CREATURES DEVELOPMENT GUIDE

COB BUILDING....1
Event Numbers....2
Object Pointers....3
Simple Objects....4
Class Calculator....7
Coumpound Objects....8
World Info....8
File Formats....8
NORN BODY DATA....8
It's all gone horribly wrong....9
MACRO LANGUAGE GUIDE....10
Object pointer operands....10
System operands....10
TARG Object operands....11
Values for ATTR....11
TARG CompoundObject, Vehicle and Lift operands....11
TARG Creature operands....12
Environmental operands....12
Truth test operands....12
Set activity state....13
system operands....13
camera, window and scrolling control....15
carry out a 'bbd:' command....18
execution-flow commands....19
Application, tool and system commands....21
Event Numbers....21
DDE Data-logging commands....22
Sound fx etc....22
Object commands....23
Values for BHVR....25
Message meanings....26
Creature commands....27
EVENT NUMBERS....31
CLASSIFIER REFERENCE LIST....32

COB BUILDING

A COB file consists of a list of macro strings – *scripts* and *imports* – that will be injected into Albia through the Objector Injector applet available from http://www.cyberlife.co.uk
A *Script* is a string of macro commands that will be stored in Creatures and gets activated at a later time from an owning object, whereas an *Import* is a macro string that will be processed straightaway.
The only difference between the structure of Scripts and Imports is that Scripts always start with the SCRP command (see Macro Language Guide) which marks the script as belonging to a particular family, genus or species.

The format for a script is:

SCRP *family genus species event*,<macro string> **endm**

Where *family genus* and *species* are values in the range 0 – 255 corresponding to the owning object for this script. *Event* is the event that triggers this script to activate.
(See Classifier list in Appendix A for a full list of existing objects and their unique classifier))

A COB will usually make reference to at least one other file – it's image file (*.SPR) which contains all the images for this object, but it can also reference a sound file (*.WAV) so that you can trigger sound events within your code.
SPR and WAV files are not the sole property of a particular COB, so you can specify a sound for your new object, for example, that is used by something else. I.e. The file DROP.WAV is a sound effect for an object hitting the floor – this can be used by any and all objects as needed.

For example,

scrp 2 6 10 7,doif posb lt limb setv var0 posb setv var1 limb subv var1 var0 mvby 0 var1 endi endm

This is a script for the object with a classifier of [2 6 10] – which is [simple object / food / coconut pieces]. Its event number shows that this is the *enterscope* script (see SCRP command for details of event numbers)

Don't worry too much about the macro string part of this yet, come back to it when you feel you know the ins and outs of the language, the important thing is the format of it that makes it a script to be stored and acted upon when a condition is right. In this case this script will activate when the coconut pieces enterscope, and if they are above ground level they will fall to the ground.

Event Numbers

0	deactivate
1	activate 1
2	activate 2
4	picked up
5	dropped
7	enter scope
9	timer
16	extra Quiescent
17	extra activate 1
18	extra activate 2
19	extra deactivate
22	extra pickup
23	extra drop

These meanings should be fairly intuitive – the only ones that may be unclear are enterscope, timer, and the extra<something> events.

Enterscope scripts are activated when an object enters the world of Albia – this is usually due to injection but also happens when Creatures is started up if the object is already in the world.

Timer scripts are activated every *n* ticks – the value of *n* can be set or changed within other scripts, or the import macro.

The Extra <something> scripts are executed on the executor when they chose to do this action to the owning object. I.e. extra activate 1 is the script for a creature activating 1 that object.

The Pointer <something> scripts are what happens to the on-screen hand when it performs this action, i.e. pointer pickup scripts usually state that the hand changes to it's 'holding' poses.

50	pointer act 1
51	pointer act 2
52	pointer deactivate
53	pointer pickup
54	pointer drop

(For a full list of event numbers see the SCRP command in the Macro Commands section).

It is scripts that make up the bulk of a COB file and as they are injected they will be stored in the scriptorium, overwriting any scripts for the same classifier.

Imports are usually used to initialise and build an object and place it in the world – it will then be able to use scripts marked as belonging to this object, which have been stored in the scriptorium.

