

```

1  {{
2  -----
3  File: Parallax Serial Terminal.spin
4  Version: 1.0
5  Copyright (c) 2009 Parallax, Inc.
6  See end of file for terms of use.
7
8  Authors: Jeff Martin, Andy Lindsay, Chip Gracey
9  -----
10 }}
11
12 {
13 HISTORY:
14   This object is made for direct use with the Parallax Serial Terminal; a simple serial communication program
15   available with the Propeller Tool installer and also separately via the Parallax website (www.parallax.com).
16
17   This object is heavily based on FullDuplexSerialPlus (by Andy Lindsay), which is itself heavily based on
18   FullDuplexSerial (by Chip Gracey).
19
20 USAGE:
21   • Call Start, or StartRxTx, first.
22   • Be sure to set the Parallax Serial Terminal software to the baudrate specified in Start, and the proper COM port.
23   • At 80 MHz, this object properly receives/transmits at up to 250 Kbaud, or performs transmit-only at up to 1 Mbaud.
24
25 }
26
27 CON
28 ..
29   Parallax Serial Terminal
30   Control Character Constants
31   ..
32   CS = 16  ``CS: Clear Screen
33   CE = 11  ``CE: Clear to End of line
34   CB = 12  ``CB: Clear lines Below
35
36   HM = 1   ``HM: HoMe cursor
37   PC = 2   ``PC: Position Cursor in x,y
38   PX = 14  ``PX: Position cursor in X
39   PY = 15  ``PY: Position cursor in Y
40
41   NL = 13  ``NL: New Line
42   LF = 10  ``LF: Line Feed
43   ML = 3   ``ML: Move cursor Left
44   MR = 4   ``MR: Move cursor Right
45   MU = 5   ``MU: Move cursor Up
46   MD = 6   ``MD: Move cursor Down
47   TB = 9   ``TB: TaB
48   BS = 8   ``BS: BackSpace
49
50   BP = 7   ``BP: BeeP speaker
51
52 CON
53
54   BUFFER_LENGTH = 64           'Recommended as 64 or higher, but can be 2, 4, 8, 16, 32, 64, 128
55   or 256.
56   BUFFER_MASK   = BUFFER_LENGTH - 1
57   MAXSTR_LENGTH = 49           'Maximum length of received numerical string (not including zero
58   terminator).
59
60 VAR
61
62   long cog           'Cog flag/id
63
64   long rx_head       '9 contiguous longs (must keep order)
65   long rx_tail
66   long tx_head

```

```

82      Baud Rate field.
83  Returns   : True (non-zero) if cog started, or False (0) if no cog is available.}}
84
85  okay := StartRxTx(31, 30, 0, baudrate)
86  waitcnt(clkfreq + cnt)           'Wait 1 second for PST
87  Clear                             'Clear display
88
89  PUB StartRxTx(rxpın, txpin, mode, baudrate) : okay
90  {{Start serial communication with designated pins, mode, and baud.
91  Parameters:
92    rxpin   - input pin; receives signals from external device's TX pin.
93    txpin   - output pin; sends signals to external device's RX pin.
94    mode    - signaling mode (4-bit pattern).
95              bit 0 - inverts rx.
96              bit 1 - inverts tx.
97              bit 2 - open drain/source tx.
98              bit 3 - ignore tx echo on rx.
99    baudrate - bits per second.
100 Returns   : True (non-zero) if cog started, or False (0) if no cog is available.}}
101
102  stop
103  longfill(@rx_head, 0, 4)
104  longmove(@rx_pin, @rxpin, 3)
105  bit_ticks := clkfreq / baudrate
106  buffer_ptr := @rx_buffer
107  okay := cog := cognew(@entry, @rx_head) + 1
108
109  PUB Stop
110  {{Stop serial communication; frees a cog.}}
111
112  if cog
113    cogstop(cog~ - 1)
114    longfill(@rx_head, 0, 9)
115
116  PUB Char(bytechr)
117  {{Send single-byte character.  Waits for room in transmit buffer if necessary.
118  Parameter:
119    bytechr - character (ASCII byte value) to send.}}
120
121  repeat until (tx_tail <> ((tx_head + 1) & BUFFER_MASK))
122    tx_buffer[tx_head] := bytechr
123    tx_head := (tx_head + 1) & BUFFER_MASK
124
125  if rxtx_mode & %1000
126    CharIn
127
128  PUB Chars(bytechr, count)
129  {{Send multiple copies of a single-byte character.  Waits for room in transmit buffer if necessary.
130  Parameters:
131    bytechr - character (ASCII byte value) to send.
132    count   - number of bytechrs to send.}}
133
134  repeat count
135    Char(bytechr)
136
137  PUB CharIn : bytechr
138  {{Receive single-byte character.  Waits until character received.
139  Returns: $00..$FF}}
140
141  repeat while (bytechr := RxCheck) < 0
142
143  PUB Str(stringptr)
144  {{Send zero terminated string.
145  Parameter:
146    stringptr - pointer to zero terminated string to send.}}
147

