## amforth 2.8 Reference Card

Arithmetics		Compiler		Exceptions	
1-	( n1 n2 )	\	( )	abort	( n*x )
1+ 2/	( n1 n2 )	[']	( XT )	- h + !!	R(n*y)
2/ 2*	( n1 n2 ) ( n1 n2 )	code	( )	abort"	( n*x ) R(n*y)
abs	( n1 u1 )	:	( )	catch	( xt )
><	( n1 n2 )	:noname constant	( xt )	handler	( addr )
cell+	( n1 n2 )	does>	( )	throw	( n )
cells	( n1 n2 )	."	( )		,
d2/	( d1 d2 )	Edefer	( n <name> )</name>	TT 1	<b>A</b>
d2*	( d1 d2 )	else	( addr1 addr2)	Hardwa	are Access
dinvert	( d1 d2)	end-code	( )	rx0	( c)
d-	( d1 d2 d3 )	exit	( )	rx0?	( f)
d+ 	( d1 d2 d3)		R(xt)	>usart0	( )
invert	( n1 n2)	immediate		tx0	(c )
log2 lshift	( n1 n2 ) ( n1 n2 n3)	[	( )	tx0?	( f)
-	( n1 n2 n3 )	literal		+usart0	( )
mod	( n1 n2 n3)	( ]	( ) ( )		
m*	( n1 n2 d)	Rdefer	( n <name> )</name>	IO	
*	( n1 n2 n3 )	recurse	( )		
+	( n1 n2 n3)	s,	( addr len )	refill	( f )
+!	( n addr )	;	( )		
rshift	( n1 n2 n3 )	s"	( <cchar> )</cchar>	Interru	nt
/	( n1 n2 n3)	state	( addr )	menu	.pt
/mod	( n1 n2 rem quot)	then	( addr )	int@	( i xt )
*/	(n1 n2 n3 n4)	until	( addr )	-int	( sreg )
*/mod um/mod	( n1 n2 n3 rem quot)	uboi	( n )	+int	( )
um*	( ud u2 rem quot) ( u1 u2 d)	value	( n <name> )</name>	int!	( xt i )
u/mod	(u1 u2 rem quot)	variable	( )	#int	( n )
u*/mod	(u1 u2 u3 rem quot)	)			
0	( 0 )		l Structure	$\operatorname{Logic}$	
	-	Contro		Logic and	( n1 n2 n3 )
0	( 0 )	Contro	( addr )	_	( n1 n2 n3 ) ( n1 n2 )
	( 0 )	Contro	( addr ) ( addr )	and	( n1 n2 ) ( flag flag' )
0	( 0 )	Control again begin	( addr )	and negate	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 )
° Charac	( 0) eter IO ( 32) ( )	Control again begin do	( addr ) ( addr ) ( addr )	and negate not or	( n1 n2 ) ( flag flag' )
Charace bl cr emit	( 0)  ter IO  ( 32) () (c)	Control again begin do	( addr ) ( addr ) ( addr ) ( n )	and negate not or	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 )
Charace bl cr emit emit?	( 0)  ter IO  ( 32)  ()  (c)  (c)	Control again begin do i	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n )</pre>	and negate not or sys%or	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)
Charace bl cr emit emit? key	( 0)  ter IO  ( 32)  ()  (c)  (f)  (c)	Control	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy</pre>	and negate not or sysyor  MCU s2 loop-	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)
Charace bl cr emit emit? key key?	( 0)  ter IO  ( 32)  ()  (c)  (f)  (c)	Control again begin do i	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sys( )</pre>	and negate not or sysyor  MCU -jtag	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( )
Charace bl cr emit emit? key key? /key	( 0)  ter IO  ( 32)  ()  (c)  (f)  (c)  (f)  ()	Control again begin do i if j	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys)</pre>	and negate not or sysyor  MCU -jtag -wdt	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( )
Charace bl cr emit emit? key key? /key space	( 0)  eter IO  ( 32)  ()  (c)  ( f)  ( c)  ( f)  ()	Control again begin do i if j leave	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr )</pre>	and negate not or sys%or  MCU -jtag -wdt sleep	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( )
Charace bl cr emit emit? key key? /key space spaces	( 0)  eter IO  ( 32)  ()  (c)  (f)  (c)  (f)  ()  ()	Control again begin do i if j leave loop +loop	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr )</pre>	and negate not or systor  MCU -jtag -wdt sleep spirw	<pre>( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ) ( txbyte rxbyte)</pre>
Charace bl cr emit emit? key key? /key space	( 0)  eter IO  ( 32)  ()  (c)  ( f)  ( c)  ( f)  ()	Control again begin do i if j leave loop +loop ?do	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr )</pre>	and negate not or sys%or  MCU -jtag -wdt sleep	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( )
Charace bl cr emit emit? key key? /key space spaces	( 0)  eter IO  ( 32)  ()  (c)  (f)  (c)  (f)  ()  ()	Control again begin do i if j leave loop +loop	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr )</pre>	and negate not or sys%or  MCU -jtag -wdt sleep spirw wdr	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( ) ( txbyte rxbyte) ( )
Charace bl cr emit emit? key key? /key space spaces type	( 0)  ter IO  ( 32)  ()  (c)  (f)  (c)  (f)  ()  (n)  (addr n)	Control again begin do i if j leave loop +loop ?do repeat	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 )</pre>	and negate not or systor  MCU -jtag -wdt sleep spirw	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( ) ( txbyte rxbyte) ( )
