Serial Communication

20150405

HalfDuplex Serial Communication

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 serial
```

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

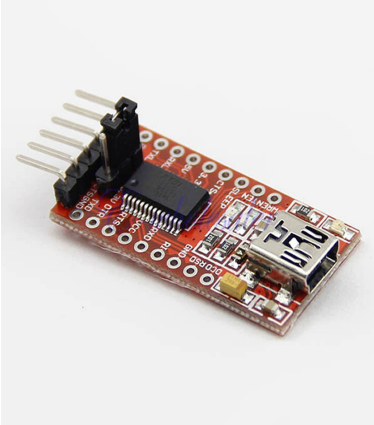
How to use;
Transmit; data serial a_hdserialTx
        Data on stack is transmitted to TX-pin.
Receive; serial a_hdserialRx
        Receiving data is on stack.

Sample;
ThermalPrinter_0.3.f

Communication between PropForth and Processing

Requrired material;
Processing2.2.1
USB-Serial convert tool

or



Fullduplex communication use word"serial".
But this must be re-defined 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
\       0 io hC8 + L!   <-- always 0 cogn sersetflags
        _serial
;
```

Only 2 lines need to comment or delete.

To communicate to Processing, Cog5 start up “serial”.
This maniipulate input pointer and output-pointer to connect Cog6 and 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:

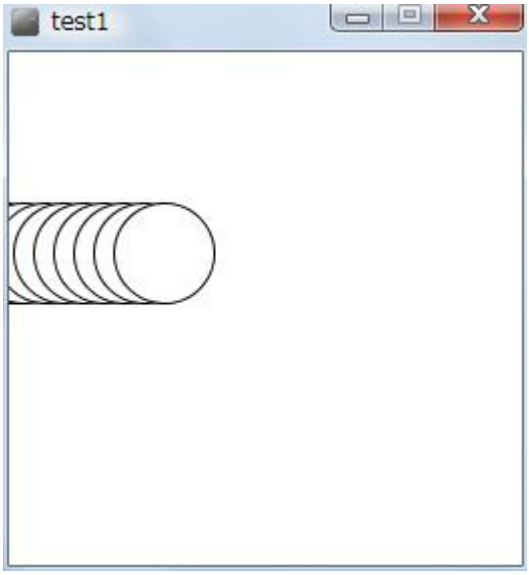
io (word) - this is the input, if the h0100 bit is set, it means the interpreter is ready to accept input. To send a byte to the input write h00cc, where cc is the byte value. This word is used by key? and key

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.

ProcessingCommunication_test1.f

test1;

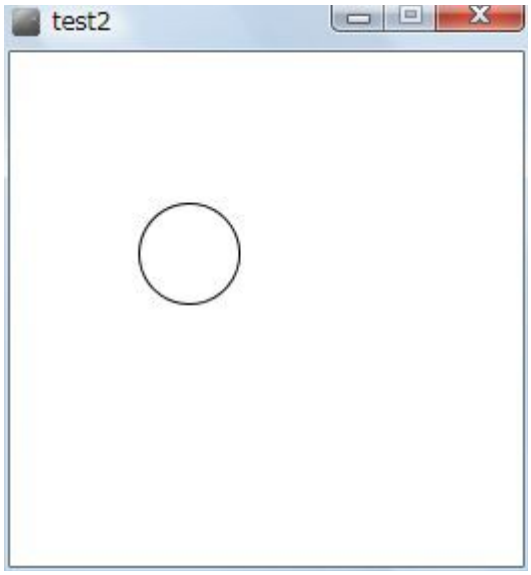
1. Firstly start "test1" on Processing
2. Start “test1”on PropForth
3. Circle moving from left to right



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

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



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

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.

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” 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.