Creatures CAOS Guide

Original Text By Mark Ashton

TABLE OF CONTENTS

Object Pointer Operands

```
TARG - retn curr targ object* as integer
OWNR - default object (owner of script, or pet if DDE)
FROM - obj who caused event leading to this script
NORN - current pet creature
PNTR - pointer object
ATTN - IT - obj that OWNR creature is attending to (may be NULL)
       NOTE: only OWNR's IT can be determined, not TARG's
CARR - object that's carrying OWNR (may be NULL)
EXEC - object who EXECuted the tool who owns this
       return (int)Exec; dde macro. NOTE: only valid for DDE
       tools who *know* that they were executed by an object
_IT_ - obj that Owner creature was attending to
EDIT - the contents of the EditObject variable (addr of object being
       placed/repositioned/deleted; EditObject is set by the EDIT macro
       or by shift-clicking an object. Use this rvalue to delete
       selected objects, etc.
OBJP - a pointer to objects that will survive
TOKN XXXX - convert 4 characters into an integer
       eg. TOKN 1234 = integer '4321'
```

System Operands

```
SNDS - sound status
       Bit 0 = Sound on/off
       Bit 1 = Sound mode (foreground only\continuous)
WINW - max allowed view window width (WORLD coords)
WINH - max allowed view window height (WORLD coords)
```

TARG Object Operands

```
ATTR - obj's attributes (INVISIBLE, CARRYABLE, etc)
```

Values for ATTR

Carryable	creature can pick up obj	**1**
Mousable	mouse can pick up obj	**2**
Activateable	can be activated with mouse	**4**
Container	carries other objs (vehicles only)	**8**
Invisible	creatures cant see it	**16**
Floatable	normally floating on screen	**32**
Wallbound	limits movement to current room	**64**
Groundbound	movement only limited by ground surface	**128**

NOTE: Wallbound OR Groundbound, can’t be both.

```
POSL/POSR/POST/POSB - retn obj's lrtb coords
WDTH/HGHT - retn obj's width/height
LIML/LIMT/LIMR/LIMB - retn obj's limits
CLAS - family+genus+sp (Classifier)
FMLY - family (in range 0-255)
GNUS - genus (in range 0-255)
SPCS - species (in range 0-255)
MOVS - MovementStatus (FLOATING, MOUSEDRIVEN, etc)
ACTV - Object's Active flag (INACTIVE=0 ACTIVE1 ACTIVE2)
```

NEID - obj's neural ID# 0-39

POSE - TARG obj's (and curr Part's) current pose

TARG CompoundObject, Vehicle, Lift and Aircraft Operands

XVEC - vehicle's x mvt vector in 1/256ths pixel

YVEC - vehicle's y mvt vector in 1/256ths pixel

BUMP - vehicle's collision data (bitflags)
b0=hit left b1=hit right b2=top b3=bottom

TARG Creature Operands

DRIV *n* - state of creature's Drive# n (hunger etc)

DRV! - creature's MOST PRESSING Drive# retns 0 (pain) if no drives
pressing Can use in: "DOIF DRIV DRV! GT 128" to test level of
strongest drive

CHEM *n* - concentration of a chemical in

SCOR - return scores stored in score.cpp -- Alima

HOUR - return the number of hours elapsed since game started

MINS - return the minutes component of time elapsed

BABY - moniker of child genome if TARG is pregnant
Useful to modify scripts for pregnant norns

ASLP - return 1 if creature is asleep

CAMN - Creatures age in mins (abus)

CAGE - Creatures age (0-7)

DEAD - Creature is dead

Environmental Operands

WIND - wind speed/dir near TARG obj (-3 to +3)

TEMP - air temperature near TARG obj (-3 to +3)

ROOM *roomnumber edge*
return world l,t,r,b or Type of given room
where "edge" = 0=l 1=r 2=t 3=b
or "edge" = 4 returns room Type (INDOORS...)
Returns 0 if no such room

RMS# - number of rooms defined on map

GND# - number of ground level data on map

GNDW - number of pixels per ground datum

GRND *x* - ground level at position x (worldx/GROUNDW)

TOTL *family genus species*
returns the number of objects in the world who fit this
description. Family, Genus and/or Species can be zero
to act as wildcards. Examples:- setv totl 4 2 0 ;retns # grendels

Truth Test Operands (Returns 1 if true, 0 if false)

```
TOUC objptr1 objptr2 - return 1 if these
       two objects are in contact, eg. DOIF TOUC TARG OWNR GT 0
       means do if ownr and targ are touching
```

TARG Object Operands

MOVS - MovementStatus (FLOATING, MOUSEDRIVEN, etc)
CLAS *family+genus+species+0*
ATTR - obj's Attributes bits
OBJP - a pointer to objects that will survive serialisation.

