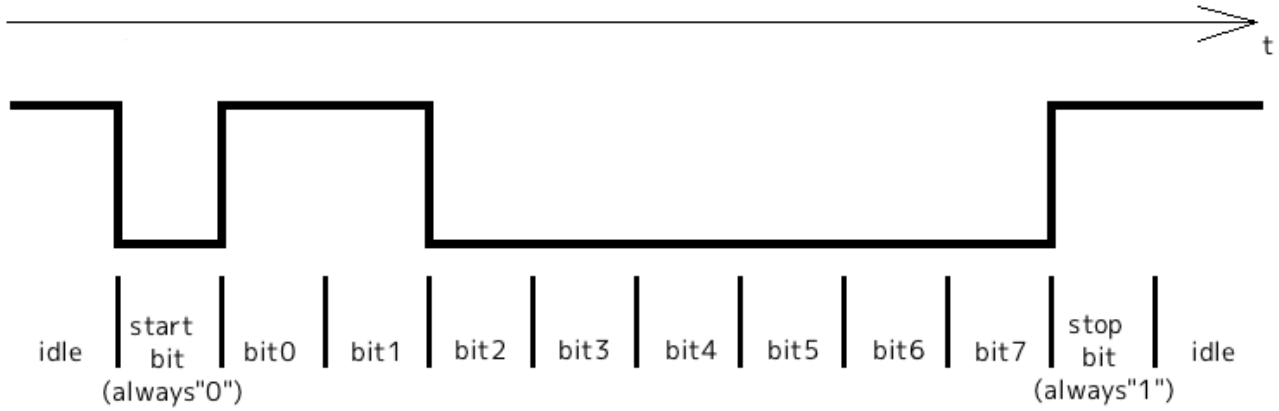
Serial Communication

~~20150408~~ ~~20151103~~ ~~20160108~~ 20160109

HalfDuplex Serial Communication



[Startbit:1bit NoParity Stopbit:1bit]

When HalfDuplex-serialcommunication is used on PropForth, it make structure below;

```
{
half duplex serial structure
00 - 03 -- bitticks
04 - 07 -- rx pin mask
08 - 0B -- tx pin mask
}
\ hdserialStruct X ( baud rxpin txpin -- )   X is structure's name
: hdserialStruct
        lockdict variable 8 allot lastnfa nfa>pfa 2+ alignl freedict
        tuck swap >m swap 8 + L!
        tuck swap >m swap 4 + L!
        swap clkfreq swap u/ swap L!
;

0 wconstant Rx
1 wconstant Tx
d19200 constant baud

baud Rx Tx hdserialStruct hd_serial
```

Defined as {Rx is P0,Tx is P1 and baudrate is 19200bit/sec} by
{baud Rx Tx hdserialStruct serial}.
Name'hd_serial' indicate top address of halfduplex serial structure.

Transmit assembler code

```
( n1 n2 -- )
entry  n1:sending data n2:hdserialStruct's name

Using register inside
      $C_treg2:loop counter
      $C_treg3:ticks(1bit)

exit   none

fl
build_BootOpt :rasm
        rdlong    __bitticks , $C_stTOS
        add       $C_stTOS , # 8
        rdlong    __txmask , $C_stTOS
        spop

        or        $C_stTOS , __stopbit
        shl       $C_stTOS , # 1

        mov       $C_treg2 , # d11
        mov       $C_treg3 , __bitticks
        add       $C_treg3 , cnt

__txloop
        test      $C_stTOS , # 1 wz
        muxz      dira , __txmask
        waitcnt   $C_treg3 , __bitticks
        shr       $C_stTOS , # 1
        djnz      $C_treg2 , # __txloop
        spop

        andn      dira , __txmask
        jexit

__bitticks
    0
__txmask
    0
__stopbit
    h300

;asm a_hdserialTx
```