Charace bl cr emit emit? key key? /key space spaces	( 0)  ter IO  ( 32)  ()  (c)  (f)  (c)  (f)  ()  (n)  (addr n)	Control again begin do i if j leave loop +loop ?do repeat	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 ) ( )</pre>	and negate not or sys%or  MCU -jtag -wdt sleep spirw wdr	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( ) ( txbyte rxbyte) ( )
Charace bl cr emit emit? key key? /key space spaces type  Compa	( 0)  ter IO  ( 32) () (c) (f) (c) (f) () (n) (addr n)	Control again begin do i if j leave loop +loop ?do repeat unloop	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 ) ( ) ; R(loop-sys )</pre>	and negate not or systor  MCU -jtag -wdt sleep spirw wdr  Memor	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3) sys1 loop-sys2) ( ) ( ) ( txbyte rxbyte) ( )
Charace bl cr emit emit? key key? /key space spaces type  Compa	( 0)  ter IO  ( 32) () (c) (f) (c) (f) () () (n) (addr n)  re  (d1 d2 flag) (d1 d2 flasg)	Control again begin do i if j leave loop +loop ?do repeat unloop while	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 ) ( ) ; R(loop-sys ) ( dest orig dest )</pre>	and negate not or systor  MCU -jtag -wdt sleep spirw wdr  Memor	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ) ( txbyte rxbyte) ( )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d> d< =	( 0)  ter IO  ( 32) () (c) (f) (c) (f) () () (n) (addr n)  re  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 ) ( ) ; R(loop-sys ) ( dest orig dest )</pre>	and negate not or sys%or  MCU -jtag -wdt sleep spirw wdr  Memor c@ cmove>	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ) ( txbyte rxbyte) ( )  y ( addr - c1 ) ( addr-from addr-to n ) ( c addr ) ( addr - n)
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0=	( 0)  ter IO  ( 32) () (c) (c) ( f) ( c) ( f) () (n) (n) (addr n)  Te  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   ( addr )   (addr1 addr2 )   ( )   ; R(loop-sys )   ( dest orig dest )</pre>	and negate not or sys%or  MCU s2loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e!	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  y ( addr - c1 ) ( addr-from addr-to n ) ( c addr ) ( addr - n) ( n addr )
Charace bl cr emit emit? key key? /key space spaces type  Compadd d d d = 0 = >	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (n) (addr n)  Te  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers	<pre>( addr ) ( addr ) ( addr ) ( n ) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr ) (addr1 addr2 ) ( ) ; R(loop-sys ) ( dest orig dest )</pre>	and negate not or sys%or  MCU s2loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ( ) ( ( ) ( txbyte rxbyte) ( )  Ty  ( addr - c1 ) ( addr-from addr-to n ) ( c addr ) ( addr - n) ( n addr ) ( addr n )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0=	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (n) (addr n)  Te  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n1 n2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   ( addr )   (addr1 addr2 )   ( )   ; R(loop-sys )   ( dest orig dest )</pre> <pre>Sion</pre> ( d1 n1 )	and negate not or sys%or  MCU s2 loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  y ( addr - c1 ) ( addr-from addr-to n ) ( addr - n) ( n addr ) ( addr n) ( c-addr u c )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0>	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (n) (addr n)  Te  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   ( addr )   (addr1 addr2 )   ( )   ; R(loop-sys )   ( dest orig dest )</pre> <pre>Sion</pre> ( d1 n1 )	and negate not or sys%or  MCU s2 loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  ( dddr - c1 ) ( addr - c1 ) ( addr - n) ( n addr ) ( addr n) ( c-addr u c ) ( addr n1 )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0> <	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (addr n)   re  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n1 n2 flag) (n1 flag) (n1 flag) (n1 flag) (n1 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s s>d	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   (addr 1 addr 2 )   ( )   ; R(loop-sys )   ( dest orig dest )</pre> <pre>Sion</pre> ( d1 n1 )   ( n1 d1 )	and negate not or sysyor  MCU -jtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@ i!	