```

```

165         Memory reserved must be large enough for all string characters plus a zero terminator (maxcount + 1).
166         maxcount - maximum length of string to receive, or -1 for unlimited.}}
167
168     repeat while (maxcount--)'While maxcount not reached
169         if (byte[stringptr++] := CharIn) == NL'Get chars until NL
170             quit
171     byte[stringptr+(byte[stringptr-1] == NL)]~'Zero terminate string; overwrite NL or
append 0 char
172
173 PUB Dec(value) | i, x
174 {{Send value as decimal characters.
175     Parameter:
176         value - byte, word, or long value to send as decimal characters.}}
177
178     x := value == NEGX'Check for max negative
179     if value < 0
180         value := |(value+x)'If negative, make positive; adjust for
max negative
181     Char("-")'and output sign
182
183     i := 1_000_000_000'Initialize divisor
184
185     repeat 10'Loop for 10 digits
186         if value => i
187             Char(value / i + "0" + x*(i == 1))'If non-zero digit, output digit; adjust
for max negative
188             value /= i'and digit from value
189             result~~'flag non-zero found
190         elseif result or i == 1
191             Char("0")'If zero digit (or only digit) output it
192             i /= 10'Update divisor
193
194 PUB DecIn : value
195 {{Receive carriage return terminated string of characters representing a decimal value.
196     Returns: the corresponding decimal value.}}
197
198     StrInMax(@str_buffer, MAXSTR_LENGTH)
199     value := StrToBase(@str_buffer, 10)
200
201 PUB Bin(value, digits)
202 {{Send value as binary characters up to digits in length.
203     Parameters:
204         value - byte, word, or long value to send as binary characters.
205         digits - number of binary digits to send. Will be zero padded if necessary.}}
206
207     value <= 32 - digits
208     repeat digits
209         Char((value <= 1) & 1 + "0")
210
211 PUB BinIn : value
212 {{Receive carriage return terminated string of characters representing a binary value.
213     Returns: the corresponding binary value.}}
214
215     StrInMax(@str_buffer, MAXSTR_LENGTH)
216     value := StrToBase(@str_buffer, 2)
217
218 PUB Hex(value, digits)
219 {{Send value as hexadecimal characters up to digits in length.
220     Parameters:
221         value - byte, word, or long value to send as hexadecimal characters.
222         digits - number of hexadecimal digits to send. Will be zero padded if necessary.}}
223
224     value <= (8 - digits) << 2
225     repeat digits
226         Char(lookupz((value <= 4) & $F : "0"..9", "A"..F"))
227

```

```
245 PUB ClearBelow
246 {{Clear all lines below cursor.}}
247
248 Char(CB)
249
250 PUB Home
251 {{Send cursor to home position (top-left).}}
252
253 Char(HM)
254
255 PUB Position(x, y)
256 {{Position cursor at column x, row y (from top-left).}}
257
258 Char(PC)
259 Char(x)
260 Char(y)
261
262 PUB PositionX(x)
263 {{Position cursor at column x of current row.}}
264 Char(PX)
265 Char(x)
266
267 PUB PositionY(y)
268 {{Position cursor at row y of current column.}}
269 Char(PY)
270 Char(y)
271
272 PUB NewLine
273 {{Send cursor to new line (carriage return plus line feed).}}
274
275 Char(NL)
276
277 PUB LineFeed
278 {{Send cursor down to next line.}}
279
280 Char(LF)
281
282 PUB MoveLeft(x)
283 {{Move cursor left x characters.}}
284
285 repeat x
286 Char(ML)
287
288 PUB MoveRight(x)
289 {{Move cursor right x characters.}}
290
291 repeat x
292 Char(MR)
293
294 PUB MoveUp(y)
295 {{Move cursor up y lines.}}
296
297 repeat y
298 Char(MU)
299
300 PUB MoveDown(y)
301 {{Move cursor down y lines.}}
302
303 repeat y
304 Char(MD)
305
306 PUB Tab
307 {{Send cursor to next tab position.}}
308
309 Char(TB)
310
```

```

328 PUB RxFlush
329 {{Flush receive buffer.}}
330
331     repeat while rxcheck => 0
332
333 PRI RxCheck : bytechr
334 {Check if character received; return immediately.
335  Returns: -1 if no byte received, $00..$FF if character received.}
336
337     bytechr~~
338     if rx_tail <> rx_head
339         bytechr := rx_buffer[rx_tail]
340         rx_tail := (rx_tail + 1) & BUFFER_MASK
341
342 PRI StrToBase(stringptr, base) : value | chr, index
343 {Converts a zero terminated string representation of a number to a value in the designated base.
344  Ignores all non-digit characters (except negative (-) when base is decimal (10)).}
345
346     value := index := 0
347     repeat until ((chr := byte[stringptr][index++]) == 0)
348         chr := -15 + --chr & %11011111 + 39*(chr > 56) 'Make "0"- "9", "A"- "F", "a"- "f" be 0 - 15,
others out of range
349         if (chr > -1) and (chr < base) 'Accumulate valid values into result;
ignore others
350             value := value * base + chr
351             if (base == 10) and (byte[stringptr] == "-") 'If decimal, address negative sign; ignore
otherwise
352                 value := - value
353
354 DAT
355
356 *****
357 * Assembly language serial driver *
358 *****
359
360     org
361
362
363     Entry
364
365 entry                mov     t1,par                'get structure address
366                     add     t1,#4 << 2            'skip past heads and tails
367
368                     rdlong   t2,t1                'get rx_pin
369                     mov     rxmask,#1
370                     shl     rxmask,t2
371
372                     add     t1,#4                'get tx_pin
373                     rdlong   t2,t1
374                     mov     txmask,#1
375                     shl     txmask,t2
376
377                     add     t1,#4                'get rxtx_mode
378                     rdlong   rxtxmode,t1
379
380                     add     t1,#4                'get bit_ticks
381                     rdlong   bitticks,t1
382
383                     add     t1,#4                'get buffer_ptr
384                     rdlong   rxbuff,t1
385                     mov     txbuff,rxbuff
386                     add     txbuff,#BUFFER_LENGTH
387
388                     test     rxtxmode,%100 wz      'init tx pin according to mode
389                     test     rxtxmode,%010 wc
390                     if_z_ne_c or     outa,txmask

