off
                ; Assign UART recieve data to RP0
                mov                     #0x0, w0        ; Receive data in assigned to RP0 - pin 21
                mov                     #RPINR18L, w1   ; goes in low byte of RPINR18 - U1RX
                mov.b           w0, [w1]
                ; Assign UART CTS to RP25
                mov                     #0x19, w0       ; CTS assigned to RP25 - pin 5
                mov                     #RPINR18H, w1   ; goes in high byte of RPINR18 - U1CTS
                mov.b           w0, [w1]
                ; set UART output pin reassignments for U1RTS to RP24 - pin 4
                mov                     #0x04, w0       ; UART RTS assigned to RP24
                mov                     #RPOR12L, w1    ; goes in low byte of RPOR12 - U1RTS
                mov.b           w0, [w1]
                ; set UART output pin reassignments for U1TX to RP1 - pin 22
                mov                     #0x03, w0       ; UART Transmit data assigned to RP1
                mov                     #RPOR0H, w1     ; goes in high byte of RPOR0 - U1TX
                mov.b           w0, [w1]

                                        return
```

Code – UART set up 2

```
; set UART parameters for communication with FTDI232
clr                 U1MODE              ; default set all bits to zero
bset                U1MODE, #13         ; discontinue module when it enters idle mode
bclr                U1MODE, #11         ; RTS pin in flow control mode
bset                U1MODE, #9          ; TX, RX, CTS & RTS pins enabled & used
bclr                U1MODE, #8          ; TX, RX, CTS & RTS pins enabled & used
bset                U1MODE, #4          ; RX idle state is 0
bclr                U1MODE, #3          ; Baud rate is Low
bclr                U1MODE, #2          ; 8 bit data no parity
bclr                U1MODE, #1          ; 8 bit data no parity
bclr                U1MODE, #0          ; One stop bit

clr                 U1STA               ; default set all bits to zero
bclr                U1STA, #14          ; TX idle state is 0
bclr                U1STA, #13          ; interrupt when bit sent
bset                U1STA, #7           ; interrupt set when receive buffer is full (4 chars)
bset                U1STA, #6           ; interrupt set when receive buffer is full (4 chars)
; set baud rate
mov                 #0x0081, w0         ; set baud rate to 19200 (baud rate is low) - value 129
mov                 w0, U1BRG
;                   ; enable SPI1 module now parameters are set
bset                U1MODE, #15         ; enable UART1 module to switch it on
bset                U1STA, #10          ; transmit enabled (must be AFTER module enable)
bset                disorgyr, #2        ; set flag to say SPI pins being used by UART
```