TARG CompoundObject, Vehicle, Lift and Aircraft Operands

XVEC - vehicle's x mvt vector in 1/256ths pixel
YVEC - vehicle's y mvt vector in 1/256ths pixel

Set Activity State

ACTV - Object's Active flag (INACTIVE=0 ACTIVE1 ACTIVE2)

Targ Creature Operands

BABY - set to 0 to abort a pregancy (or set to child moniker to make her pregnant)

System Operands

WINW - max allowed view window width (WORLD coords)
WINH - max allowed view window height (WORLD coords)
NORN - set current pet creature

DDE: SCRP *family genus species event*
fetch a script from the scriptorium and send it (used by script editor for reading out & editing existing scripts

DDE: PUTV *RValue*
Send an integer Rvalue

DDE: PUTS *[literal string]*
Send a string - useful for debugging macros, or for returning the results of macro commands to test the truth of some condition

DDE: GETB *'option'*
get buffer
gets string and writes to dde buffer

dde: getb data
get all creatures data

dde: getb cnam

get creature's name

dde: getb ctim
get time creature has been alive

dde: getb monk
get creature's moniker

dde: getb ovvd
returns the following fields (each seperated by a "|" symbol)for every creature (where creatures are seperated by a "&" symbol).

```
Name
Moniker
Sex          (either "1" or "2") 1=male 2=female
Age          (in "hours:mins")
Pregnancy    (either "N/A", "No" or <number>)
Life-Force   (either <number> terminated in % or
                                         "Dead")
Medical      (either "Healthy", "Sick" or "Dead")
Room         (number of room they're in)
Xpos
Ypos
```

DDE: PUTB *[literal string] 'option'*
write from string to location determined by option token

dde: putb *[literal string]* **data**
set all creatures detaills

dde: putb *[literal string]* **cnam**
set the creature's name from the string

DDE: PICT - take snapshot of the current subject create a standard windows bmp pass file name back to client

DDE: NEGG - Update Number of Natural eggs in world

DDE: HATC - Update Number of Norns in world if egg hatches voluntarily

DDE: LIVE - Update Number of Norns in world if egg hatches voluntarily

DDE: DIED - Update Number of Norns in world if egg hatches voluntarily

DDE: PANC - Alima simple macro to pan camera to creature before the owners kit takes a photo

DDE: LOBE - output the locations of the brain lobes of the subject of the macro format is " 'x_start'y_start'width'height' " after a leading count of the number of lobes based on the 64x48 grid of neurones

DDE: GENE - Output the numbers of each of the 12 types of genes

DDE: WORD *index* - read a word/idea from targ BLACKBOARD's list. Sends

"###|text|", where ### is the vocabulary slot (WD_xxx) for the idea represented by the bbd picture whose index is Index, and 'text' is the word associated with that picture Used by blackboard editor tools to fetch words for editing See "WORD" cmd for writing words into object

DDE: CELL *lobe cell dentype*
Get statistics about this neurone. Used by brain debug/analysis tools. Stores the following data in buffer: Output | State | number-of-dens-of-that-type | total Susceptibility | total STW | total LTW | total Strength | The dendrite values are totalled from all dendrites of the given type in that cell - the magnitude will vary according to the number of dendrites, which is given in the returned string (so that gauges and graphs can be scaled appropriately, or mean values calculated).

carry out a '**sys:**' command to control the system (windows, menus, quitting, etc.)

