J1b SwapForth Reference

James Bowman
jamesb@excamera.com
Excamera Labs
Pescadero
California USA

ANS Forth Compliance Label

J1b SwapForth is an ANS Forth System

Providing names from the **Core Extensions** word set
Providing the **Double-Number** word set
Providing the **Double-Number Extensions** word set
Providing names from the **Exception** word set
Providing the **Exception Extensions** word set
Providing the **Facility** word set
Providing names from the **Facility Extensions** word set
Providing the **String** word set
Providing the **Programming-Tools** word set
Providing names from the **Programming-Tools Extensions** word set

Contents

Chapter 1

Getting started

J1b SwapForth is a 32-bit version of SwapForth, intended as an interactive Forth system using very little logic and RAM. The J1b and peripherals use 1200 logic elements. SwapForth uses 10.5 Kbytes of RAM, leaving about 22 Kbytes for the application.

After installing the Papilio tools, you can run on a Papilio DUO like this

```
git clone git@github.com:jamesbowman/swapforth.git
cd swapforth/j1b
papilio-prog -v -f xilinx/j1-papilioduo.bit
python shell.py -h /dev/ttyUSB0
```

(where `/dev/ttyUSB0` is the appropriate port your DUO was assigned). You should see something like

```
Contacting... established
Loaded 299 words
>
```

And you can now try the usual Forth things, e.g.

```
1 2 + .
3   ok
```

There is a complete core ANS-compatible Forth system running on the board, including a compiler.

1.1 Some demos

There is an Easter date calculator

```
new
#include ../demos/easter.fs
```

Now you can do

```
>2015 .easter
2015 April 5   ok
```

Or even

```
>: 20easters
+  2035 2015 do
+    cr i .easter
+  loop
+;
 ok
>20easters

2015 April 5
2016 March 27
```

```
    2017 April 16
    2018 April 1
    2019 April 21
    2020 April 12
    2021 April 4
    2022 April 17
    2023 April 9
    2024 March 31
    2025 April 20
    2026 April 5
    2027 March 28
    2028 April 16
    2029 April 1
    2030 April 21
    2031 April 13
    2032 March 28
    2033 April 17
    2034 April 9   ok
```

1.2 Building from Scratch

After installing the icestorm tools, run

```
rm build/*
make -C xilinx
```

This will produce `j1-papilioduo.bit` - but it only contains the very barebones system; the rest of SwapForth still needs to be compiled. To do this, load `j1-papilioduo.bit` and start the shell:

```
$ papilio-prog -v -f xilinx/j1-papilioduo.bit
$ python shell.py -h /dev/ttyUSB0 -p ../common/
Contacting... established
Loaded 127 words
>
```

Then compile the rest of SwapForth and write the finished executable with these commands:

```
#include swapforth.fs
#flash build/nuc.hex
#bye
```

Now run `make -C xilinx` again - this compiles an FPGA image with the complete code base built-in.

Chapter 2

Available Words

2.1 ANS Core Words

J1b SwapForth implements most of the core ANS 94 Forth standard. Implemented words are:

```
   !  # #> #s ’ ( * */ */mod + +!  +loop , - .  ." / /mod 0< 0=
 1+ 1- 2!  2* 2/ 2@ 2drop 2dup 2over 2swap :  ; < <# = > >body
 >in >number >r ?dup @ abort abort" abs accept align aligned
allot and base begin bl c!  c, c@ cell+ cells char char+ chars
constant count cr create decimal depth do does> drop dup else
emit environment?  evaluate execute exit fill find fm/mod here
hold i if immediate invert j key leave literal loop lshift m*
 max min mod move negate or over postpone quit r> r@ recurse
repeat rot rshift s" s>d sign sm/rem source space spaces state
 swap then type u.  u< um* um/mod unloop until variable while
word xor [ [’] [char] ]
```

