Caster Quick Reference

1. Numbers

numb [minus]<numbers>[f] ..... **print digits**
word numb <0...9> ..... **print as words**

2. Alphabet[a]

alpha	golf	mike	santa	yankee
bravo	hotel	nova	tango	zulu
carla	india	oscar	utah	
delta	julia	papa	vespa	
echo	kilo	quebec	whiskey	
foxy	lima	romeo	x-ray	

3. Punctuation[d]

(shin) tab ..... (SHIFT-)TAB
ace ..... SPACE
quote [f] ..... "
chicky [f] | apostrophe '
ticky [f] ..... `
par [bg] ..... ()
brax [bg] ..... []
angle [bg] ..... <>
curly [bg] ..... {}
comma [ce] ..... ,
dot | period [c] ..... .
underscore ..... _
dunder ..... __
pipe (sim | symbol) .. |
bang ..... !
question ..... ?
[f] plus [deh] ..... +
[f] minus [deh] ..... -
star | times [dh] ..... *
slash | divide [dh] ..... /
[f] backslash ..... \
equals [de] ..... =
atty | at symbol ..... @
hashtag ..... #
dollar ..... $
ampersand ..... &
mod [deh] ..... %
colon ..... :
semicolon ..... ;
carrot ..... ^
tilde ..... ~

4. Comparisons

greater than[d] ..... >
less than[d] ..... <
equal to[d] ..... ==
greater eq[d] ..... >=
less eq[d] ..... <=
not eq[d] ..... !=

[a] Prefix with "big" for capitals
[b] Left key is pressed after
[c] Space bar is pressed after
[d] Space inserted before and after
[e] Prefix with "short" to not insert space(s)
[f] Prefix with "double" to insert two of these characters
[g] Left or right available by prefixing with "open" or "close"
[h] Suffix with "eq" (pronounced "eek") to append '='

5. Directions

up [d] ..... **up**
down [d] ..... **down**
left [d] ..... **left**
right [d] ..... **right**

6. Editing

[c] clear [d] ..... **bkspc**
[c] deli [d] ..... **del**
enter [d] ..... **enter**
cancel ..... **escape**
save ..... **ctrl - s**

7. Selection, Movement

shackle ..... **select line**
shin [a b] ..... **shift - <dir>**
queue [a b] ..... **ctrl - shift - <dir>**
fly [a b] ..... **ctrl - <dir>**
[c] home ..... **home**
[c] end ..... **end**
kraken ..... **ctrl + space**
hug <enclose> [e] ..... **encloses target**

8. Copy and Paste

copy [c] ..... **copy**
cut [c] ..... **cut**
paste [c f] ..... **paste**
grab [c] ..... **double-click + copy**
drop [c] ..... **double-click + paste**
duplicate [b] ..... **duplicate line**

[a] Takes optional direction parameter, one of the direction keys from table (5)
[b] Takes optional number parameter for repetition
[c] Prefix with "big" for ctrl+
[d] Repeatable by saying number after
[e] Takes par, brax, curly, angle, quote, chicky, ticky
[c] Takes optional number parameter, which is used to determine a persistent clipboard slot
[f] Takes optional formatting parameters, e.g. "paste title snake"

9. Text Formatting

Capitalisation

yell ..... **SOME WORDS**
title [b] ..... **SomeWords**
camel [b] ..... **someWords**
sing ..... **Some words**
laws ..... **some words**

Spacing

ace (default) ..... **some words**
gum | gun ..... **somewords**
kebab ..... **some-words**
snake [b] ..... **some_words**
pebble ..... **some.words**
incline ..... **some/words**
dissent | descent ..... **some\words**

Capitalisation and spacing can be combined into a single command. They may be suffixed by "bow" and then dictation to format (ex: "title snake bow some words") to aid recognition.
Must pause after issuing command. This is not a CCR/merge command.

10. Free Dictation - Engine Formatting [c]

say ..... **default engine formatting**
cap ..... **Initial letter capitalized**
slip ..... **initial letter lowercase**

[c] Pass-through text formatting as defined by speech recognition engine.
[b] title and camel default to gum, snake to laws

11. Python (Visual Studio Code)

enable Python ..... **activate module**

bit shift (left|right)
bitwise and ..... `&`
bitwise or ..... `|`
bitwise complement . `~`
bitwise xor ..... `^`
logical and ..... `and`
as
break
class
class method
toggle comment
comment
big comment
continue
pie deli ..... `del`
derived class
elif
else
except
false
finally

for
from
function
global
if
in
import
is
method
none
not
logical or ..... `or`
pass
raise
return
static method
true
try
while
with
short with

delete line
expand [a]
shrink [a]
fold [all]
unfold [all]
scroll [page] up [a]
scroll [page] down [a]
hover
IntelliSense
line <n>
(go|peek|side) definition
peek references
go symbol
go bracket
show command palette

search for file
(next|previous) tab [a]
close tab [a]
split editor
close editor
(next|previous) editor [a]
(hide|show) explorer
focus (explorer|editor)
terminal
debug
debug continue
debug stop
breakpoint
step (in|out|over) [a]
lint