Object Pointers

Macro commands usually operate on the ***target*** object – TARG – but it is possible to change TARG to point at a different object using macro commands. TARG is set as soon as an object is created, so during an install script TARG will refer to the last object created, and so all commands that act on TARG will affect this object.
Other useful object pointers are:
OWNR – this is the owner of the script (i.e. the object specified in the **SCRP *family genus species event*** header)
FROM – the object who caused this event to happen (i.e. if Azzam (a Norn) activates the spinning top, then he is the FROM object as far as the top is concerned.
NORN – the currently selected Norn from the Norn menu.

To change TARG so that it uses a different object pointer you just specify; TARG OWNR (for example).
Other commands such as ENUM and RTAR can also change TARG – these are specified in the Macro Commands section.

Simple Objects

Simple objects are the most common kind of objects in Albia, they all have certain universal characteristics which make them different from Compound Objects – namely gravity and the potential to be carried. Both the hand and creatures can carry simple objects, and when they are dropped they will fall to the floor.

Simple Objects all belong to family 2, and their classifiers must reflect this if you want the object to behave like a simple object. The most useful genus' of simple objects are listed below;

2. Call button
4. Good herbs
5. Eggs
6. Food
7. Drinks
8. Food
9. Instruments
10. Animals
11. Hot
12. Soothing
13. Small Toys
14. Large Toys
15. Bad Herbs

Below is an example Simple Object COB, with a chunk-by-chunk description of the scripts and import. The example uses the Bed-time bear from Object pack 2

This is the import macro code for the Bed-Time Bear

inst

This will make the rest of the macro run in an instance, this makes sure it is not interrupted by other macros already running.

sys: wtop

This moves the main game window to the top.

vrsn 2

This checks that the version of Creatures you are running is version 1.0.2 or higher. If not the COB will not execute.

new: simp pets 1 24 700 0

This is the main macro for generating a new object from a sprite file.
NEW: SIMP states that a new simple object is being defined,

TARG is set to this new object, so all further commands that use TARG will apply to this object.
Pets 1 24 700 0 states that the image(s) for this object are in the file called 'pets.spr', there is only 1 image and it is at position 24 within the SPR file. 700 is the objects image plane – how far into the screen the image is to be placed. The '0' on the end states that there is no need for a cloned image gallery – this is true for nearly every object you will want to create.

setv clas 34408704

This sets the *classifier* of the current TARG. The classifier is its unique family/genus/species id. This number is calculated from the hexadecimal version of the long form classifier (see class calculator below)

setv attr 67

This sets the attributes of the TARG. In this case this sets the bed-time bear as being wall-bound and carryable by both hand and creature (see Macro Guide for listing of values for ATTR).

Bhvr 0 1

This sets the behaviour of the TARG. In this case it sets the Bed-Time bear as being only activatable by the creatures, not the hand (see Macro Guide for listing of values for BHVR).

Mvto 2712 892

This moves TARG to co-ordinates 2712,892 (near the incubator)

sys: camt

This moves the camera to point at the current TARG

mesg writ targ 8

This tells the current TARG to enterscope, this has the effect of almost 'kicking' the object to life. Without this command the object would not react to gravity or its attributes. Objects do not have to specifically have an enterscope script defined to send this message.

Endm

Compulsory end of macro command

Ok, so that's created and initialised an object that now sits near the incubator. But what happens if it is picked up or activated? That is where the scripts come in. The Bed-time bear only uses two scripts and these are explained below.

scrp 2 13 9 4

This marks this script as belong to the object who's class is [2 13 9] and it is for event 4 which is "picked up".

stim writ from 0 255 0 0 40 50 42 50 23 50 34 50

This stimulates the FROM target (the object/creature that picked up the owner (OWNR)) with the list of chemicals specified (see Macro Guide for details of this command). What this does is stimulate the FROM target with 50 moles of loneliness--, 50 moles of fear–, 50 moles of sleepiness++ and 50 moles of need_for_pleasure--

endm

scrp 2 13 9 1

This is the Activate1 script for the Bed-Time Bear.

stim writ from 0 255 0 0 40 50 42 50 23 50 34 50

This is identical to the picked up script.

setv actv 0

This marks the object as having finished activating, without this the object would not be able to be activated again until it received some form of deactivate signal.