J1b SwapForth also implements the following standard words:

```
   -trailing .( .r .s /string 0<> 0> 2>r 2constant 2literal 2r>
 2r@ 2variable :noname <> ?  ?do again ahead at-xy blank c" case
 cmove cmove> compare compile, convert cs-pick cs-roll d+ d- d.
 d.r d0< d0= d2* d2/ d< d= d>s dabs dmax dmin dnegate dump
endcase endof erase false hex key?  m*/ m+ marker ms nip of
pad page parse pick refill restore-input roll save-input search
 see sliteral source-id throw to true tuck u.r u> unused value
within words [compile] [else] [if] [then] \
```

Double numbers are supported using the standard `.` suffix. The Forth 200x

number prefixes are supported: `$` for hex, `#` for decimal, `%` for binary, and `'c'` for character literals. **`parse-name`** is also implemented.

2.2 Additional Words

The following words are not standard. Some are traditional Forth words, others are specific to the J1b SwapForth implementation.

.x

```
( n -- )
```

display n as an 8-digit hex number

-rot

```
( x1 x2 x3 -- x3 x1 x2 )
```

rotate the top three stack entries

---

bounds

```
( start cnt -- start+cnt start )
```

prepare to loop on a range

---

forth

```
( -- a )
```

variable: most recent dictionary entry

---

io!

```
( x a -- )
```

store `x` to IO port `a`

---

io@

```
( a -- x )
```

fetch from IO port `a`

---

new

```
( -- )
```

restore code and data pointers to the power-up state

---

s,

```
( a u -- )
```

add the u-character string a to the data space

---

serialize

```
( -- )
```

display all of current memory in base 36

---

tth

```
( -- a )
```

variable: tethered mode

---

w!

```
( -- a )
```

16-bit store

---

w@

```
( -- a )
```

16-bit signed fetch

---

uw@

```
( -- a )
```

16-bit unsigned fetch

---

pinMode

`( mode pin -- )`

Set `pin` to either output or input

Chapter 3

Using SwapForth

3.1 Raw UART access

At boot, SwapForth listens for a command on the UART. Connection parameters are 921600 8N1, and any terminal program should be able to connect. Note that the hardware board uses DTR as a reset signal, so you should make sure that it is set OFF by the terminal program:

```
$ miniterm.py --dtr=0 /dev/ttyUSB5 921600
--- Miniterm on /dev/ttyUSB5: 921600,8,N,1 ---
--- Quit: Ctrl+]  |  Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
--- forcing DTR inactive

  ok
  ok
  ok
```

3.2 The SwapForth shell

The SwapForth shell is a Python program that runs on the host PC. It has a number of advantages over raw UART access:

- command-line editing
- command history
- word completion on TAB

- local file `include`
- `^C` for interrupt

3.2.1 Invocation

The shell is a Python program. To run it, go to the appropriate directory and type:

```
python shell.py -h /dev/ttyUSB0
```

3.2.2 Command reference

`#bye` - quit SwapForth shell

`#flash` - copy the target state to a local file

`#include` - send local source file

`#noverbose` - turn off include echo

`#time` - measure execution time

3.3 Tethered Mode

J1b SwapForth supports *tethered mode*, which makes the UART protocol easier to use for host programs. The SwapForth shell uses tethered mode. To enter tethered mode, write one to the variable **`tth`** :

```
1 tth !
```

In tethered mode, **`accept`** transmits byte value 30 (hex `1e`, ASCII code RS). This allows the listening program to know that the target machine is ready to accept a line of input. In addition, **`accept`** does not echo characters as they are typed.

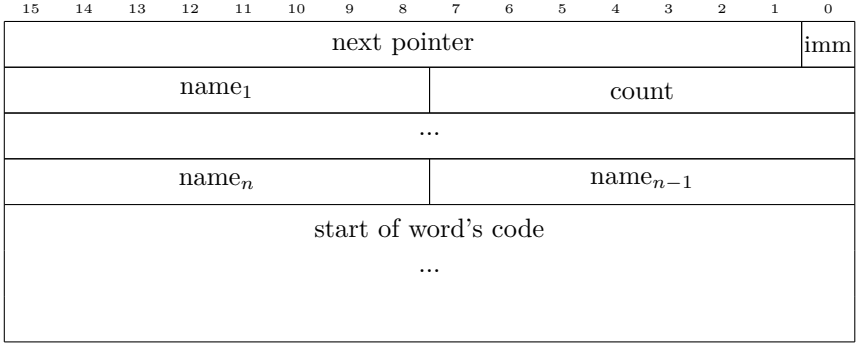
Chapter 4

Memory

4.1 Memory map

The J1b SwapForth implementation uses 16 Kbytes of RAM for code, and 16 Kbytes data. The standard Forth words access this RAM. Cells are 32-bits, and must be aligned to a 32-bit boundary.