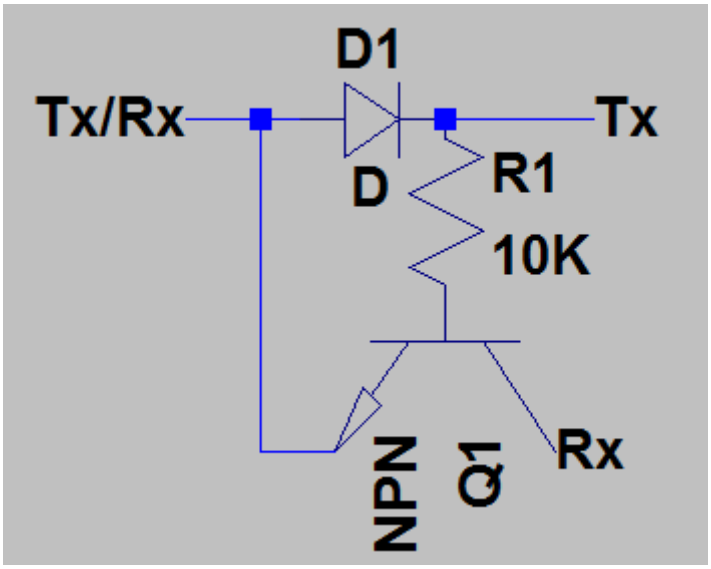
Receive assembler code

```
( n1 -- n2 )
entry  n1:hdserialStruct's name

Using register inside
      $C_treg1:loop counter
      $C_treg2:next bit count(1st:1.25bit, 2nd-8th:1bit)

exit   n2:receive data

fl
build_BootOpt :rasm
        rdlong    __bitticks , $C_stTOS
        add       $C_stTOS , # 4
        rdlong    __rxmask , $C_stTOS

        mov       $C_treg1 , # 8
        mov       $C_stTOS , # 0
        mov       $C_treg2 , __bitticks
        shr       $C_treg2 , # 2
        add       $C_treg2 , __bitticks

        \ Wait from hi to lo transition
        waitpeq   __rxmask , __rxmask
        waitpne   __rxmask , __rxmask

        \ first loop tick count (1.25bit+cnt)
        add       $C_treg2 , cnt
__rxloop
        \ 1bit+cnt
        waitcnt   $C_treg2 , __bitticks
        test      __rxmask , ina wz
        shr       $C_stTOS , # 1
        muxnz     $C_stTOS , # h80
        djnz      $C_treg1 , # __rxloop

        jexit

__bitticks
    0
__rxmask
    0

;asm a_hdserialRx
```

How to use;
Transmit; data hd_serial a_hdserialTx
　　Data on stack is transmitted to TX-pin.
Receive; hd_serial a_hdserialRx
　　Receiving data is on stack.

Sample;
ThermalPrinter_0.4.f

Above code use 2pin for Tx and Rx.
Tried to communicate by using only 1pin.
I refered below;
http://nerdralph.blogspot.ca/2014/01/avr-half-duplex-software-uart.html



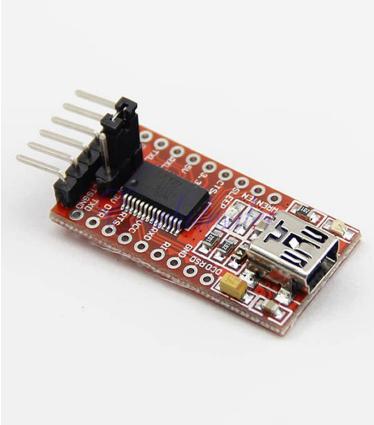
Connected P1 to Tx/Rx.
Conected Tx to printer Tx and Rx to printer Rx

Sample;
ThermalPrinter_0.5.f

Fullduplex Communication between PropForth and Processing

Requrired material;
Processing2.2.1
USB-Serial convert tool

or



Fullduplex communication use word"serial".
But this must re-define because this has bug.

```
: serial
        4*
        clkfreq swap u/ dup 2/ 2/
\
\ serial structure
\
\
\ init 1st 4 members to hFF
\
        hFF h1C2
        2dup COG!
        1+ 2dup COG!
        1+ 2dup COG!
        1+ tuck COG!
\
\ next 2 members to h100
\
        1+ h100 swap 2dup COG!
        1+ tuck COG!
\
\ bittick/4, bitticks
\
        1+ tuck COG!
        1+ tuck COG!
\
\ rxmask txmask
\
        1+ swap >m over COG!
        1+ swap >m over COG!
\ rest of structure to 0
        1+ h1F0 swap
        do
```

```
                0 i COG!
        loop
\
        c" SERIAL" numpad ccopy numpad cds W!
        4 state andnC!
\       0 io hC4 + L!   <-- always “0 cogn sersetbreak” (driver don't transmit a breaklength
                                                minimumlength is 16 cycle, at 80MHz this is 200 nanoseconds
\       0 io hC8 + L!   <-- always “0 cogn sersetflag” (CR[d13] is transmitted as CR [d13]LF [d10])
        _serial
;
```

Only 2 lines need to comment or delete.