SYS: loading and saving

QUIT - Saves world & closes Vivarium
THIS MUST BE THE ONLY/LAST COMMAND IN THE MACRO
ABRT - Abandons changes to world & closes Vivarium
THIS MUST BE THE ONLY/LAST COMMAND IN THE MACRO
WRLD *[filename.viv]* - Opens a new document (world) after saving the current one (if any)
THIS MUST BE THE ONLY/LAST COMMAND IN THE MACRO

SYS: menu commands

CMND *id#* - issue an ID_XXX command message to the application. This allows macros to activate ANY menu command. Note that command will get executed LATER - fn doesn't wait before returning!
id# is the decimal ID_XXX value - look these up in the resource file & list them for users

camera, window and scrolling control

WPOS *x y width height* - attempt to position vivarium frame window to this size (in pixels) Actual size will be limited to maximum view size or size of screen, if neces

SYS: WTOP - Set vivarium's window to be foreground window (useful in editor tools etc to allow user access to vivarium for selecting objects etc)

SYS: EDIT *l t r b*
Set CDisplay::EditBox, so that a rectangle is drawn on screen at the given WORLD coordinates. Use "SYS: EDIT 0 0 0 0" to remove the box when finished. This macro is used by map editors and suchlike to mark out rooms and floor levels during map construction

SYS: CMRA *x y* - Disconnect camera from logged-on creature & position it at these world coordinates (eg. when editing map etc.)

SYS: CAMT - moves camera to point at current TARG

SYS: GRND *x y* - set ground level at position x (worldx/GROUNDW) (see GND# and GNDW macros for establishing useful constants)

// carry out a 'new:' command to create a new object of given type
// The 'new:' prefix has been read, so read the next token to determine what type of object to create.
// NOTE: These commands change the TARG object to that which has just been created, so that any further commands in the script refer to the new object and can thus be used to alter other member variables as required.
// After creating, use EDIT macro to allow user to position object (unless object was created by another object on the fly)

NEW: SCEN *imagefile numimages imagenumber plane*
Create a scenery object
- imagefile is a 4-byte token representing the filename of the image file
- numimages is the TOTAL number of images IN THAT FILE
- imagenumber is the image associated with this object
- plane is the plot plane (0=back, 9000=front)
example: new: scen SCN1 37 3 9000

NEW: SIMP *imagefile numimages imagenumber plane clone*
Create a SimpleObject
- imagefile is a 4-byte token representing the filename of the image file
- numimages is the number of images BELONGING TO THIS OBJECT
- imagenumber is the offset of the first image associated with this object
- plane is the plot plane (0=back, 9000=front)
- clone is 0 normally, or 1 to create a cloned image gallery. example: new: simp TOYS 3 19 7000 0

Default object has these properties:-

```
attributes: none
classifier: SIMPLE, no genus or species
behaviour:  dumb (no mouse or creature
                        activation)
events:           no scripts
animation:  none
```

ALL THESE VALUES MAY NEED TO BE SET BY FURTHER MACRO COMMANDS

NEW: CBTN *imagefile numimages imagenumber plane*
Create a CallButton object
- imagefile is a 4-byte token representing the filename of the image file
- numimages is the number of images BELONGING TO THIS OBJECT
- imagenumber is the offset of the first image associated with this object
- plane is the plot plane (0=back, 9000=front)
example: new: cbtn LIFT 2 19 7000

NEW: COMP *imagefile numimages imagenumber clone*

Create a CompoundObject
- clone is 0 normally, or 1 to create a cloned image gallery. example: new: comp ENGN 3 19 0
Default object has these properties:-

```
          attributes: none
          classifier: COMPOUND, no genus or species
          parts:           none
          hotspots:   none
          events:          no scripts
```

ALL THESE VALUES MAY NEED TO BE SET BY FURTHER MACRO COMMANDS MUST use NEW: PART to add one or more parts to object (initially has none)

NEW: PART *part relx rely imageoffset plane*

Add a part to the current TARG CompoundObject
Call immediately after NEW: COMP (TARG will point to the new object) to add one or more parts to this object
- part is the part number (0-9 (0=main part))
- relx,rely are the position of the part RELATIVE to part 0 (use 0,0 for part 0)
- imageoffset is the base sprite for this part relative to first sprite for OBJECT (not to first sprite in file)
- plane = plot plane (0-9000)
After this command, PART is left pointing to this part number (for subsequent part-relative commands)

