

1.CogDataArea

(figure is decimal)

\$H_cogdata = 840 pointer to the main cogdata area

\$S_cdsz = 224 The size of the cog's data area

cogdata	HUB address
cog0	----- 840 = 840 + (\$S_cdsz*0) [0 cogio .] -----
cog1	1064 = 840 + (\$S_cdsz*1) [1 cogio .] -----
cog2	1288 = 840 + (\$S_cdsz*2) [2 cogio .] -----
cog3	1512 = 840 + (\$S_cdsz*3) [3 cogio .] -----
cog4	1736 = 840 + (\$S_cdsz*4) [4 cogio .] -----
cog5	1960 = 840 + (\$S_cdsz*5) [5 cogio .] -----
cog6	2184 = 840 + (\$S_cdsz*6) [6 cogio .] -----
cog7	2408 = 840 + (\$S_cdsz*7) [7 cogio .] -----

'io' print out CogDataAerea of current-cog.(this address is also io-channel)

Prop0 Cog6 ok

io .

2184 Prop0 Cog6 ok

2.Print-out to TeraTerm

word'.' print out figure from 0 to FFFFFFFF

And it follow space.

Control-code(for example TAB=9) is ignore.

If you don't add space, you can use word'.byte/.word/.long'.

Using sample;

Prop0 Cog6 ok

hex

Prop0 Cog6 ok

0 .

0 Prop0 Cog6 ok

9 .

9 Prop0 Cog6 ok

FFFFFFFF .

-1 Prop0 Cog6 ok

9 .byte

09Prop0 Cog6 ok

FFFFFFFF .byte

FFProp0 Cog6 ok

FFFFFFFF .long

FFFF_FFFFProp0 Cog6 ok

Word'emit' print out character-code.

This is used for printing character.

Control-code also print out.

And space is not added.

Printed characters are depended on your PC's character-code.

Your system might not print out 'B1 emit'. Or different character.

Using sample;

Prop0 Cog6 ok

hex

Prop0 Cog6 ok

0 emit

Prop0 Cog6 ok

9 emit

Prop0 Cog6 ok

30 emit

0Prop0 Cog6 ok

Prop0 Cog6 ok

7D emit

}Prop0 Cog6 ok

7E emit

?Prop0 Cog6 ok

7F emit

Prop0 Cog6 ok

A1 emit

Prop0 Cog6 ok

B1 emit

↯ Prop0 Cog6 ok

3.Definition of port

Using sample;

```
hex
Prop0 Cog6 ok
d16 wconstant _latch
Prop0 Cog6 ok
_latch >m constant _latchm
Prop0 Cog6 ok
_latchm .long
0001_0000Prop0 Cog6 ok
```

If desired saving memory, you can use word 'wconstant'.
(ACTUALLY, IT ALWAYS SHOULD USE 'CONSTANT')
But port can define only "P0 - P15"

Using sample;

```
Prop0 Cog6 ok
d8 wconstant _latch
Prop0 Cog6 ok
_latch >m wconstant _latchm    \ saving 2bytes
Prop0 Cog6 ok
_latchm .long
0000_0100Prop0 Cog6 ok
```

```
Prop0 Cog6 ok
d16 wconstant _latch
Prop0 Cog6 ok
_latch >m wconstant _latchm    \ saving 2bytes
Prop0 Cog6 ok
_latchm .long
0000_0000Prop0 Cog6 ok
```

4.String

Case1 Using defined word when string is 2,3 pieces.