Endm

And that's it! So now we have a bear that can be picked up by both hand and creature and also activated by a creature. When it is picked up or activated it makes the creature feel less fearful and lonely and also sleepy – a comforter for young insecure creatures.

So to summarise:

- A COB is a collection of *Scripts*, and possibly an *Import* too.
- Scripts are headed with a *classifier* and an *event*, they are placed into the scriptorium and are called whenever an object with that classifier has that event.
- Imports are injected live and acted upon immediately.
- Any object created in an import needs a classifier, some attributes and behaviour – and a location to appear if it is a visible object. Remember to tell the object to enterscope after creation.

Hints, tips and examples

- "edit" in the import macro will place the object in the hand.

```
Inst vrsn 2 sys: wtop new: simp deth 1 0 300 0 setv clas 34408448 setv attr 71 bhvr 2 1 edit endm
```

This is the import macro for Solution X from Object Pack 2

- You can use the object variables (OBV0, OBV1 and OBV2) to hold information you want to persist beyond the running of a particular script.

```
doif obv0 le 0 gsub hunt endi doif obv0 ge 2 wait 10 gsub hive endi
```

This is a chunk from the bees enterscope script – basically it checks to see if they have a supply of nutrients from a plant (obv0 > 0) and if not they go to a subroutine to fly to a plant (hunt). If they are above a certain level of nutrients then they will return to the hive. Other scripts can modify this object variable, for example the hive would reset the value to 0 when the bee returned to deposit its nutrients whereas successful plant maneuvering would increase the count.
Honey Jars also use object variables, but as a 'use' counter. When the number of uses is down to zero the jar is empty and needs re-filling.

Class Calculator

The classifier of an Object is it's one unique identifier and so calculating the class of an object is vital – if the class is wrong you may end up overwriting scripts that exist already in the scriptorium. The classifier list in Appendix A lists all currently known objects and the classes they use. Before you start to make new objects you must find an appropriate free classifier.
Norns generalise about objects at the genus level so it is important that if you create a new object it's classifier is chosen to fit it into the most appropriate genus, otherwise your creatures will not act towards it as you might have expected.

The classifier id has 2 forms: informal and formal.

The informal version takes the form *family genus species* (i.e. 2 13 9 for the Bed-time bear).

The formal version is calculated from the hexadecimal expression of the informal form. This hexadecimal expression is first arranged as an 8 digit number (the last 2 digits are always 00), and this is then converted into a decimal number.

For example, Bed-time bear has a classifier of [2 13 9], this is expressed in a hexadecimal form as [020d0900]. The decimal version of this is 34408704 – the Bed-Timer bear's unique classifier in formal form.
Alexander Laemmle's COE has a built in class calculator that can turn the informal form to the formal – it is highly recommended as a Creatures development tool.

Coumpound Objects

World Info

File Formats

NORN BODY DATA

Two types of file are used to define a norns body – sprite files (.SPR) and attachments (.ATT).

A norns images are built into it's own SPR file using a series of base variants, a norns genetics dictate which variant it uses for it's body, head, arms and legs. These base variants files are numbered with the following scheme:

L X Y Z .ext				
L: Body Part	X: Gender	Y: Stage of Life	Z: Variant	.ext: SPR or ATT

BODY PART	GENDER	STAGE OF LIFE	VARIANT
A: Head B: Body C: Left thigh D: Left shin E: Left foot F: Right thigh G: Right shin H: Right foot I: Left humerus J: Left radius K: Right humerus L: Right radius	0: Norn Male 1: Grendel Male 4: Norn Female 5: Grendel Female	0: Baby 1: Adolescent 2: Adult 3: Old	0: Brown mouse 1: White haired pixie 2: Devil/blondie 3: Santa 4: Purple Mountain

So, for example, file H421.SPR is the image file for the right foot of an adult female pixie norn.