NEW: VHCL *imagefile numimages imagenumber*

Create a Vehicle
For default object properties, see CompoundObject above

NEW: LIFT *imagefile numimages imagenumber*

Create a Lift
For default object properties, see CompoundObject above

NEW: BKBD *imagefile numimages imagenumber bkgndcolour chalkcolour aliascolour textx texty*

Create a Blackboard (or wordbook or poster)
- bkgndcolour chalkcolour aliascolour are the colour numbers to use for plotting text
- textx texty are the coords of the place to plot text, relative to part 0
example: new: bkbd BBD1 18 0 240 241 242 4 4
For default object properties, see CompoundObject above

NEW: CREA *moniker sex*

Create a newborn creature.
MONIKER is the moniker to use to locate the child's genome file (this file is generated by: a) the Gene Editor, b) a parent creature or C) the NEW: GENE macro, called by the Hatchery to breed a unique egg)
SEX is 1 if the creature is to be male, 2 if it's to be female or 0 if the sex is to be determined randomly. Normally, sex is randomly determined, but the initial eggs may need to be pre-sexed. All the other creature parameters are determined by the resultant genome.
NOTE: the moniker must be supplied as an INTEGER, not a string literal, so that, for example, EGG objects can store the moniker in OBV0 during incubation.
If I need to store a moniker in a macro as a token, then I must use the TOKN rvalue to convert it to integer.

```
Examples:
                    NEW: CREA OBV0 0              ; create
               creature bred from moniker stored in var
                    NEW: CREA TOKN EVE1 0         ; create
               from explicitely named genome

0=random 1=male 2=female
```

NEW: GENE *mum dad child*

Create a new genome file from mum's and dad's (or just mum's if dad=0) genomes, and store the new genome's moniker in the LVALUE child.
eg. "new: gene tokn eve_ tokn adam obv0" will create a child of Adam and Eve and store the child's genome moniker in TARG's OBV0 variable.
Use this to conceive a child outside the womb - for example from the Hatchery.

Carry out a 'bbd:' command (related to Blackboard objects)

BBD: WORD *index ID [text]* - Install a word/idea into targ Blackboard's list. Used by blackboard editor tools to store edited results, and by Object editor when constructing blackboards. See "DDE: WORD" cmd for reading words

BBD: SHOW *n* - draws the current text string text[Obv[0]] onto part0 (if n=1) or wipes text from bbd (if n=0)

BBD: EMIT - 'speak' the current word so that nearby norns can read it and learn the association between text and concept.
N determines the type of output:
If n=0, word will be broadcast as if it had been read, i.e. to those creatures looking at bbd, with no visible consequences. If n>0 word will be broadcast as if it were a sound, i.e. it is sent to all creatures in EARSHOT, and the word appears in a speech bubble above the bbd. Use n=0 in timer ticks for posters etc. and n=1 when eg. a norn presses a button on a language computer to change the picture.

BBD: EDIT *n* - Allow user to edit the current word (n=1). Prevent further editing and relinquish kbd (n=0)

Execution-Flow Commands

STOP - Stop execution (eg. following error, or before subroutine definitions start)

ENDM - Compulsory cmd at end of macro, placed there by Macro constructor Macro is terminated and maybe self-destructs only STOP (never ENDM) commands may be placed in the body of macro. ENDM is string terminator

SUBR *label* - Identifies a Subroutine. 'label' is a 4-char unique label name GSUB takes us to point AFTER SUBR labl, so only reach here through normal code flow. Therefore, treat SUBR the same as STOP (STOP is therefore not needed before the start of any subroutines).

