amforth 2.7 Reference Card

Arithmetics

```
1-        ( n1 -- n2 )
1+        ( n1 -- n2 )
2/        ( n1 -- n2 )
2*        ( n1 -- n2 )
abs       ( n1 -- u1 )
><        ( n1 -- n2 )
cell+     ( n1 -- n2 )
cells     ( n1 -- n2 )
d2/       ( d1 -- d2 )
d2*       ( d1 -- d2 )
dinvert   ( d1 -- d2)
d-        ( d1 d2 -- d3 )
d+        ( d1 d2 -- d3)
invert    ( n1 -- n2)
log2      ( n1 -- n2 )
lshift    ( n1 n2 -- n3)
-         ( n1 n2 -- n3 )
mod       ( n1 n2 -- n3)
m*        ( n1 n2 -- d)
*         ( n1 n2 -- n3 )
+         ( n1 n2 -- n3)
+!        ( n addr -- )
rshift    ( n1 n2 -- n3 )
/         ( n1 n2 -- n3)
/mod      ( n1 n2 -- rem quot)
*/        (n1 n2 n3 -- n4)
*/mod     ( n1 n2 n3 -- rem quot)
um/mod    ( ud u2 -- rem quot)
um*       ( u1 u2 -- d)
u/mod     (u1 u2 -- rem quot)
u*/mod    ( u1 u2 u3 -- rem quot)
0         ( -- 0 )
```

Character IO

```
bl        ( -- 32 )
cr        ( -- )
emit      ( -- eaddr)
emit?     ( -- c)
key       ( -- c )
key?      ( -- f)
/key      ( -- f)
space     ( -- )
spaces    ( n -- )
type      ( addr n -- )
```

Compare

```
d>        ( d1 d2 -- flag )
d<        ( d1 d2 -- flasg)
=         ( n1 n2 -- flag )
0=        ( n -- flag )
>         ( n1 n2 -- flag )
0>        ( n1 -- flag )
<         ( n1 n2 -- flasg)
0<        ( n1 -- flag)
max       ( n1 n2 -- n1|n2 )
min       ( n1 n2 -- n1|n2 )
<>        ( n1 n2 -- flag)
0<>       ( n -- flag )
u>        ( u1 u2 -- flag )
u<        ( u1 u2 -- flasg)
```

Compiler

```
\         ( -- )
[']       ( -- XT )
code      ( -- )
:         ( -- )
:noname   ( -- xt )
constant  ( n -- )
does>     ( -- )
."        ( -- )
Edefer    ( n <name> -- )
else      ( addr1 -- addr2)
end-code  ( -- )
exit      ( -- )
          R(xt --)
immediate ( -- )
[         ( --  )
literal   ( n -- )
(         ( -- )
]         ( --  )
Rdefer    ( n <name> -- )
recurse   ( -- )
s,        ( addr len -- )
;         ( -- )
s"        ( <cchar> -- )
state     ( -- addr )
then      ( addr -- )
until     ( addr -- )
user      ( n -- )
value     ( n <name> -- )
variable  ( -- )
```

Control Structure

```
again     ( addr -- )
begin     ( -- addr )
do        ( -- addr )
i         ( -- n )
          ; R( loop-sys -- loop-sys)
if        ( -- addr )
j         ( -- n )
          ; R( loop-sys1 loop-sys2 -- loop-sys1 loop-sys2)
leave     ( -- )
          R(loop-sys --)
loop      ( addr -- )
+loop     ( addr -- )
?do       ( -- addr )
repeat    (addr1 -- addr2 )
unloop    ( -- )
          ; R(loop-sys -- )
while     ( dest -- orig dest )
```

Conversion

```
d>s       ( d1 -- n1 )
s>d       ( n1 -- d1 )
```

Dictionary

```
,         ( n -- )
compile   ( -- )
create    ( -- )
'         ( -- XT )
```

Exceptions

```
abort     ( n*x -- )
          R(n*y --)
abort"    ( n*x -- )
          R(n*y --)
catch     ( xt -- )
handler   ( -- addr )
throw     ( n -- )
```

Hardware Access

```
rx0       ( -- c)
rx0?      ( -- f)
>usart0   ( -- )
tx0       (c -- )
tx0?      ( -- f)
+usart0   ( -- )
```

IO

```
refill    ( -- f )
```

Interrupt

```
int@      ( i -- xt )
-int      ( -- sreg )
+int      ( --  )
int!      ( xt i -- )
#int      ( -- n )
```

Logic

```
and       ( n1 n2 -- n3 )
negate    ( n1 -- n2 )
not       ( flag -- flag' )
or        ( n1 n2 -- n3 )
xor       ( n1 n2 -- n3)
```

MCU

```
-jtag     ( n1 -- n2 )
-wdt      ( n1 -- n2 )
sleep     ( -- )
wdr       ( -- )
```