There is inside PropForthStartKernel.f.

```
: (version) c" PropForth v5.0 2012JAN09 14:30 0" ;
```

Using sample;

```
Prop0 Cog6 ok
```

```
(version) .cstr cr
```

```
PropForth v5.0 2012JAN09 14:30 0
```

```
Prop0 Cog6 ok
```

```
Prop0 Cog6 ok
```

```
hex
```

```
Prop0 Cog6 ok
```

```
(version) 40 dump
```

```
0AC8 0040:
```

```
0AC8: 20 50 72 6F 70 46 6F 72 74 68 20 76 35 2E 35 20  PropForth v5.5
```

```
0AD8: 32 30 31 33 46 65 62 32 30 20 31 31 3A 33 30 20  2013Feb20 11:30
```

```
0AE8: 30 00 61 00 BC 0A 84 70 72 6F 70 00 4F 00 B2 0A  0.a....prop.O...
```

```
0AF8: EE 0A 87 76 65 72 73 69 6F 6E 4F 00 C8 0A FA 0A  ...versionO.....
```

```
Prop0 Cog6 ok
```

Case2 Using defined word when string's length is same.

```
: test
```

```
c" JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
```

```
1+ swap hC min 1 max 1- 3 u* +
```

```
3 0 do dup C@ emit 1+ loop drop
```

```
;
```

Using sample;

```
Prop0 Cog6 ok
```

```
2 test cr
```

```
FEB
```

```
Prop0 Cog6 ok
```

Prop0 Cog6 ok

hex

Prop0 Cog6 ok

words test

NFA (Forth/Asm Immediate eXecute) Name

44BE F test

Prop0 Cog6 ok

44BE 30 dump

44BE 0030:

44BE: 84 74 65 73 74 00 66 26 24 4A 41 4E 46 45 42 4D .test.f&\$JANFEBM

44CE: 41 52 41 50 52 4D 41 59 4A 55 4E 4A 55 4C 41 55 ARAPRMAYJUNJULAU

44DE: 47 53 45 50 4F 43 54 4E 4F 56 44 45 43 00 A0 14 GSEP OCTNOVDEC...

Prop0 Cog6 ok

Case3 Allocating strings when using many different length strings

: s, parsenw dup C@ 1+ bounds dup rot2 do C@++ c, loop drop ;

wvariable main -2 allot s, Input s, Output s, Collection s, Feature s, EndCollection
wvariable global -2 allot s, UsagePage s, LogicalMinimum s, LogicalMaximum
s, PhysicalMinimum s, PhysicalMaximum s, UnutExponent s, Unit s, ReportSize s,
ReportID

s, ReportCount s, Push s, Pop

wvariable local -2 allot s, Usage s, UsageMinimum s, UsageMaximum s, DesignatorIndex
s, DesignatorMinimum s, StringIndex s, StringMinimum s, StringMaximum s, Delimiter
wvariable Attribute -2 allot s, Data s, Constant s, Array s, Variable s, Absolute s, Relative
wvariable Collection -2 allot s, Physical s, Application s, Logical s, Report s, NamedArray
s, UsageSwitch s, UsageModifier

wvariable UsagePage -2 allot s, Undefined s, Generic_Desktop_Controls s,
Simulation_Controls s, VR_Controls s, Sport_Controls

s, Game_Controls s, Generic_Device_Controls s, Keyboard/Keypad s, LEDs s, Button s,
Ordinal s, Telephony s, Consumer

wvariable Generic_Desktop_Page -2 allot s, Undefined s, Pointer s, Mouse s, Reserved s,
Joystick s, Game_Pad s, Keyboard

s, Keypad s, Multi-axis_Controller s, X s, Y s, Z s, Rx s, Ry s, Rz s, Slider s, Dial s, Wheel

Prop0 Cog6 ok

hex

Prop0 Cog6 ok
main 200 dump

4B82 0200:

4B82: 05 49 6E 70 75 74 06 4F 75 74 70 75 74 0A 43 6F .Input.Output.Co
4B92: 6C 6C 65 63 74 69 6F 6E 07 46 65 61 74 75 72 65 llection.Feature
4BA2: 0D 45 6E 64 43 6F 6C 6C 65 63 74 69 6F 6E 7A 4B .EndCollectionzK
4BB2: 86 67 6C 6F 62 61 6C 76 4F 00 09 55 73 61 67 65 .globalvO..Usage
4BC2: 50 61 67 65 0E 4C 6F 67 69 63 61 6C 4D 69 6E 69 Page.LogicalMini
4BD2: 6D 75 6D 0E 4C 6F 67 69 63 61 6C 4D 61 78 69 6D mum.LogicalMaxim
4BE2: 75 6D 0F 50 68 79 73 69 63 61 6C 4D 69 6E 69 6D um.PhysicalMinim
4BF2: 75 6D 0F 50 68 79 73 69 63 61 6C 4D 61 78 69 6D um.PhysicalMaxim
4C02: 75 6D 0C 55 6E 75 74 45 78 70 6F 6E 65 6E 74 04 um.UnutExponent.
4C12: 55 6E 69 74 0A 52 65 70 6F 72 74 53 69 7A 65 08 Unit.ReportSize.
4C22: 52 65 70 6F 72 74 49 44 0B 52 65 70 6F 72 74 43 ReportID.ReportC
4C32: 6F 75 6E 74 04 50 75 73 68 03 50 6F 70 20 B2 4B ount.Push.Pop .K
4C42: 85 6C 6F 63 61 6C 4F 00 05 55 73 61 67 65 0C 55 .localO..Usage.U
4C52: 73 61 67 65 4D 69 6E 69 6D 75 6D 0C 55 73 61 67 sageMinimum.Usag
4C62: 65 4D 61 78 69 6D 75 6D 0F 44 65 73 69 67 6E 61 eMaximum.Designa
4C72: 74 6F 72 49 6E 64 65 78 11 44 65 73 69 67 6E 61 torIndex.Designa
4C82: 74 6F 72 4D 69 6E 69 6D 75 6D 0B 53 74 72 69 6E torMinimum.Strin
4C92: 67 49 6E 64 65 78 0D 53 74 72 69 6E 67 4D 69 6E gIndex.StringMin
4CA2: 69 6D 75 6D 0D 53 74 72 69 6E 67 4D 61 78 69 6D imum.StringMaxim
4CB2: 75 6D 09 44 65 6C 69 6D 69 74 65 72 42 4C 89 41 um.DelimiterBL.A
4CC2: 74 74 72 69 62 75 74 65 4F 00 04 44 61 74 61 08 ttributeO..Data.
4CD2: 43 6F 6E 73 74 61 6E 74 05 41 72 72 61 79 08 56 Constant.Array.V
4CE2: 61 72 69 61 62 6C 65 08 41 62 73 6F 6C 75 74 65 ariable.Absolute
4CF2: 08 52 65 6C 61 74 69 76 65 61 C0 4C 8A 43 6F 6C .Relativea.L.Col
4D02: 6C 65 63 74 69 6F 6E 72 4F 00 08 50 68 79 73 69 llectionrO..Physi
4D12: 63 61 6C 0B 41 70 70 6C 69 63 61 74 69 6F 6E 07 cal.Application.
4D22: 4C 6F 67 69 63 61 6C 06 52 65 70 6F 72 74 0A 4E Logical.Report.N
4D32: 61 6D 65 64 41 72 72 61 79 0B 55 73 61 67 65 53 amedArray.UsageS
4D42: 77 69 74 63 68 0D 55 73 61 67 65 4D 6F 64 69 66 witch.UsageModif
4D52: 69 65 72 69 FE 4C 89 55 73 61 67 65 50 61 67 65 ieri.L.UsagePage
4D62: 4F 00 09 55 6E 64 65 66 69 6E 65 64 18 47 65 6E O..Undefined.Gen
4D72: 65 72 69 63 5F 44 65 73 6B 74 6F 70 5F 43 6F 6E eric_Desktop_Con

Prop0 Cog6 ok

Using sample;

\ Print out string

\ (n1 n2 --) n1:index(0,1,2,...,n) n2:string-array's address

: dispStr

swap dup 0 <>

if


```
0 do
  dup C@ + 1+
loop
else
  drop
then
.cstr
;
```

```
: test 3 main dispStr cr ; \ Print out index=3 in 'main'string-array
```

```
Prop0 Cog6 ok
test
Feature
Prop0 Cog6 ok
```

5.ROM font

Reference:

Propeller Manual V1.1 Chapter1 Page30 - Page34

HydraGameDevManual-v1.0.1.pdf Chapter16 Page336 - Page338

ROM-font inside propeller is \$8000-\$BFFF(4096Longs).