GSUB *label* - Gosubs to given SUBR label. Often has to scan macro for subroutine start, but always remembers the address of the last subr visited, so most subrs will execute quickly in loops

RETN - returns from a GSUB

REPS *#* - repeat the following code # times, up to next REPE (# >= 1) NOTE: REPS/REPE may be nested, but loops must NOT be jumped out of

REPE - end repeat loop

LOOP - Top of LOOP UNTL statement or LOOP EVER statement (qv)

UNTL *val1* **EQ** *val2* - Part of LOOP UNTL statement. Repeat LOOP unless condition is true Valid conditions are **EQ NE GT LT GE LE BT BF** LOOPs may be nested, but MUST NOT be jumped out of

EVER - Part of LOOP EVER statement. Repeat LOOP forever (usually a dumb thing to do, but OK for eg. some creature's actions, where macro is certain to get replaced by another when action changes) LOOPs may be nested, but MUST NOT be jumped out of

ENUM *family genus species ...* **NEXT** - Iterate through each object which conforms to the given classification, setting TARG to point to each valid object in turn. Family, Genus and/or Species can be zero to act as wildcards.

Example:

```
ENUM 4 0 0              ; for every creature in world
      KILL TARG         ; destroy it
NEXT                    ; repeat till done
```

NEXT (part of ENUM...NEXT)

RTAR *family genus species*
Randomly selects a member from the given classification and sets it as TARG. Null if no members exist.

RNDV *var min# max#* - Set a variable V0-V9 to random # between min# & max# inclusive (could use with REPS/REPE for random # repeats)

SETV *var value#* - Set a variable to a constant/variable value

DOIF *val* **EQ** *val* - do next instructions if condition is true, else skip to after correct nested ELSE or ENDI
Valid conditions are **EQ NE GT LT GE LE BT BF**

ELSE - Hit an ELSE during normal processing (ie. previous DOIF was true), so jump from here to corresponding ENDIF, skipping any nested DOIFs en route

ENDI - Marks end of a DOIF or DOIF/ELSE statement. Just ignore it.

WAIT *ticks* - wait for n ticks (approx n/10 secs) before continuing with next instruction

```
ADDV lvalue rvalue                  ; lvalue = lvalue + rvalue
SUBV lvalue rvalue
MULV lvalue rvalue
DIVV lvalue rvalue
MODV lvalue rvalue
NEGV lvalue                         ; lvalue = 0 - lvalue
ANDV lvalue rvalue                  ; lvalue = lvalue AND rvalue
ORRV lvalue rvalue                  ; lvalue = lvalue OR rvalue
```

DBUG *Rvalue* - Performs in an INSTANCE: sends RValue as a TRACE message that I can view on the debugger. A good use for this is to trace macro sequence of execution. Another use is to display data values, and a third is to put a breakpoint here, so that I can trace macro execution in code.

DBGM *[String]* - Does nothing in release version, but debug version sends String as a TRACE message that I can view on the debugger.

INST - Make the rest of this macro execute in a single tick, regardless of the state of the Repeat variable. Use this instruction at the head of DDE macros that must execute a series of instructions without being interefered with by FastUpdate() calls, etc.
For example, any macro that creates an object should use this so that the object has been fully initialised before FastUpdate() gets to look at it (especially true for CompoundObjects, whose Parts don't get created until several instructions after the NEW: COMP has occurred)

Application, Tool and System commands

SYS: - Prefix to all system commands, such as SYS: QUIT

APP: - prefix to all applet macros that are NOT dde calls these are macros that control the applets rather then talk to them

SCRP *family genus species event* - All the rest of this macro is to be installed in the system as a Script, making it available as a new/replacement script for a given type of object and a given event. This command should normally be the first in the macro.
DDE programs can thus install new scripts into the world by 'executing' the required script, heading it with a SCRP command.
Family, genus and species are numbers that identify the type of object - they relate to the top three bytes of the object's Classifier.
NOTE: each of these parameters is a BYTE value (0-255), rather than the absolute value for that byte ie. A SimpleObject's Family param is 2, not 0x02000000.
Event is the number of the event that will invoke this script: 0=deactivate, 1=act1, 2=act2, etc.
The Species param can be zero - this means that this script applies to ALL objects of this family+genus, if they don't have a script that identifies them exactly. Likewise, both Genus and Species can be zero, meaning that the script is a default script

for all members of that family.

SCRX *family genus species event*
remove any script answering to this description from the Scriptorium (eg. used by ObjEd to delete scripts that are no longer needed)

TOOL *[fsp] [menutext] [helptext] glyph#*
Issued by a DDE tool app to register itself with the toolbar.

