R3 CheatSheet - https://github.com/phreda4/ - PHREDA

Block construction			**Nameless definition**		
(		Start block for IF or WHILE	[		Start nameless definition
)		End block for IF or WHILE	]	| – v	End nameless definitions
Control flow					
;		End of Word	EX	| v --	Run a word from address
Conditional					
0?	| a -- a	is TOS=Zero? conditional	1?	| a -- a	is TOS<>Zero? conditional
+?	| a -- a	is TOS>=0?	-?	| a -- a	is TOS<0?
<?	| a b -- a	is a<b? remove TOS	>?	| a b -- a	is a>b? remove TOS
=?	| a b -- a	is a=b? remove TOS	>=?	| a b -- a	is a>=b? remove TOS
<=?	| a b -- a	is a<=b? remove TOS	<>?	| a b -- a	is a<>b? remove TOS
AND?	| a b -- c	is a AND b? remove TOS	NAND?	| a b -- c	is a NAND b? remove TOS
BT?	| a b c -- a	is a<=b<=c? remove TOS			
Stack movements					
DUP	| a – aa	duplicate TOS	DROP	| a --	remove TOS
OVER	| ab -- aba	duplicate Second of Stack	PICK2	| abc -- abca	Pick 3 element
PICK3	| abcd -- abcda	Pick 4 element	PICK4	| abcde -- abcdea	Pick 5 element
SWAP	| ab -- ba	swap TOS ans NOS	NIP	| ab -- b	remove NOS
ROT	| abc -- bca	Rotate 3 top element	2DUP	| ab -- abab	Duplicate 2 values of top
2DROP	| ab --	Remove 2 elements	3DROP	| abc --	Remove 3 elements
4DROP	| abcd --	Remove 4 elements	2OVER	| abcd -- abcdab	Copy 2 lower elemenst
2SWAP	| abcd -- cdab	Swap 4 elements			
Return Stack					
>R	| a --	rstack: -- a	R>	| -- a	rstack: a --
R@	| -- a	rstack: a -- a			
Logic operators					
AND	| a b -- c	c=a AND b	OR	| a b -- c	c=a OR b
XOR	| a b -- c	c=a XOR b	NOT	| a -- b	b=NOT a
Aritmetic operators					
+	| a b -- c	d=a+b	-	| a b -- c	d=a-b
*	| a b -- c	d=a*b	/	| a b -- c	d=a/b
<<	| a b -- c	d=a shift left b	>>	| a b -- c	d=a shift rigth b
>>>	| a b -- c	d=a shift rigth b w/o sign	MOD	| a b -- c	d=a mod b
/MOD	| a b -- c d	c=a/b d=a mod b	*/	| a b c -- d	d=a*b/c - not bit loss
*>>	| a b c -- d	d=(a*b)>>c – not bit loss	<</	| a b c -- d	d=(a<<c)/b – not bit loss
NEG	| a -- b	b=-a	ABS	| a -- b	b=|a|
SQRT	| a -- b	b=square root(a)	CLZ	| a -- b	b=count lead zeros of a
Memory fetch and store					
@	| a -- [a]	fetch dword adress	C@	| a -- b[a]	fetch byte from adress
Q@	| a -- q[a]	fetch qword adress	@+	| a -- b [a]	fetch value and increment 4
C@+	| a -- b b[a]	fetch byte and increment 1	Q@+	| a -- b q[a]	fetch qword and increment 8
!	| a b --	store A in adress B	C!	| a b --	store byte A in adress B
Q!	| a b --	store qword A in adress B	!+	| a b -- c	store A in B and inc 4
C!+	| a b -- c	store byte A in B and inc 1	Q!+	| a b -- c	store qword A in B and inc 8
+!	| a b --	increment in mem B, A	C+!	| a b --	increment in mem B, byte A
Q+!	| a b --	increment in mem B, A			
Auxiliary registers					
>A	| a --	load register A	B>	| -- a	push register B
A>	| -- a	push register A	>B	| a --	load register B
A@	| -- a	fetch from A	B@	| -- a	fetch from B
A!	| a --	store in mem A	B!	| a --	store in mem B
A+	| a --	add to A	B+	| a --	add to B
A@+	| -- a	fetch A and increment 4	B@+	| -- a	fetch B and increment 4
A!+	| a --	store in mem A, increment 4	B!+	| a --	store in mem A, increment 4
Memory copy and fill					
MOVE	| d s c --	copy S to D, C dword	MOVE>	| d s c --	copy from S to D, C dword in rev.
FILL	| d v c --	fill D, C dword with V	CMOVE	| d s c --	copy from S to D, C bytes
CMOVE>	| d s c --	copy S to D, C bytes in rev.	CFILL	| d v c --	fill from D, C bytes with V