4.2 Dictionary Layout

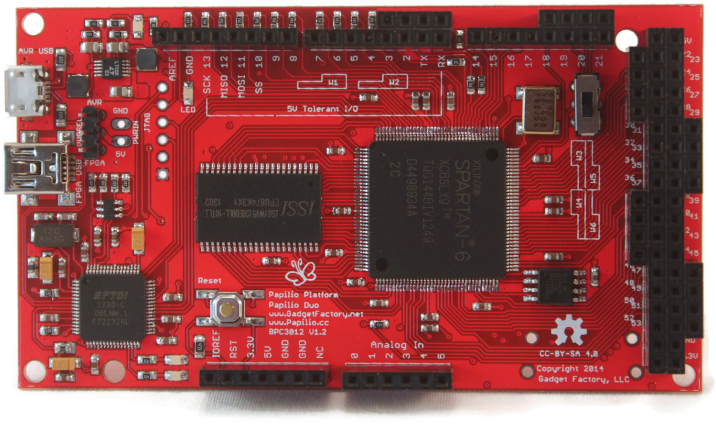


The SwapForth dictionary is a linked list; the variable `forth` holds the start of this list. Each dictionary entry has the following fields:

- **next pointer** - address of the next dictionary entry, or zero for the last dictionary entry
- **imm** - immediate bit, set if the word is immediate
- **count** - length of the name, in characters, 1-31
- **name$_1$ - name$_n$** - characters in name. If the length of the name is even, then a padding byte is appended

Chapter 5

Papilio DUO Hardware interface



The J1a for Papilio DUO includes connections to the Papilio DUO IO pins. Access to peripherals is via the `io@` and `io!` words. Peripherals are port-mapped into a 16-bit IO address space.

For example, to flash the onboard LED

```
: blink
  OUTPUT 13 pinMode \ Set pin 13 to OUTPUT
  0                 \ LED initially off
  begin
    dup 13 io!      \ write to LED
    invert          \ flip LED state
    200 ms          \ pause 1/5 of second
  again
;
blink
```

5.1 Port Map

5.1.1 $0000-$036: IO pins

The read-write port at address $0001 is for direct access to the 54 IO pins on the Papilio DUO header. Reading a port gives the current state of the pin. If the pin is set for OUTPUT, then writing 0 or 1 to the port sets the pin state.

Note that pin direction is controlled by the corresponding registers at $0100-$136

5.1.2 $0040: switch SW1

Reads the current state of the user switch, either 0 or 1.

5.1.3 $0100-$136: IO pin direction

Each port controls the direction of the corresponding IO pin. 0 sets the pin to input, 1 means sets the pin to output. At boot all pins are inputs. The SwapForth words `INPUT`, `OUTPUT` and `pinMode` can be used to set the direction of pins, for example:

```
OUTPUT 13 pinMode \ Set pin 13 to OUTPUT
INPUT 52 pinMode  \ Set pin 52 to INPUT
```

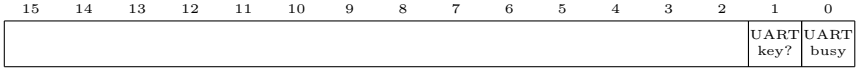
5.1.4 $1000: UART data

The read-write port at address $1000 is for UART transmission or reception. Writing to the port starts transmission of a byte, reading the port returns the incoming byte.

Standard words **`key`** , **`key?`** and **`emit`** can be used to access this port.

5.1.5 $2000: UART status

Read-only port $2000 contains the input signals from the UART.



Index