[a]Takes optional number parameter for repetition

12. C++ (Visual Studio)

enable C plus plus ..... **activate module**

bit shift (left|right)
bitwise and ..... `&`
bitwise complement . `~`
logical or ..... `||`

not ..... `!`
bitwise or ..... `|`
bitwise xor ..... `^`
logical and ..... `&&`

auto
bool
break
catch
class
concept
const
const cast
continue
default
delete
do while
elif
else
enum
static assert
define [derived] class

explicit
export
extern
false
final
float
for
range for
friend
if
import
inline
int
long
module

mutable
namespace
new
noexcept
operator
override
private
protected
public
requires
signed
static
static cast
struct
switch
reinterpret cast
dynamic cast

this
throw
todo
true
try
typename
unsigned
using
virtual
void
volatile
while

terminate
toggle comment
[big] comment
delete line
hover
expand [a]
shrink [a]
go bracket
select bracket
[un]fold all
[toggle] [un]fold
fold to definitions
scroll (up|down) [a]
line <n>

(go|peek) definition
find references
quick actions
go (file|type)
go (member|symbol)
(next|previous) issue
(next|previous) tab [a]
close tab [a]
keep tab
search explorer
add new folder
add new header
add new source file

[a]Takes optional number parameter for repetition

arrow ..... `->`
pointer | de-reference ..... `*`
address | reference ..... `&`
double reference ..... `&&`
scope ..... `::`
align as ..... `alignas()`
align of ..... `alignof()`
case ..... `case:`
car ..... `char`
const eval ..... `consteval`
const expression ..... `constexpr`
const init ..... `constinit`
declared type ..... `decltype()`
defined ..... `defined()`
dub ..... `double`
return ..... `return_`
short return ..... `return;`
null pointer ..... `nullptr`
type short ..... `short`
standard ..... `std::`
template ..... `template<>`
type ID ..... `typeid()`
shared pointer ..... `shared_ptr<>`
unique pointer ..... `unique_ptr<>`
weak pointer ..... `weak_ptr<>`
hash (include|if|elif|else|endif|define|pragma) `#(opt.)`
hash un-define ..... `#undef`

(focus|show) (bookmarks | call hierarchy | class view | error list | explorer | output | test explorer)

configuration <n>
platform <n>
format
bookmark
next bookmark [a]
previous bookmark [a]
analyze (file|project|solution)

build [solution]
(run | debug)
debug continue
debug stop
breakpoint
step (in|out|over) [a]

[a]Takes optional number parameter for repetition

13. Mouse Replacement

kick ..... **left click**
kick mid ..... **middle click**
psychic ..... **right click**
shift right click ..... **shift right click**
shift click ..... **shift click**
squat ..... **left button down**
bench ..... **left button up**
lean ..... **right button down**
hoist ..... **right button up**
colic ..... **control left click**
scree [a] ..... **scroll wheel**
curse ..... **move cursor by pixels**
douglas ..... **move cursor on grid**
rainbow ..... **alternate grid**
legion ..... **smart text selection**

14. Window Management

minimise win ..... **minimise window**
maximise win ..... **maximise window**
switch window ..... **ctrl+alt+tab. Use tab|shin+tab, enter**
show windows **(Dragon) show list of windows**
move window ..... **move with arrow keys**
window (left|right) . **snap window to left|right**
send monitor (left|right) ..... **send the current window to left/right monitor**
show work ..... **show active workspaces**
new work ..... **create a new workspace**
close work ..... **close the current workspace**
close all work ..... **close all workspaces**
previous work [b] **go to the previous workspace**
next work [b] ..... **go to the next workspace**
go work <n> ..... **go to workspace number n**
send work <n> . **send the current window to workspace n**
move work <n> **move the current window to workspace n**

15. Repetition

again do ..... **repeat last dictation**
again <n> times ..... **repeat last dictation*n**

16. Dragon

(command | dictation | normal | spell | numbers) mode on ..... **switch dragon modes (native dragon commands)**
scratch that ..... **undo previous dictation**
sratch <n> ..... **undo previous n dictations**
fix dragon double . **delete initial dictated char**

[a] Takes optional direction parameter, one of the direction keys from table (5)
[b] Takes an optional number parameter for repetition