EXEC *[fsp.exe] [params]*
EXEC [c:\path\fsp.exe] [params]
Execute a tool or other application. If fsp contains backslashes, it must be a full path, so execute GP program.
If no backslashes, assume it's a tool, so try both hard drive AND CD-ROM [params] are the command-line params for the program (or use [] if none)

ROOM *room# l t r b type*
Set up a room on map. room# is the room to set up (may be a new room) l t r b = room rectangle in world coords type = 0=INDOORS 1=SURFACE 2=UNDERSEA

DDE Data-Logging Commands

DDE: *other data*
DDE: prefix means that some stuff should be written out to the data-logging buffer (at DDEOut). Operand after the DDE: specifies what to send

Sound Effects etc.

SNDF *function* - Set the sound status

```
       Function = ON__  -     Sound on
                  OFF_  -     Sound off
                  FORE  -     Sound only plays when
                              application is in foreground
                  CONS  -   Sound plays all the time
```

SNDV *[filename WITHOUT.WAV suffix]*
Now replaced by SNDE (sound effect) which doesn't require []
This has been kept for back compatibility/ Play sound if TARG obj is visible on screen Change volume according to distance from screen

SNDE *filename* (four letter token)
Play sound effect if TARG obj is visible on screen. Change volume according to distance from screen. This replaced SNDV and doesn't require []'s

SNDQ *filename* (four letter token) *delay*
Play sound effect after a short delay if TARG obj is visible on screen Change volume according to distance from screen
SNDC *filename* (four letter token)
Start controlled sound if TARG obj is visible. Change volume according to distance from screen

SNDL *filename* (four letter token)

Start controlled loop if TARG obj is visible. Change volume according to distance from screen

STPC - Stop any controlled sound currently playing
FADE - Fade out any controlled sound currently playing
PLDS *token* - Preload sound into sound cache if TARG obj is visible or just off screen

Object Commands

TARG *Rvalue* - Set Targ object pointer to point at given object
TARG OWNR - (re)set Targ to point at default object (macro owner, or pet if DDE)
TARG FROM - set Targ to point at cause of this event (no change if isn't an event macro)
TARG NORN - set Targ to point at the current Pet

NEW:

Create a new Scenery, SimpleObject, CompoundObject or Creature

KILL *rvalue*

Delete the object whose address is rvalue, eg. "kill edit" removes any object that's been shift-clicked on (EditObject), "kill targ"
deletes the target object.
THIS INSTRUCTION MUST BE LAST ONE IN MACRO IF IT KILLS THE OWNER OF THAT MACRO!

EDIT

Attach TARG obj to mouse (even if it's not carryable) so that user can position it.
Used by Object Editor to allow NEW: objects to be positioned
Do this by setting the EditObject variable in VivDoc.cpp.
This causes the TaskSwitcher to make this object follow the mouse until a mouse button is pressed.

ANIM *[123432R]* - objects
ANIM *[010203R]* - creatures
Start animation of DEST object/part using these poses
CREATURE: poses refer to entries in the pose table; anims are TWO-digit numbers fr creatures

OVER

Wait until the current DEST object's animation is over, before continuing. CARE: anims ending in 'R' will never stop. COMPOUND, it's the current PART's anim that's checked.

POSE *n*

stop any animation of DEST obj's entity, and set it to POSE# n (pose, not abs image#. ie. same effect as using ANIM [n])
CREATURE: Will continue with next instruction ONLY when target pose has been reached.

PRLD *[1234]*

Pre-load image cache with these poses, to make for smoother animation later CREATURE: n/a

BASE *n*

Specify the base image number for this object/part. Can be used to allow anims from large tables of images, by moving base sprite# around table. Value is an ABSOLUTE index into this object's image gallery. CARE: no error checks!

PART *part#*

Set part# for future actions on CompoundObjects, eg. Animations

MVTO *x y*

move object to abs locn and redraw

MVBY *xd yd*

move object by relative amount and redraw

BHVR *click creature*

Set SimpleObject's reactions to clicks by mouse and activation requests from creatures.