It is from these files that the individual images for particular norns are made and complied into one file that holds all of it's images (for that stage of life) – this file will have the same number as the owners moniker (eg. 1kqy.spr). When a norn grows it's image file is remade, picking the images from the next stage of life for that variant and gender.

From looking at the existing *LXYZ.SPR* files that the images are all in an ordered sequence – this sequence is important to replicate if you want to make your own variant files.

NOTE: The poses are generally arranged in a bottom/back to top/front order.

Head
The images for the heads are arranged in the following order;
EAST 0 1 2 3, WEST 0 1 2 3, FRONT, BACK, HAPPY, SAD, ANGRY
Where 0 1 2 3 are poses in the stated direction.

Body and Limbs
The rest of the body parts are arranged in the following order;
EAST 0 1 2 3, WEST 0 1 2 3, FRONT, BACK
Where 0 1 2 3 are poses in the stated direction.

Attachments

The attachment files are strings of co-ordinates – it is these co-ordinates that enable the image files to join together smoothly. The co-ordinates are distances into the image file.

Head
For the head the attachment list is 10 sets of co-ordinate pairs – (Center of Head, Mouth) – for poses in the following order: EAST 0 1 2 3, WEST 0 1 2 3, FRONT, BACK.

Body
For the body the attachment points are more complicated, they are in the following order – (head, left leg, right leg, left arm, right arm, tail) – where tail is (0,0) for Norns and Grendels.

Limbs
The limbs attachments are (top of limb, bottom of limb) – so for the humerus the points are (shoulder, elbow), for the radius they are (elbow, wrist), for the thigh they are (hip, knee), for the shin they are (knee, ankle) and for the foot they are (ankle, end of foot)

It's all gone horribly wrong…

So you've injected your COB and now Creatures is reporting an error, this could be because of a syntax error or something more fundamental. Creatures will produce an error message that should provide useful information about where the error has happened.
Here are some of the common reasons for failures:

- An image (SPR) specified in the import macro does not exist in the creatures\images directory. The image file is specified in the NEW: commands and if this is not found then you'll get an error message as soon this command is acted on.
- A sound (WAV) specified in a macro does not exist in the creatures\sounds directory.
- The sprite offsets stated in a macro command are not valid. For example, ANIM [01234] when there are only 3 images in this sprite file. This can be very easy to do when you use large sprite files with many images and the BASE command.
- Wrong TARG. It can be very easy to forget to change TARG back to OWNR after you use a command such as ENUM or RTAR to chose a new TARG. Without this change of focus back

to the owner all commands will operate on the selected object – and this may be enough to make the system hang if the chosen object is not intended to carry out the owners commands.

- Spacing errors. All commands are separated by a single space and if this is duplicated or omitted then the parser will fail.
- Typing errors. Your fingers are flying away at 200 wpm and you try to EUNM across a species.
- Old version of the Creatures.exe. New commands were introduced in versions 1.0.1 and 1.0.2 of Creatures – any COB that uses these commands will cause an error on earlier versions.

