

```

'' *****
'' * Full-Duplex Serial Driver v1.1 *
'' * Author: Chip Gracey *
'' * Copyright (c) 2006 Parallax, Inc. *
'' * See end of file for terms of use. *
'' *****

```

VAR

```

long cog                'cog flag/id

long rx_head            '9 contiguous longs
long rx_tail
long tx_head
long tx_tail
long rx_pin
long tx_pin
long rxtx_mode
long bit_ticks
long buffer_ptr

byte rx_buffer[16]      'transmit and receive buffers
byte tx_buffer[16]

```

```

PUB start(rxpin, txpin, mode, baudrate) : okay

```

```

'' Start serial driver - starts a cog
'' returns false if no cog available
,,
'' mode bit 0 = invert rx
'' mode bit 1 = invert tx
'' mode bit 2 = open-drain/source tx
'' mode bit 3 = ignore tx echo on rx

stop
longfill(@rx_head, 0, 4)
longmove(@rx_pin, @rxpin, 3)
bit_ticks := clkfreq / baudrate
buffer_ptr := @rx_buffer
okay := cog := cognew(@entry, @rx_head) + 1

```

```

PUB stop

```

```

'' Stop serial driver - frees a cog

if cog
  cogstop(cog~ - 1)
  longfill(@rx_head, 0, 9)

```

```

PUB rxflush

```

```

'' Flush receive buffer

repeat while rxcheck => 0

```

```

PUB rxcheck : rxbyte

```

```

'' Check if byte received (never waits)

```

```
'' returns -1 if no byte received, $00..$FF if byte
```

```
rxbyte--
if rx_tail <> rx_head
    rxbyte := rx_buffer[rx_tail]
    rx_tail := (rx_tail + 1) & $F
```

```
PUB rxtime(ms) : rxbyte | t
```

```
'' Wait ms milliseconds for a byte to be received
'' returns -1 if no byte received, $00..$FF if byte
```

```
t := cnt
repeat until (rxbyte := rxcheck) => 0 or (cnt - t) / (clkfreq / 1000) > ms
```

```
PUB rx : rxbyte
```

```
'' Receive byte (may wait for byte)
'' returns $00..$FF
```

```
repeat while (rxbyte := rxcheck) < 0
```

```
PUB tx(txbyte)
```

```
'' Send byte (may wait for room in buffer)
```

```
repeat until (tx_tail <> (tx_head + 1) & $F)
tx_buffer[tx_head] := txbyte
tx_head := (tx_head + 1) & $F
```

```
if rxtx_mode & %1000
    rx
```

```
PUB str(stringptr)
```

```
'' Send string
```

```
repeat strsize(stringptr)
    tx(byte[stringptr++])
```

```
PUB dec(value) | i
```

```
'' Print a decimal number
```

```
if value < 0
    -value
    tx("-")
```

```
i := 1_000_000_000
```

```
repeat 10
    if value => i
        tx(value / i + "0")
        value //= i
        result~~
    elseif result or i == 1
        tx("0")
    i /= 10
```

PUB hex(value, digits)

'' Print a hexadecimal number

value <<= (8 - digits) << 2

repeat digits

tx(lookupz((value <-= 4) & \$F : "0".."9", "A".."F"))

PUB bin(value, digits)

'' Print a binary number

value <<= 32 - digits

repeat digits

tx((value <-= 1) & 1 + "0")

DAT

```

' *****
' * Assembly language serial driver *
' *****

```

```

,
,
,
' Entry
,
entry          mov     t1,par           'get structure address
               add     t1,#4 << 2      'skip past heads and tails

               rdlong  t2,t1           'get rx_pin
               mov     rxmask,#1
               shl     rxmask,t2

               add     t1,#4           'get tx_pin
               rdlong  t2,t1
               mov     txmask,#1
               shl     txmask,t2

               add     t1,#4           'get rxtx_mode
               rdlong  rxtxmode,t1

               add     t1,#4           'get bit_ticks
               rdlong  bitticks,t1

               add     t1,#4           'get buffer_ptr
               rdlong  rxbuff,t1
               mov     txbuff,rxbuff
               add     txbuff,#16

               test    rxtxmode,##100  wz      'init tx pin according to mode
               test    rxtxmode,##010  wc
               or      outa,txmask
               or      dira,txmask

               mov     txcode,#transmit 'initialize ping-pong multitasking
,
,
' Receive

```

```

,
receive                jmpret  rxcode,txcode          'run a chunk of transmit code, then return

                        test    rxtxmode,%001  wz          'wait for start bit on rx pin
                        test    rxmask,ina      wc
if_z_eq_c             jmp     #receive

                        mov     rxbits,#9        'ready to receive byte
                        mov     rxcnt,bitticks
                        shr     rxcnt,#1
                        add     rxcnt,cnt

:bit                  add     rxcnt,bitticks    'ready next bit period

:wait                 jmpret  rxcode,txcode    'run a chunk of transmit code, then return

                        mov     t1,rxcnt        'check if bit receive period done
                        sub     t1,cnt
                        cmps   t1,#0          wc
if_nc                jmp     #:wait

                        test    rxmask,ina      wc          'receive bit on rx pin
                        rcr     rxdata,#1
                        djnz   rxbits,#:bit

                        shr     rxdata,#32-9    'justify and trim received byte
                        and     rxdata,#$FF
if_nz                test    rxtxmode,%001  wz          'if rx inverted, invert byte
                        xor     rxdata,#$FF

                        rdlong  t2,par         'save received byte and inc head
                        add     t2,rxbuff
                        wrbyte  rxdata,t2
                        sub     t2,rxbuff
                        add     t2,#1
                        and     t2,$0F
                        wrlong  t2,par

                        jmp     #receive        'byte done, receive next byte

,
,
: Transmit
,
transmit              jmpret  txcode,rxcode    'run a chunk of receive code, then return

                        mov     t1,par         'check for head <> tail
                        add     t1,#2 << 2
                        rdlong  t2,t1
                        add     t1,#1 << 2
                        rdlong  t3,t1
if_z                 cmp     t2,t3          wz
                        jmp     #transmit

                        add     t3,txbuff      'get byte and inc tail
                        rdbyte  txdata,t3
                        sub     t3,txbuff
                        add     t3,#1
                        and     t3,$0F
                        wrlong  t3,t1

                        or     txdata,$100    'ready byte to transmit
                        shl    txdata,#2
                        or     txdata,#1

```

```

mov    txbits,#11
mov    txcnt,cnt

:bit    test    rxtxmode, #%100    wz    'output bit on tx pin according to mode
        test    rxtxmode, #%010    wc
        if_z_and_c  xor    txdata,#1
        shr     txdata,#1        wc
        if_z    muxc    outa,txmask
        if_nz   muxnc   dira,txmask
        add     txcnt,bitticks    'ready next cnt

:wait   jmpret   txcode,rxcode    'run a chunk of receive code, then return

        mov     t1,txcnt        'check if bit transmit period done
        sub     t1,cnt
        cmps    t1,#0          wc
        if_nc   jmp     #:wait

        djnz    txbits, #:bit    'another bit to transmit?

        jmp     #transmit        'byte done, transmit next byte

```

Uninitialized data

```

t1      res    1
t2      res    1
t3      res    1

rxtxmode  res    1
bitticks  res    1

rxmask    res    1
rxbuff    res    1
rxdata    res    1
rxbits    res    1
rxcnt     res    1
rxcode    res    1

txmask    res    1
txbuff    res    1
txdata    res    1
txbits    res    1
txcnt     res    1
txcode    res    1

```

{{

TERMS OF USE: MIT License

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

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

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

}}