1Block-size(including 2characters) is 128byte(32Longs).

All are 128locks(128 X 2characters).

2Character's column(horizon derrection) length are 4byte(1Long).

Even character is even dots(b0,2,...b26,b28,b30).

Odd character is odd dots(b1,2,...b27,b29,b31).

2Character's row(vertical derrection) length are 32.

Character is 16 X 32pixel.

1. Word'font1 display combined fonts.

fl

: demo1

h8000

begin

dup hC000 = if drop h8000 then

dup ." \$" hex . cr

d32 0 do

dup L@ 1

d32 0 do 2dup and if 1 else 0 then . 1 lshift loop 2drop cr

4 +

loop

cr cr

fkey? swap drop until

drop

;

There are "@" and "A" below; But it cannot see characters.

\$9000

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
```

0000000001010101010101010101000000
000000010101011111111110101010000
00000001011110101010101111010000
00000101111010101010101011110100
00000111111010000000001011110100
00000111111000000000000011111100
000011111110000000000000001111100
000011111110000101010101001111100
000011110101111111111111101111100
000011110101111101011111011111100
000011110101111101011111011111100
000011110100101010101010111111100
00001111110000101010001011110100
000011111100000000000000001010100
000001111110000000000000001111100
0000011111101000000000000011111100
00000101111010101010101011110100
00000101011010101010101011010100
00000101010000101010100001010100
000001010100000000000000001010100
000001010100000000000000001010100
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000

2. A little bit separate even and odd character

fl

: demo2

h8000

begin

dup hC000 = if drop h8000 then

dup ." \$" hex . cr

d32 0 do

dup L@ 1

d32 0 do 2dup and

if

i 2 u/mod drop

if

```

        h23    \ # - odd
    else
        h21    \ ! - even
    then
    else
        bl
    then
    emit 1 lshift loop 2drop cr
    4 +
loop
cr cr
fkey? swap drop until
drop
;

```

\$9000

```

# # # # #
# # # # # # # #
# # # #!#!#!# # # #
# #!#! ! ! ! !#!# #
# #!#! ! ! ! ! !#!# #
#!#!#! !      !#!# #
#!#!#!      !#!#!#
!#!#!#      #!#!#
!#!#!#  ! ! ! ! #!#!#
!#!# # #!#!#!#!#!# #!#!#
!#!# # #!#!# # #!#!# #!#!#
!#!# # #!#!# # #!#!# #!#!#
!#!# # ! ! ! ! !#!#!#
!#!#!#  ! ! !  !#!# #
!#!#!#      # # #
#!#!#!      #!#!#

```

```

#####
# # # ! ! ! ! ! ! ! ! # # #
# # # ! ! ! ! ! ! ! ! # # #
# # # ! ! ! ! ! ! ! ! # # #
# # # ! ! ! ! ! ! ! ! # # #
# # # ! ! ! ! ! ! ! ! # # #
# # # ! ! ! ! ! ! ! ! # # #

```

3. Display even/odd character separately

fl

```

wvariable odd
: demo3
h8000
begin
  dup hC000 = if drop h8000 then
    dup ." $" hex . cr
    dup
    2 0 do
      i 1+ odd W!
      d32 0 do
        dup L@ odd W@
        d16 0 do \ print 1-row (16bits)
          2dup and
          if
            h2A \ *
          else
            bl \ blank
          then
            emit
            2 lshift
          loop
        2drop
      cr
      4 +
    loop
  loop

```