MACRO LANGUAGE GUIDE

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
TCAR – Returns carrier of TARG (may be NULL)
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. NOTE: This shouldn’t really be set to a Norn – there’s nothing stopping you using it but things have the potential to go wrong if OBJP points to a norn who then dies.
TOKN *XXXX* - convert 4 characters into an integer
e.g. 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

```
POSL/POSR/POST/POSB - retn obj's lrtb coords
WDTH/HGHT - retn obj's width/height
LIML/LIMT/LIMR/LIMB - retn obj's limits (e.g. limits of current
                                          room/vehicle)
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)

enum {
            AUTONOMOUS = 0,         default - normal obj in world
            MOUSEDRIVEN,            if *SIMPLEOBJ* is connected to mouse
            FLOATING,               if obj is in fixed place on screen
            INVEHICLE,              if obj is carried in vehicle
        CARRIED,                    if obj is carried by a creature


ACTV - Object's Active flag (INACTIVE=0 ACTIVE=1)
NEID - obj's neural ID# 0-39
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.

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

TARG CompoundObject, Vehicle and Lift 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. Set to 0 to abort a pregnancy (or set to child moniker to make her pregnant)
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

(return 1 if true, 0 if false)

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

Set activity state

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

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 separated by a "|" symbol)for every creature (where creatures are separated 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 details

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 co-ordinates. 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 co-ordinates (e.g. 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.
EGG Objects *must* have OBV) set to specify the moniker of the developing creature.
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

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.
DBGV *Rvalue* - Sends Rvalue to debug window. Same as DBUG but does not run in an instance.
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.

Event Numbers

0	deactivate
1	activate 1
2	activate 2
4	picked up
5	dropped
7	enter scope
9	timer
16	extra Quiescent
17	extra activate 1
18	extra activate 2
19	extra deactivate
22	extra pickup
23	extra drop

These meanings should be fairly intuitive – the only ones that may be unclear are enterscope, timer, and the extra<something> events.
Enterscope scripts are activated when an object enters the world of Albia – this is usually due to injection but also happens when Creatures is started up if the object is already in the world.
Timer scripts are activated every *n* ticks – the value of *n* can be set or changed within other scripts, or the import macro.
The Extra <something> scripts are executed by the executer when they chose to do this action to the owning object. I.e. extra activate 1 is the script for a creature activating 1 that object.

50	pointer act 1
51	pointer act 2
52	pointer deac
53	pointer pickup
54	pointer drop
64	involuntary action 0
65	involuntary action 1
66	involuntary action 2
67	involuntary action 3
68	involuntary action 4
69	involuntary action 5
70	involuntary action 6
71	involuntary action 7
72	Creature death scipt

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.

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 fx 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!

Because the ANIM command for objects uses a single digit for image numbers BASE is needed if you are using a sprite file with a lot of entries.
The example below is from the Cloud Butterfly COB and shows the use of the BASE command – in all other ways the two subroutines below are identical.

```
subr left
base 0 anim [0123] over
anim [450]
mvby -3 0
retn

subr rite
base 6 anim [0123] over
anim [450]
mvby 3 0
retn
```

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

0	clicks have no effect
1	monostable: clicks activate, further clicks have no effect until object is inactive again.
2	retriggerable monostable: clicks activate even if already active
3	toggle: 1st click activates, 2nd deactivates again
4	cycle: 1st click activate1, 2nd activate2, 3rd deactivate

Touch - creature interaction

0	creature can take no actions
1	act1
2	act2
3	act1 act2
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

knobs 0-2 are act1,act2,deac for creature; knobs 3-5 are act1,act2,deac for hand.

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)

Message meanings

0	Activate 1	4	Pick Up
1	Activate 2	5	Drop
2	Deactivate	8	Enterscope

These are the messages that you can send between objects and creatures, objects/objects or creatures/creatures.

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.

MATE

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
```

VRSN *number*

only run this script if Creatures build ID is equivalent or higher. Ie. If macro starts “VRSN 2” then Creatures must be version 1.0.2 or higher to run this script.

VRSN

Lvalue to get Creatures Build ID. Ie. Setv var1 vrsn

EVENT NUMBERS

0. deactivate
1. activate 1
2. activate 2
3. hit
4. picked up
5. dropped
6. collision
7. enter scope
8. Leave scope
9. timer
16. extra Quiescent
17. extra activate 1
18. extra activate 2
19. extra deactivate
20. extra seek
21. extra avoid
22. extra pickup
23. extra drop
24. extra say need
25. extra rest
26. extra go west
27. extra go east
28. extra undef 1
29. extra undef 2
30. extra undef 3
31. extra undef 4
32. intro Quiescent
39. intro drop
40. intro say need
41. intro rest
42. intro go west
43. intro go east
44. intro undef 1
45. intro undef 2
46. intro undef 3
47. intro undef 4
50. pointer act 1
51. pointer act 2
52. pointer deac
53. pointer pickup
54. pointer drop
64. involuntary action 0
65. involuntary action 1
66. involuntary action 2
67. involuntary action 3
68. involuntary action 4
69. involuntary action 5
70. involuntary action 6
71. involuntary action 7
72. Creature death scipt