MCU

```
spirw     ( txbyte -- rxbyte)
```

Memory

```
c@        ( addr - c1 )
cmove>    (addr-from addr-to n -- )
c!        ( c addr -- )
e@        ( addr - n)
e!        ( n addr -- )
@         ( addr -- n )
fill      ( c-addr u c -- )
i@        ( addr -- n1 )
i!        ( n addr -- )
!         ( n addr -- )
```

Multitasking

```
pause     ( -- )
```

Numeric IO

```
base      ( -- addr )
decimal   ( -- )
digit     ( c base -- number flag )
.         ( n -- )
hex       ( -- )
hld       ( -- addr )
hold      ( c -- )
<#        ( -- )
number    (addr -- n )
#         ( d1 -- d2)
#>        ( d1 -- addr count )
#s        ( d1 -- 0)
sign      ( n -- )
u.        ( n -- )
```

Stack

```
depth     ( -- n )
drop      ( n -- )
dup       ( n -- n n )
over      ( n1 n2 -- n1 n2 n1 )
?dup      ( n1 -- [ n1 n1 ] | 0)
rot       ( n1 n2 n3 -- n2 n3 n1)
r@        ( -- n)
          R(n -- n)
r>        ( -- n )
          ; R( n --)
swap      ( n1 n2 -- n2 n1)
>r        ( n -- )
          ; R( -- n)
```

Stackpointer

```
rp0       ( -- addr)
rp@       (  -- n)
rp!       ( n  -- )
          ; R( -- xy)
sp        ( -- addr)
sp0       ( -- addr)
sp@       (  -- n)
sp!       ( addr -- i*x)
```

String

```
count     ( addr -- addr+1 n)
cscan     ( addr1 n1 c -- addr1 n2 )
cskip     ( addr1 n1 c -- addr2 n2 )
parse     ( char "ccc" -- c-addr u )
place     ( addr1 len1 addr2 -- )
/string   ( addr1 u1 n-- addr2 u2 )
```

System

```
accept    ( addr n1 -- n2 )
allot     ( n -- )
cold      ( -- )
defer@    ( xt1 -- xt2 )
defer!    ( xt1 xt2 -- )
execute   ( xt -- )
f_cpu     ( -- f_cou )
>in       ( -- addr )
interpret ( -- )
          ; R(i*x - j*x )
is        ( xt1 c<char> -- )
#tib      ( -- addr )
?execute  ( xt|0 -- )
quit      ( -- )
source    ( -- addr n )
up@       ( -- addr )
up!       ( addr -- )
```

System Pointer

```
dp        ( -- eaddr)
edp       ( -- eaddr)
head      ( -- eaddr)
heap      ( -- eaddr)
here      ( -- addr )
pad       ( -- addr )
tib       ( -- addr )
turnkey   ( -- eaddr)
```

System Value

```
baud0     ( -- v)
```

Time

```
1ms       ( -- )
```

Tools

```
[char]    ( -- c )
char      ( -- c )
.s        ( -- )
find      ( addr -- [ addr 0 ] | [ xt [-1|1]] )
icount    ( adr -- adr n )
itype     ( addr n --  )
noop      ( -- )
to        ( n <name> -- )
unused    ( -- n )
ver       ( -- )
word      ( c -- addr )
words     ( -- )
```

internal/hidden

```
(branch)  (-- )
(?branch) (f -- )
(constant)(-- addr )
(create)  (--  )
(do)      (limit counter -- )
          R(-- limit counter )
(does>)   (-- )
(defer)   (i*x -- j*x )
(literal) (-- n1 )
(loop)    (-- )
          R(limit counter -- limit counter+1|)
(+loop)   (n1 -- )
          R(llimit counter -- limit counter+n1|)
(?do)     (limit counter -- )
          R(-- limit counter| )
(rp0)         .dw XT_FETCH
              .dw XT_EXIT
(sp0)     ( -- addr)
(spm)     (spmcsr x addr -- )
(to)      ( n -- )
          R(IP -- IP+1)
(user)    (-- addr )
(variable)(-- addr )
Edefer@   ( xt1 -- xt2 )
Edefer!   ( xt1 xt2 -- )
>mark     ( -- addr )
>resolve  ( addr -- )
hiemit    (w -- )
int_restore( sreg -- )
<mark     ( -- addr )
<resolve  ( addr -- )
Rdefer@   ( xt1 -- xt2 )
Rdefer!   ( xt1 xt2 -- )
(sliteral)( -- addr n)
spmbuf    (x addr -- )
spmerase  (addr -- )
spmpageload(addr -- )
spmrww    (-- )
spmrww?   (-- )
spmwrite  (spmcsr x addr -- )
Udefer@   ( xt1 -- xt2 )
Udefer!   ( xt1 xt2 -- )
```

unclassified

```
icompare  ; ( addr-ram addr-flash  --  f)
```