QMOVE	| d s c --	copy S to D, C qwords	QMOVE>	| d s c --	copy from S to D, C qwords in rev.
QFILL	| d v c --	fill D, C qwords with V			
Operating System					
UPDATE	| --	update SO events	REDRAW	| --	refresh graphic buffer
MEM	| -- a	start memory free	VFRAME	| -- a	frame buffer adress
SH	| -- a	screen height	SW	| -- a	screen width
XYPEN	| -- x y	position of mouse or pen	BPEN	| -- a	key state of mouse or pen
KEY	| -- a	key code	CHAR	| -- a	character ascii code
TIME	| -- a	Hour(8):min(8):sec(8)	DATE	| -- a	Year(16):month(8):day(8)
MSEC	| -- a	milisecond of system	APPEND	| m cnt "fn" --	append file from M, C bytes
LOAD	| m "fn" -- lm	load file in M, last in LM	SAVE	| m cnt "fn" --	save file from M, C bytes
FFIRST	| "f" -- s	get first struct of folder "f"	FNEXT	| a -- s	next struct or 0 to end
SYS	| "sys" --	call SO to run program			
Graphics drawing					
INK	| -- color	value of pen color	'INK	| -- 'ink	adress of color to set
ALPHA	| a --	set alpha value	OP	| x y --	set last point
OPX	| -- opx	last x point	OPY	| -- opy	last y point
LINE	| x y --	lineto	CURVE	| x y x y --	curve cuadratic bezier
CURVE3	| x y x y x y --	curve qubic bezier	PLINE	| x y --	lineto polygon
PCURVE	| x y x y --	curve cuadratic bezier poly	PCURVE3	| x y x y x y --	curve qubic bezier polygon
POLI	| --	fill polygon			
Sound and Music					
SLOAD	| "fn" -- s	Load sound, stack adr	MLOAD	| "fn" -- m	Load music, stack adr
SFREE	| s --	Free sound with adr	MFREE	| m --	Free music with adr
SPLAY	| s --	Play sound, 0 stop	MPLAY	| m --	Play music, 0 stop
Video Playback (r3v version only)					
VIDEO	| "fn" w h --	0 close video	VIDEOSHOW	| w h -- v	
VIDEOSIZE	| w h --				

Prefix	
:	define CODE, :: Export word
#	define DATA, ## Export word
^	Include source code in filename
'	Adress of word, code or data
|	Commento to end of the line
“	String to next “, “” for “ character
$	Hex numbers
%	Binary numbers, 0 can be .

Data Definition	
dword	#var 0
dword list	#list 1 2 3 4 5
byte list	#blist (1 2 3 4)
memory	#buffer * 1024 | 1kb size
vectors	#vector ‘actionword
list jump	#listj ‘a1 ‘a2 ‘a3

Control Flow	
REPEAT	(loop)
IF	?? (true branch)
WHILE	(while ?? loop)
MULTI WHILE	(while ?? while ?? loop)
IF-ELSE	factoring to new word :ifelse ?? (true ;) false ;

Comment work like option switchs	
|WIN|	in win, the line is not a comment
|LIN|	in linux,…
|WEB|	In web,…
|MAC|	In MAC,…
|RPI|	In Raspberry Pi,…
|FULL	set fullscreen mode
|SCR 640 480	screen or window size
|MEM 640	data memory size (in kb) min 1kb