CLASSIFIER REFERENCE LIST

Last updated on:	Sunday, 24.08.1997			
FAMILY	**GENUS**	**SPECIES**	SPR File and offset	IMAGE PLANE
0 Special / Override				
	0 Override Scripts For IT=CREATURE			
1 System Macros (use 1st 4 event#s)				
	1 Scenery Object			
2 Simple Object				
	1 System			
		1 Mouse Pointer	SYST 0-8	
		2 Speech Bubble	SYST 9-12	
		3 Norn Indicator	INDI	
	2 Call Button			
		1 Callbutton		750
	3 Invisible			
		1 Smoke	SMOK	250
		2 Water Fall	FALL	0
		3 Flames	FLAM	50
		4 Wave		9000
		5 Drop	DROP	1
		6 Tree House Flags	FLAG	4000
		7 Windsock	ANIM 16-23	0
		8 Vane	LTVN	0
		9 Garden Invisible		
		10 Jungle Invisible		
		11 Cave Invisible		
		12 Dome Invisible		
		13 Sleep Indicator		9000
		14 Cage control box	CBOX	
		211 Grendel Guard (S.Linkletter)		
	4 Good Herbs			
		9 Feverfew	HERB 0-2	0
		10 Morning Glory	HERB 3-5	0
		11 Tomato	HERB 6-8	0
		13 Campanula	HERB	0

		12-14	
	14 Beelocanth	FLWR 0-9	450
5 Eggs			
	2 Egg For Incubator	EGGS	90
	4 Grendel Egg	GREG	
	6 Grendel Egg Layer	GREG	500
6 Food			
	1 Cheese	FOOD 3-5	750
	2 Honey		50
	3 Carrot		500
	4 Lemon	FOOD 0-2	0
	5 Pudd	HOLI 0-2	750
	6 Turkey	HOLI 3-5	750
	7 Breakable Honey Jars	JARS	50
	8 Bouncy Carrots	PARS	500
	9 Beelacanth Fruit	FLWR 10	450
	10 Coconut Flesh	COCO 6-8	
	12 Beer (S.Kuske)		
	112 Chocolate Bar (S.Kuske)		
	113 Spaghetti (S.Kuske)		
	114 Apple (S.Kuske)		
	116, 117 Cooking Pot (S.Kuske)		
	118 Chips (S.Kuske)		
	120 Icecream (S.Kuske)		
	198 Electric Fan remover (S.Kuske)		
	209 Chocolate Bunny (S.Linkletter)		
	210 Strained Carrots (S.Linkletter)		
	211 Grendel Guard Remover (S.Linkletter)		
	212 Grendel-X (S.Linkletter)		
	220 Milkshake remover (S.Kuske)		
	245 Strawberry Cake (S.Kuske)		
	248 AntiBodies 4567 (A.Laemmle)		
	249 AntiBodies 0123		

		(A.Laemmle) 250 Power Infusion (A.Laemmle) 252 Hyper Tomato (A.Laemmle) 253 Selector Lemmon (A.Laemmle) 254 Fruit Of The Doom (A.Laemmle) 255 Hyper Carrot Dispensor (A.Laemmle)		
	7 Drinks	1 Coffee	KITC 4-7	750
		2 Hootch	HTCH	750
		99 Coffee Cup (A.Laemmle) 114 Beer (S.Kuske) 212 Pink Lemonade Bottle (S.Linkletter) 225 Milkshake (S.Kuske)		
	8 Food Venders	1 Hive	HIVE	1
		2 Still		50
		3 Vending Machine	DISP	100
		4 Beelacanth Seed launcher	LAUN	600
		79 TV remover (S.Kuske) 80 Another TV ! (S.Kuske)	ANTE,	
		99 Coffee Machine (A.Laemmle)		
		200 Electric Fan (S.Kuske) 212 Pink Lemonade Dispenser (S.Linkletter)		
		225 Milkshake (S.Kuske)		
	9 Instruments	1 Harp	INST 0-9	500
		2 Drum	INST 10	3500
		3 Trumpet	INST 19-22	900
		4 Pianola	INST 23-28	500
		5 Jukebox	INST 30-37	50
		6 Jukebox back	INST 29	50