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ) ( txbyte rxbyte) ( ) ( addr - c1 ) ( addr - from addr-to n ) ( c addr ) ( addr - n) ( n addr ) ( addr n ) ( c-addr u c ) ( addr n1 ) ( n addr )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0> < 0<	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (addr n)   Te  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   (addr 1 addr 2 )   ( )   ; R(loop-sys )   ( dest orig dest )</pre> <pre>Sion</pre> ( d1 n1 )   ( n1 d1 )	and negate not or sys%or  MCU s2 loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3)  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  ( addr - c1 ) ( addr - c1 ) ( addr - n) ( n addr ) ( addr n) ( c-addr u c ) ( addr n1 )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0> < 0< max	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (addr n)   re  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag)	Control again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s s>d	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   (addr 1 addr 2 )   ( )   ; R(loop-sys )   ( dest orig dest )  sion   ( d1 n1 )   ( n1 d1 )</pre>	and negate not or sys%or  MCU s2loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@ i! !	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3 )  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  ( txbyte rxbyte) ( )  ( addr - c1 ) ( addr-from addr-to n ) ( addr - n) ( n addr ) ( addr n ) ( c-addr u c ) ( addr n1 ) ( n addr ) ( n addr ) ( n addr ) ( n addr )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0> < 0< max min <> 0<>	( 0)  ter IO  ( 32) () (c) (cf) (c) (f) (c) (n) (n) (addr n)   re  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag) (n1 n2 flag) (n1 n2 n1   n2) (n1 n2 flag)	again begin do i if j leave loop +loop ?do repeat unloop while  Convers d>s s>d  Diction , compile	( addr ) ( addr ) ( addr ) ( n) ; R( loop-sys loop- ( addr ) ( n ) ; R( loop-sys1 loop-sy ( ) R(loop-sys) ( addr ) ( addr ) ( addr 1 addr 2 ) ( ) ; R(loop-sys ) ( dest orig dest )   Sion ( d1 n1 ) ( n1 d1 )   ary ( n ) ( )	and negate not or sysyor  MCU -jtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@ i!	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3 )  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  ( dddr - c1 ) ( addr - c1 ) ( addr - n) ( n addr ) ( addr n ) ( c-addr u c ) ( addr n1 ) ( n addr ) ( n addr ) ( n addr ) ( n addr )
Charace bl cr emit emit? key key? /key space spaces type  Compa d> d< = 0= > 0> < 0< max min <>	( 0)  ter IO  ( 32) () (c) (c) (f) (c) (f) () (n) (addr n)   re  (d1 d2 flag) (d1 d2 flag) (n1 n2 flag)	again begin do i  if j leave loop +loop ?do repeat unloop while  Convers d>s s>d  Diction	<pre>( addr )   ( addr )   ( addr )   ( n )   ; R( loop-sys loop-   ( addr )   ( n )   ; R( loop-sys1 loop-sy   ( )   R(loop-sys)   ( addr )   ( addr )   (addr 1 addr 2 )   ( )   ; R(loop-sys )   ( dest orig dest )  sion   ( d1 n1 )   ( n1 d1 )</pre>	and negate not or sys%or  MCU s2loopjtag -wdt sleep spirw wdr  Memor  c@ cmove> c! e@ e! @ fill i@ i! !	( n1 n2 ) ( flag flag' ) ( n1 n2 n3 ) ( n1 n2 n3 )  sys1 loop-sys2) ( ) ( ) ( ( ) ( txbyte rxbyte) ( )  ( dddr - c1 ) ( addr - c1 ) ( addr - n) ( n addr ) ( addr n ) ( c-addr u c ) ( addr n1 ) ( n addr ) ( n addr ) ( n addr ) ( n addr )

```
Numeric IO
                                   System
                                                                      internal/hidden
           ( -- addr )
 base
                                              ( addr n1 -- n2 )
                                    accept
                                                                        (branch) (-- )
           ( -- )
 decimal
                                              ( n -- )
                                                                        (?branch) (f -- )
                                    allot
           ( c base -- number flag )cold
                                              ( -- )
 digit
                                                                        (constant)(-- addr )
                                              ( xt1 -- xt2 )
           ( n -- )
                                    defer@
                                                                        (create) (-- )
           ( -- )
                                    defer!