To connect to Processing, Cog5 start up “serial”.
This manipulate input pointer and output-pointer to connect Cog6 to Cog5.
Please read section6.2 in PropForth.html about these pointer.

IO for propforth is done via an io channel. An io channel is a long which is treated as 2 words. The io channel which connects to the interpreter is at the beginning of the cog data area. It is defined as io. Any cogs io is defined as n cogio.

The structure of the long is 2 words as follows:

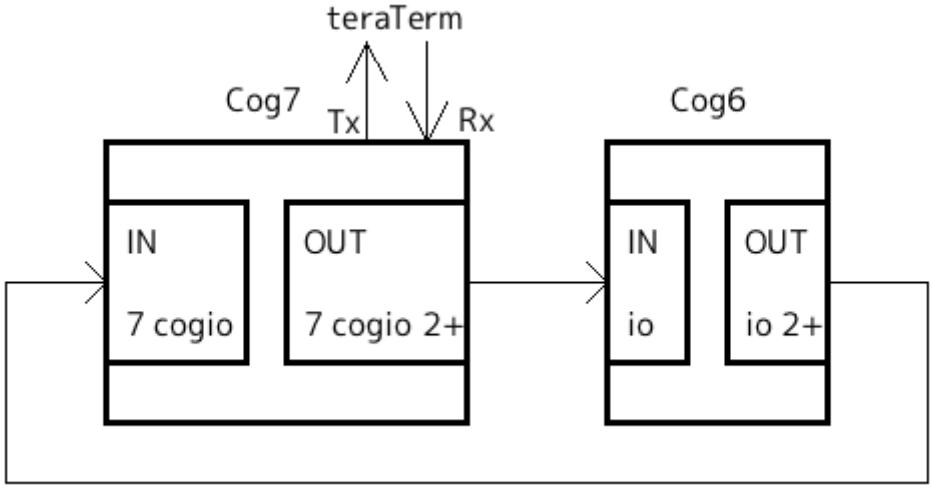
io (word) - this is the input, if the h0100 bit is set, it means the interpreter is ready to accept input. To send a byte to the input write h00cc, where cc is the byte value. This word is used by key? and key

io + 2 (word) - this is a pointer to the where the output of the channel goes This word is used by emit? and emit. If this word is 0, the ouput destination is not valid and emit will simply "throw away the output. If it is not zero, it is assumed to be a pointer to an io channel. Thus the output of an io channel always points to the input of another io channel.

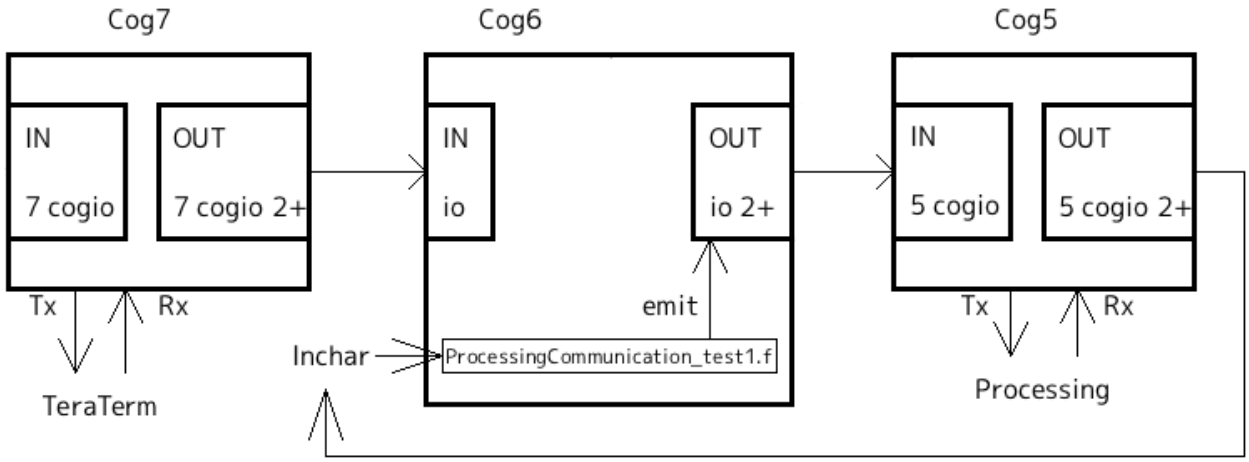
Sample code:Forth_Processing_0.1.f
Word”pre” must be executed before connecting cog6 to cog5.
Output of cog5 is changed to inchar.
Output of cog6 is changed to input of cog5

Word”prost” must be back output and input for cog6 and cog5.
Output of cog5 is back to original.
Output of cog6 is back to input of cog7.