	10 Animals	1 Fish	ANIM 24-33	50
		2 Seahorse	ANIM 0-7	0
		3 Bees	BEES, BEEZ	445
		4 Humming Bird	ANIM 38-41	0
		5 Flying Bird	ANIM 34-37	0
		6 Nesting Bird		0
		7 Jellyfish	ANIM 8-15	0
		9 Goldfish Bowl	PETS 0-9	0
		10 Snowman	HOLI 25-31	750
		11 Reindeer	HOLI 15-24	750
		12 Bug	BUGS	400
		13 Cave Fly	MOSQ	8000
		14 Catapiller	CATA 0-28	
		15 Butterfly	CATA 29-32	
		16 Venus Fly trap		
		17 Crow		
		212 Dragon Fly (S.Linkletter)		
	11 Hot	1 Kitchen Fire	PETS 15-20	0
		2 Cannon	CAN2 2-8	200
		3 Firework Sparks		
		4 Weed Killer MIST	DETH 1-14	
		5 Bug Spray MIST		
		6 Crystal Ball field	CRYS 2-9	
		212 Space Heater (S.Linkletter)		
	12 Soothing	1 Shower	QTOY	4000
		2 Clock	CLOK	0
		3 Tree	HOLI 7-9	750
		4 Grendel Scarer Mist	SCAR 3-20	
		210 Teddy Bear		

		(S.Linkletter)		
	13 Small Toys	1 Spinning Top	STOP	4500
		2 Ball	BALL	4500
		3 Radio	AUVI 22-25	750
		4 Bouncing Heads		4500
		5 Ming Vase		100
		6 Firework		
		7 Bug Spray Bottle		
		8 Weed Killer Bottle	DETH 0	
		9 Bed-Time Bear	PETS 24	
		10 Coconut Husk	COCO 0-5	
		99 TV (A.Laemmle)		
		110, 111 Remote controlled car (S.Kuske)		
		119 Grendel killing machine (S.Kuske)		
		240 Power Pulsar Infusion (A.Laemmle)		
	14 Large Toys	1 Pop-Up-Helicopter	HELI	600
		2 Robot	NTOY 0-7	500
		3 Jack-In-The-Box	NTOY 8-16	900
		4 Crystal Ball	CRYS 0-1	
		6 Grendel Scarer	SCAR 0-1	
	15 Bad Herbs	1 Pyrethium	BHRB 0-2	0
		2 Nightshade	BHRB 3-5	0
		3 Ugly Tomato	BHRB 6-8	0
		4 Gentian	BHRB 9-11	0
		5 Deathcap	MUSH	0
		6 Baobab	HERB 9-11	0
		7 Laburnum	HERB 12-15	0
		8 Holly	HOLI 10-12	750
		9 Mistletoe	HOLI 13-14	750
		15 Grendel Machine		

		Toxin (S.Kuske)		
3 Compound object	1 Vehicle	1 Teleporter		600
		2 Shelf		
		3 Incubator	INCU	100
		4 Cannon		200
		6 Submarine	SUBM	4000
		7 Island Boat	BOAT	4000
		9 Cable car	CABL	4000
		10 Cart	CART	4000
		11 Pull Raft	RAF2	4000
		12 Underground Raft	RAF2	4000
		13 Restraining Cage	CAG2	
		212 Soap Bubble (S.Linkletter)		
	2 Lift	1 Cane lift	LIFT	1000
	3 Computer	1 Computer	COMP	0
		208,209,210,211,212 Encyclopedia Nornica (S.Linkletter)		
	4 A/V equipment	1 Slide Projector	AUVI 0-21	0
		1 JukeBox		0
	5 Cannon	1 Cannon	CAN2 0-1	200
4 Creature	1 Norn	0 Scripts For Both Sexes		
		1 Scripts For Males		
		2 Scripts For Females		
	2 Grendel	0 Scripts For Both Sexes		
		1 Scripts For Males		
		2 Scripts For Females		
	4 Side (add-on sp.)	2 Scripts For Females		
		1 Scripts For Males		
		0 Scripts For Both Sexes		