                                              ( xt1 xt2 -- )
                                                                        (do)
                                                                                 (limit counter -- )
                                              ( xt -- )
 hld
           ( -- addr )
                                    execute
                                                                                 R(-- limit counter )
 hold
           ( c -- )
                                              ( -- f_cou )
                                                                                 (-- )
                                    f_cpu
                                                                        (does>)
           ( -- )
                                              ( -- addr )
 <#
                                    >in
                                                                                 (i*x -- j*x )
                                                                        (defer)
           (addr -- n )
( d1 -- d2)
 number
                                    interpret ( -- )
                                                                        (literal) (-- n1 )
 #
                                              ; R(i*x - j*x )
                                                                                 (-- )
                                                                        (loop)
           ( d1 -- addr count )
 #>
                                              ( xt1 c<char> -- )
                                    is
                                                                                 R(limit counter -- limit counter+1|)
           ( d1 -- 0)
 #s
                                    #tib
                                              ( -- addr )
                                                                                 (n1 -- )
                                                                        (+loop)
           ( n -- )
( n -- )
 sign
                                    ?execute
                                              ( xt|0 -- )
                                                                                 R(llimit counter -- limit counter+n1|)
                                                                                 (limit counter -- )
                                              ( -- )
 u.
                                    quit
                                                                        (?do)
                                              ( -- addr n )
                                    source
                                                                                 R(-- limit counter| )
                                              ( -- addr )
                                                                                      .dw XT_FETCH
                                    Ogu
                                                                        (rp0)
Stack
                                              ( addr -- )
                                    up!
                                                                                     .dw XT_EXIT
                                                                                 ( -- addr)
                                                                        (sp0)
 depth
           ( -- n )
                                                                                 (spmcsr x addr -- )
                                   System Pointer
                                                                        (spm)
           ( n -- )
 drop
                                                                                 ( n -- )
R(IP -- IP+1)
                                                                        (to)
           ( n -- n n )
 dup
                                              ( -- eaddr)
                                    dр
           ( n1 n2 -- n1 n2 n1 )
 over
                                              ( -- eaddr)
                                    edp
                                                                        (user)
                                                                                 (-- addr )
 ?dup
           ( n1 -- [ n1 n1 ] | 0)
                                              ( -- eaddr)
                                                                        (variable)(-- addr )
                                    head
           ( n1 n2 n3 -- n2 n3 n1)
 rot
                                              ( -- eaddr)
                                    heap
                                                                                ( xt1 -- xt2 )
                                                                       Edefer@
 r@
           ( -- n)
                                              ( -- addr )
                                    here
                                                                        Edefer!
                                                                                ( xt1 xt2 -- )
           R(n -- n)
                                              ( -- addr )
                                    pad
                                                                       >mark
                                                                                 ( -- addr )
           ( -- n )
 r>
                                              ( -- addr )
                                    tib
                                                                       >resolve ( addr -- )
           ; R( n --)
                                    turnkey
                                              ( -- eaddr)
                                                                       hiemit (w -- )
           ( n1 n2 -- n2 n1)
 swap
                                                                       int_restor(e sreg -- )
           ( n -- )
 >r
                                   System Value
                                                                        <mark
                                                                                ( -- addr )
           ; R( -- n)
                                                                        <resolve ( addr -- )
                                    baud0
                                              ( -- v)
                                                                       Rdefer@ ( xt1 -- xt2 )
Stackpointer
                                                                       Rdefer!
                                                                                 ( xt1 xt2 -- )
                                                                        (sliteral)( -- addr n)
                                   Time
                                                                        spmbuf (x addr -- )
           ( -- addr)
 rp0
                                    1ms
                                              ( -- )
                                                                        spmerase (addr -- )
           ( -- n)
 rp@
           ( n -- )
                                                                       spmpageloa@addr -- )
 rp!
           ; R( -- xy)
                                                                        spmrww
                                                                                 (-- )
                                   Tools
                                                                       spmrww?
                                                                                 (-- )
           ( -- addr)
                                    [char]
                                              ( -- c )
                                                                        spmwrite (spmcsr x addr -- )
 sp0
           ( -- addr)
                                              ( -- c )
           ( -- n)
                                    char
                                                                       Udefer@ ( xt1 -- xt2 )
 sp@
                                              ( -- )
                                    .s
                                                                       Udefer! ( xt1 xt2 -- )
 sp!
           ( addr -- i*x)
                                              ( addr -- [ addr 0 ] | [ xt [-1|1]] )
                                    find
                                    icompare ( addr-ram addr-flash -- f)
String
                                    icount
                                              ( adr -- adr n )
                                              ( addr n -- )
                                    itype
 count
           ( addr -- addr+1 n)
                                              ( -- )
                                    noop
           ( addr1 n1 c -- addr1 n2 t)o
 cscan
                                              (n < name > --)
           (addr1 n1 c -- addr2 n2 wnused
                                              ( -- n )
 cskip
           ( char "ccc" -- c-addr u ver
                                              ( -- )
 parse
 place
           ( addr1 len1 addr2 -- ) word
                                              ( c -- addr )
```

( addr1 u1 n-- addr2 u2 )words

/string