Before word”pre”;



After word”pre”;

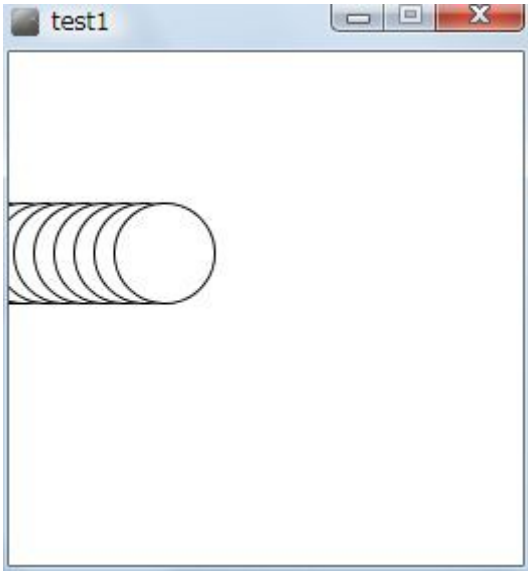


Cog7(serial)'s output still connect to Cog6's input.
If you type any key from TeraTerm, there are in cog6's input buffer.
After executing word”post”, there are something in stack of cog6.

I don't think this is big problem.
But if you dislike this, please modify these code(“pre” and “post”).

Sample code:ProcessingCommunication_test1.f

word”test1”;

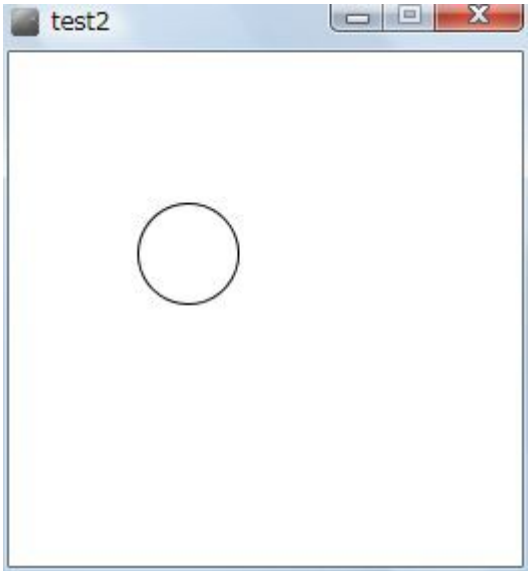


1. Firstly start "test1" on Processing
2. Start “test1”on PropForth
3. Circle moving from left to right

PropForth'test1” merely send data.
Processing”test1” merely receive data.

This check communication from PropForth to Processing.

word”test2”;



1. Start "test2" on Processing
2. Start “test2”on PropForth
3. Clicking mouse on Processing”test2”window
4. Circle moving from left to right

This check communication between PropForth and Processing .

When clicking mouse on Processing”test2”window, Processing send d65 to PropForth.
When PropForth receiving d65, it send data.

word”test3”;

1. Start "test3" on Processing
2. Start “test3”on PropForth
3. Clicking mouse on Processing”test3”window
4. Circle moving from left to right
5. Hitting any key on Processing”test3”window, PropForth”test3” stop

When clicking mouse on Processing”test2”window, Processing send d65 to PropForth.
When PropForth”test3” receiving d65, it send data.
When hitting any key on Processing”test3”, it send d66 to PropForth and
PropForth”test3” stop.

word”test4”;

1. Start "test4" on Processing
2. Start “test4”on PropForth
3. Clicking mouse on Processing”test4”window
4. Repeat counting n Processing”test4”window
5. Hitting any key on Processing”test4”window, it stop.
6. After a while, PropForth”test4” also stop.

When clicking mouse on Processing”test4”window, Processing send d65 to PropForth.
When PropForth”test4” receiving d65, it send data.
When hitting any key on Processing”test4”, it send d66 to PropForth.
Processing”test4 stop.
PropForth”test4” also stop after a while.