Values for BHVR			
Click - user interaction		**Touch -** creature interaction	
0	clicks have no effect	**0**	creature can take no actions
1	monostable: clicks activate, further clicks have no effect until object is inactive again.	**1**	act1
2	retriggerable monostable: clicks activate even if already active	**2**	act2
3	toggle: 1st click activates, 2nd deactivates again	**3**	act1 act2
4	cycle: 1st click activate1, 2nd activate2, 3rd deactivate	**4**	deac
		5	act1 deac
		6	act2 deac
		7	act1 act2 deac

TICK *#ticks*

Set the TARG object's timer to given rate.
TIMER scripts will be executed whenever this timer times out.
Set to 0 to disable TIMER events

SPOT *spot#* left *top right bottom*

Set up a CompoundObj hotspot, for users/creatures to click on (See KNOB for how to assign a hotspot to an activation function)
spot# = hotspot# 0-5, ltrb = coords of hotspot on object RELATIVE to part[0]
Set ltrb to -1 -1 -1 -1 to remove a hotspot

KNOB *activationfn# hotspot#*

Attach a CompoundObj's activation function (ACT1=0 ACT2=1...) to a given hotspot
(eg. to make hotspot# 0 into a Deactivate button, use KNOB 2 0) set KNOB activationfn -1 to disable an action button

CABN *l t r b*

Set the relative coords of TARG VEHICLE, LIFT or AIRCRAFT'S Cab (cabin rectangle)

GPAS - get passengers

DPAS - drop passengers

SPAS *vehicle creature* - get this particular passenger

Load all nearby creatures into TARG VEHICLE or LIFT, or drop them again. Normal ACTIVATE# scripts for vehicles should call GPAS and normal DEACTIVATE scripts for vehicles should call DPAS. Any vehicle's COLLISION script that effectively deactivates the vehicle on collisions should also call DPAS.
These functions are at the discretion of the designer, in case special behavior is reqd.
SPAS is used to get a single creature into a vehicle; the first param is explicit because eggs use this command to get a given creature into the incubator at hatch time.

BBD:

Prefix for various blackboard-related commands

MESG SHOU *message*
- "shout" send message to all creatures that can hear OWNR obj

MESG SIGN *message*
- "signal" see OWNR

MESG TACT *message*
- "tactile" are in contact with OWNR

MESG WRIT *object message*
- "write" send message to a specific object
Object is a pointer to an object (TARG, OWNR, FROM or NORN)

STM# SHOU *stimulus#*
STM# SIGN *stimulus#*
STM# TACT *stimulus#*
STM# WRIT *object stimulus#*

Emit one of the hard-wired stimuli (STIM_DISAPPOINT, etc.)
Stimulus# is a value from 0 to NUMSTIMULI-1, and refers to one of the built-in stimuli in the stimulus library.
Often this command will be enough, but if a more specialised stimulus is required, use the STIM command (see below)
Object is a pointer to an object (TARG, OWNR, FROM or NORN)

STIM SHOU *list of stimulus items*
STIM SIGN *list of stimulus items*
STIM TACT *list of stimulus items*

STIM WRIT *object list of stimulus items*

```
             Emit a specialised stimulus to a given creature or nearby
             creatures If one of the built-in stimuli will do, use the
             STM# command (above), but if none of these is suitable,
             specify the exact stimulus data using this cmd.
             Object is a pointer to an object (TARG, OWNR, FROM or NORN)
       "list of stimulus items" refers to a list of values, as follows:
               Significance;   - amount to nudge significance neurone by
               Input;          - sensory lobe neurone# (or 255 if none)
               Intensity;      - Amount to nudge input neurone by
               Features;       - bit record of features
            chemical0,amount0,   - 4 chemicals to emit into bloodstream
                                                (0==unused)
            chemical1,amount1,   -  with amounts to emit (0-255 moles)
            chemical2,amount2,
            chemical3,amount3
```

Creature Commands

All these commands apply to the TARG object, which must be a creature TAKE CARE to return TARG to pointing at OWNR before using these commands after changing TARG (eg. to IT (ATTN)

FIRE *x y amount*

Fire the neurone whose position is XY (used by PET scanner, etc.) 'amount' is the signal strength - 0-255 is a 'safe' signal, >255 is lethal to the cell and 'kills' it (useful for brain surgery!)
NOTE: KILLING CELLS IS NOT YET IMPLEMENTED

