AlphaBase HM7 Documentation
BUILT: 10, v3.6.8; by Wesker aka DekraN

Summary:

General Introducing In AlphaBase Environment:

Commands may be roughly divided into three categories: simple, compact and complex commands. The first and the second types
splitting parameters method is by introducing a space, except at the last pa ram. So, the only difference between them
is that the first one accepts more than one par am, while the second accepts only one, in order to fix the problem of
introducing a path which contains space and to get more powerful whilst rea ding results;
The third commands type is an evolution of the second type: Commands lik e this may accept more than one param,
preserving the power of the 2nd type com mands, at one single condition: pa ram must be split by the

Complex Parameter Seperator (see Chapter 9), that is default-set to: '~'.
Commands Params is signed by symbols (only for estetical reason, so symbol s mustnt appear in the inputline),
( ) and [ ], respectively the first one type could be omitted, while the second couldnt.
Commands Helper: if you forgot the usage of a command, just type it withou t parameters and A.B. will provide to display
you the usageprint and more informations about the interested PSP parts; so mehow just be carefull
that some commands have the shortcut system (pre-imposted parameters) so they can execute unappropriate code for you.
NOTE: In this Documentation we'll use the '~' in the complex commands us ageprint. However,
It refers to the current Complex Parameter Seperator. Keyboard Type can b e changed by typing the command 'keyb'.
Danzeff and Official OSKs are currently availables; Programme Mode can b e changed by typing 'mode': Must be 'Normal'
(more powerful) or 'Extended' (more information in AB p rints but makes s lower the programme);
Desktop Mode can be set (the only difference from a simple image loading/bli tting is that
this special image will be blit before printing the AlphaBase Strings and ot her User Execution).

Basical Error

P.S.E.R. Parameters Commands
(Path Single Extension Requires):

The P.S.E.R. Commands are commands which generate directly input and the y can be obviously divided into two categories,
Ending Require and Not Ending Require commands. Here's a list with simpl e explanations:
* "prx [PATH]~(MODE)" - Loads the specified PRX in PATH. It will load in the Kernel Memory if MODE is 1;
  "eboot [PATH]", - Runs the specified EBOOT in path after 3 seconds;
  "psx [PATH]", - Starts the specified PSX GAME in PATH;
  "upd [PATH]", - Starts the specified UPDATE in PATH ;
* "elf [PATH]", - Loads the specified ELF in PATH;
  "iso [PATH]", - Starts the specified ISO in PATH;
* "dfile [PATH]" - Loads the specified LUA (or other readable formats) fil

e in PATH.

* = Not Ending Require. This commands have the noextension proof system, so, you can safely omit the extension;
however, it's highly recommended to put it in any case to increase the execu tion speed.

Useful Simple Commands:

"ext [PATH]~(FPATH)~(PASSWORD)", - Extracts the specified ZIP or RAR (don't know if rar also works) in PATH;
"list [PATH]~(PASSWORD)", - List the Path (which may be a Directory, Z IP or RAR) content.
For those two, requires a PASSWORD param; "create [PATH]", - Create the Directory in the specified PATH;
"rem [PATH]", - Remove the specified PATH; "rename [PATH]~[NEWNA ME]", - Rename the specified PATH as NEW NAME;
"size [PATH]", - Gets the size of PATH. If it's a directory then will require more time than the necessary,
cause the HM7 functions completely doesnt work; so, i had to introduce a f unction made by myself;
"exists [PATH]", - It will tell you if PATH exists or not;
"copy [PATH]~[FPATH]~(MODE)", - Copy the content of PATH in FPATH. MODE must be 0 for null, 1 for deleting PATH;
"dirs", - Will create the basic PSP Folders
"md5 [STRING]" - Return the MD5 of the given STRING;
"info" - Return all the information about all the PSP Parts (or almost...).
It will give you more information about the track, if it's currently playing;
"lua" - It will show you a message containing the LUAPlayer information (cr edits);
"sdata [SAVE_DATA]~(SAVE_NAME)~(GAME_NAME)~(DETAILS)~(ID)", - Will create a new Save;
"ldata [ID]" - Will load the Save by ID.

Utility Commands:

"flash [FLASH]", - It will assign or unassign the specified FLASH on the bas e of status;
"usb (FLASH)", - Establishes an USB/FLASH connection, except with the fl ash2 cause a bug.