```

```

408
409 :bit          add    rxcnt,bitticks    'ready next bit period
410
411 :wait          jmpret  rxcode,txcode    'run chunk of tx code, then return
412
413              mov     t1,rxcnt          'check if bit receive period done
414              sub     t1,cnt
415              cmps    t1,#0             wc
416 if_nc          jmp     #:wait
417
418              test    rxmask,ina        wc    'receive bit on rx pin
419              rcr     rxdata,#1
420              djnz    rxbits,#:bit
421
422              shr     rxdata,#32-9      'justify and trim received byte
423              and     rxdata,#$FF
424              test    rctxmode,#%001    wz    'if rx inverted, invert byte
425 if_nz          xor     rxdata,$$FF
426
427              rdlong  t2,par            'save received byte and inc head
428              add     t2,rxbuff
429              wrbyte  rxdata,t2
430              sub     t2,rxbuff
431              add     t2,#1
432              and     t2,$BUFFER_MASK
433              wrlong  t2,par
434
435              jmp     #receive          'byte done, receive next byte
436
437
438 ' Transmit
439
440 transmit       jmpret  txcode,rxcode    'run chunk of rx code, then return
441
442              mov     t1,par            'check for head <> tail
443              add     t1,#2 << 2
444              rdlong  t2,t1
445              add     t1,#1 << 2
446              rdlong  t3,t1
447              cmp     t2,t3             wz
448 if_z           jmp     #transmit
449
450              add     t3,txbuff          'get byte and inc tail
451              rdbyte  txdata,t3
452              sub     t3,txbuff
453              add     t3,#1
454              and     t3,$BUFFER_MASK
455              wrlong  t3,t1
456
457              or      txdata,$$100      'ready byte to transmit
458              shl     txdata,#2
459              or      txdata,#1
460              mov     txbits,#11
461              mov     txcnt,cnt
462
463 :bit           test    rctxmode,$%100    wz    'output bit on tx pin
464              test    rctxmode,$%010    wc    'according to mode
465 if_z_and_c     xor     txdata,#1
466              shr     txdata,#1         wc
467 if_z           muxc    outa,txmask
468 if_nz          dira    txmask
469              add     txcnt,bitticks    'ready next cnt
470
471 :wait          jmpret  txcode,rxcode    'run chunk of rx code, then return
472
473              mov     t1,txcnt          'check if bit transmit period done

```

```
491
492 rxmask          res    1
493 rxbuff          res    1
494 rxdata          res    1
495 rxbits          res    1
496 rxcnt           res    1
497 rxcode          res    1
498
499 txmask          res    1
500 txbuff          res    1
501 txdata          res    1
502 txbits          res    1
503 txcnt           res    1
504 txcode          res    1
```

```
505
506 {{
```

```
507
508 |-----|
509 |                TERMS OF USE: MIT License                |
```

```
510 |-----|
511 |Permission is hereby granted, free of charge, to any person obtaining a copy of this
512 |software and associated documentation files (the "Software"), to deal in the Software
513 |without restriction, including without limitation the rights to use, copy, modify,
514 |merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
515 |permit persons to whom the Software is furnished to do so, subject to the following
516 |conditions:
```

```
517 |-----|
518 |
519 |The above copyright notice and this permission notice shall be included in all copies
520 |or substantial portions of the Software.
```

```
521 |-----|
522 |THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
523 |INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
524 |PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
525 |HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
526 |OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
527 |SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
528 |-----|
529 |}}
530
```

```
518 |The above copyright notice and this permission notice shall be included in all copies |
519 |or substantial portions of the Software. |
520 |
```

```
521 |THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, |
522 |INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A |
523 |PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT |
524 |HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION |
525 |OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE |
526 |SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. |
527 |-----|
528 }}
```