TRIG *lobe cell amount*

Fire this particular neurone

CHEM *chemical amount*

Add this much chemical n to TARG's bloodstream

APPR

Approach IT.
Choose a walking gait according to chemo-receptors, then start walking towards _IT_. Continue with next instruction when you are WITHIN REACH

WALK

Walk indefinitely.
Choose a walking gait according to chemo-receptors, then start walking.
If extraspective, you'll continuously walk towards _IT_, but this command is primarily for introspective walking, such as "wander east", so creature will walk in current direction using the given gait.

TOUC

Reach out and touch IT.
Normally preceeded by APPR macro. Continue with next instruction when you have successfully touched IT (or when you are as close as you are going to get).
If total failure (no IT, or IT gone below floor level) then the present action schema is suppressed (action has failed) and the macro is terminated.

POIN

Point to IT.
As for TOUC, but creature reaches out to object with head facing camera. This can be used to allow a creature to ask the user what an object is called,
for example. See TOUC for usage.

AIM: *act*

Set the target point on the IT object for subsequent APPR and/or TOUC commands

```
     VALUES FOR ACT
     0: act1   1: act2   2: deac
```

SAY# *n*

Speak word n in a speech bubble, and send that word as a SIGNAL message to all creatures in earshot

SAY$ *[string]*

Speak given string in a speech bubble (no signals sent)

SAYN

Speak your most pressing need

IMPT *n*

Signify how important this (voluntary) action is (how unlikely it is that another action will override this one before it has finished).
value is the amount that gets used to nudge the current decision neurone. This instruction should be used at the start of EVERY creature action macro, and may be used within a macro if the importance changes during a later phase. Values should be low numbers!

DONE

Creatures only. This voluntary or involuntary action has been completed. For voluntary actions: resets the decision neurone to force creature to make a new decision, and ensures current importance is zero.
Put this cmd at the end of any TRANSIENT voluntary action (eg. act1 but not walkeast)
and after EVERY involuntary action

LTCY *action mindelay maxdelay*

Set the Latency for the TARG creature's given Involuntary Action (0-7).
Only relevant to Involuntary Action scripts (Creature's relex actions).
Prevent this action repeating for at least DELAY*4 ticks (DELAY is in 4/10th sec intervals, as decision-making fn gets called only every 4 ticks, and is a random number between min and max).
This command may be called at the end of an involuntary action script to prevent reactivation until the chemical which triggered the action has subsided. A random latency can be useful for actions such as "languish due to lack of strength", to make them OCCASIONALLY override willed actions.

ASLP *0/1*

Go to sleep (close eyes, become insensible to some stimuli) or wake up.
Instruction doesn't change pose - macro must do this after ASLP instr.
Any change of action will automatically wake creature up again.

DREA *max*

Start dreaming, ie. start processing any pending instincts, instead of receiving sensory data from environment. Normally, this should be done only during deepest sleep phase, plus during embryology, while the creature is in limbo before hatching. Once activated, MAX pending instincts will be processed, then the dream state switches off automatically.
Each instinct takes about 5 secs, during which the creature is insensible.
Set MAX to a suitable value - too low and insincts take too many sleeps to get processed, too high and creatures remain insensible for too long

DROP

Drop any object(s) that you are carrying.

FK**

Only relevant to male creatures:
Pass any waiting sperm to female (if IT is a female of same genus).
Female will conceive if she's in the right condition (fertile & receptive)

SNEZ

TARG creature sneezes - infect nearby creatures or environment with any live bacteria he has in him

SLIM

Set the limits of the target object

MCRT *x y*

Move a carrot to x y
to x,y and moves the camera with it

TELE *x y*

Teleport all of the vehicles occupants
to x,y and moves the camera with it

EVNT *object*

Add an object onto the Event bar
(either a newborn, and egg or a death)

RMEV *object*

Remove an event from the event bar

```
// do all asynchronous instrs at once, but let others execute at
// one instr per tick, UNLESS Immediate is set, in which case ALL
// instrs get executed in a single pass
```