```

    cr cr
    d128 -
loop
drop
    d128 +      \ next character
fkey? swap drop until
drop
;

$9000

```

```

    ****
    ****
    ****
    ****      ****
    ***       ***
    ***       **
    ***  ****  ***
    **  ****  ***
    **  **  **  **
    **  **  **  **
    **  ****
    ***  ****  ****
    ***
    ***       **
    ****      ****
    ****
    ****
    ****

```


*** ***
*** ***
*** ***
*** ***
*** ***
*** ***

*** ***
*** ***
*** ***
*** ***
*** ***
*** ***
*** ***
*** ***
*** ***
*** ***

6. *Analysys POV*

POV display 8X8-pixels by using QuickStart board.

Word'drive_LED' drive LED-On/Off according to LED_data array.

Word'getRow' get row-data.

Word'pov' operate shorting-pixel (from16 X 32pixel to 8 X 8pixel).

Actually, 8X8fonts looks bad.

Converting colum from 16dots to 8dots.

Converting row from 32dots to 8dots.

Word'demo4' print out datas saved inside LED_data to TeraTerm.

Character's top is screen-right.

Not clear font.

Changing column data by bit_mask's data.

Changing row data by operation inside [do-loop]. (refering A-B inside demo4's code)

Prop0 Cog6 ok

pov

-- LEDs od QS board start to blink ---

Prop0 Cog6 ok

demo4

-- It start to display on TeraTerm ---

-----> character top

*

*

*

** * *

** ** *

* *

**

* *

** **

* ** *

* * * *
* * * *
* * * *

* *

* *

* *

* *

* * *
* * *
* * *
* * *

* *

* *
* *
* *
* *
* *

* *
* *
* *
* *

* * *
* * *
* * *
* *
* *

* *
* *
* *
*
*

*

* *
* *
* *
* * *
***** *

7. Operating on another cog

When operating code on another cog, cogID is pointed out

```
: test 5 0 do i . loop ;
```

```
c" test" 5 cogx
```

Automatically, cogID is assigned on demo5;

Sample;

```
Prop0 Cog6 ok
```

```
demo5
```

```
available cogID:5
```

```
available cogID:4
```

```
available cogID:3
```

```
available cogID:2
```

```
available cogID:1
```

```
available cogID:0
```

```
No free Cog
```

```
Prop0 Cog6 ok
```

```
cogID:0x60543210
```

```
Cog:0 #io chan:1          RUNNING TEST-F
```

```
Cog:1 #io chan:1          RUNNING TEST-E
```

```
Cog:2 #io chan:1          RUNNING TEST-D
```

```
Cog:3 #io chan:1          RUNNING TEST-C
```

```
Cog:4 #io chan:1          RUNNING TEST-B
```

```
Cog:5 #io chan:1          RUNNING TEST-A
```

```
Cog:6 #io chan:1 PropForth v5.5 2013Feb20 11:30 0 6(0)->7(0)
```

```
Cog:7 #io chan:1          SERIAL 7(0)->6(0)
```

```
Prop0 Cog6 ok
```

```
Prop0 Cog6 ok
```

```
-- Reset all cogs --
```

```
all_cog_reset
```

```
CON:Prop0 Cog0 RESET - last status: 0 ok
```

CON:Prop0 Cog1 RESET - last status: 0 ok

CON:Prop0 Cog2 RESET - last status: 0 ok

CON:Prop0 Cog3 RESET - last status: 0 ok

CON:Prop0 Cog4 RESET - last status: 0 ok

CON:Prop0 Cog5 RESET - last status: 0 ok

Cog:0 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:1 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:2 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:3 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:4 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:5 #io chan:1 PropForth v5.5 2013Feb20 11:30 0

Cog:6 #io chan:1 PropForth v5.5 2013Feb20 11:30 0 6(0)->7(0)

Cog:7 #io chan:1 SERIAL 7(0)->6(0)

Prop0 Cog6 ok

Prop0 Cog6 ok