For establish an USB/UMD connection just type something else instead of FLASH;
"tick", - Will send a Power Tick to the PSP. I'm not sure how useful it is inshell...
"mem", - It frees the memory; It's highly recommended to do this before doing heavy commands;
"lcd", - It will enable or disable the LCD Timer and return the Time Remaining before LCD turns off.
"oa" - It will enable or disable the OA functions (Music and Voice);
"cpu (SPEED)" - Will set the Cpu Speed as SPEED, which is default-set to the initial cpu_clock var (SETTINGS.INI);
"cpu2 (MODE)"- Will set the Cpu Speed at 333 MHz if MODE is 0, else if MODE is 1, Cpu Speed will be set at 266 MHz,
else if MODE is 2, It will be set at 133 MHz (However, it's highly recommended to don't set at Low...);

PSP S.M. Commands:

"sby", - It puts the PSP in Standby Mode;
"sdown", - It will completely turn off the PSP;
"sleep [SECONDS]", - It puts the PSP in Sleep Mode for SECONDS;
"q(uit)", - It Quits AlphaBase and returns to dashboard;
"oload", - It forces the freeze of the PSP after 3 seconds.

Input/Output Commands:

"msg [STRING]~(MODE)", - It will show you a System Message displaying MESSAGE.
MODE must be 0 for showing only the back option, 1 for showing the Yes, No and Back options;
Returns the I/O schedule if mode is Extended;
"sio [STRING]~(BAUD)", - Will send STRING via Serial Port. You can also set the baudrate as BAUD.
It works only for FW 1.51 and lower;
"irda [STRING]", - Will send STRING via Irda. PSP Model must be PHAT, else will return an error message;
"ah [STRING]", - Will send STRING via Adhoc and gets the respective OUTPUT;
"sacc [SOCKET]~(STRING)", - Will send STRING via Normal Socket and

gets the respective OUTPUT;
"udps [SOCKET]~(STRING)" - Will send STRING via UDP Socket and get
s the respective OUTPUT.

AH, WLAN, Sockets and UDP Sockets:

"ahs [PATH]", - Will send the PATH to another PSP via ADHOC;
"ahr [PATH]", - Will receive the PATH sent by another PSP via ADHOC;
"server [SOCKET/ID] (PORT)", - Will create a server socket, which listens
on the specified PORT;
- UDP:
"udp", - Will socket-connect to the standard UDP server for netlib 2.0,
or will close connection to any UDP Server Socket opened;
"net [ID]~[DATA]~(MODE)", - Sends the specified DATA to the ID file on
the server;
MODE must be 0 for write, 1 for append;
"ninfo [ID] (MODE)", - Will enable or disable the Sending File Information
process;
"mail [TO]~[FROM]~(MESSAGE)~(SUBJECT)", - Sends an email to TO fr
om FROM with the subject of
SUBJECT and the message of MESSAGE;
"call [CONTACT] [DESTINATION]", - Sets up a call between CONTAC
T and DESTINATION;
"sms [TO]~[FROM]~(MESSAGE)" - Sends an SMS to TO from FROM and
the message of MESSAGE.

Loading and Unloading an element:

Loading a font, an image, a color or connecting to one (UDP) Socket is gett
ing easily with AlphaBase 2.0
new environment management. The following comands setup the element
management:

"load [ELEMENT]~(PORT)~(MODE)", - Load an element. PORT and MODE
are only for (UDP) Sockets,
respectively the listening of a SOCKET and the MODE, 0 for Normal, 1 fo
r UDP;
"unload [NAME/ID]~(MODE)", - Unload the element recognized as NAME/ID
. MODE must be: - 0, Fonts; - 1, Images; - 2, Sockets;
Now you can unload all elements from AlphaBase environment by typing o

nly the command without parameter.
But be careful and check if you're using some element with User Execution. In that case, just type before 'res'
(without params);
"load2 [MODE]~(X/R,Y/G, /B *ONLY FOR MODE 0 and 3*)~(NAME)" - Loads other special elements,
like instances of Mono Spaced Fonts, Proportional Fonts, Empty Images or Colors, and assignes it an identifier,
NAME (optionally).

String and Special Characters Commands:

If the Environment find n times in the inputline the R.R.C. (Repeat Results Char), It will split the inputline
in n sub-inputlines and will execute them once-per-time.
"char [FONT/ID]~(W,H)~[DPIX,DPIY] - Sets the FONT/ID Pixels Size or Chars Size;
"schar (CHAR)~(REPLACE)", - Will set the Special Char CHAR and will assign it the replace pattern REPLACE;
"ochar (MODE)~(CHAR)", - Will set the Other Chars as CHAR. MODE must be 0 for R.R.C, or 1 for C.P.S. (Complex Parameter Seperator);
"sub [STRING]~[STRING2]~(LINE)", - Makes a substitution in the specified LINE of STRING patterns to STRING2 patterns;
"text [FONT/ID]~[STRING]", - Gets the Size of specified STRING printed using the FONT/ID specified;
"print [STRING]~(ID)" - Print an AlphaBase STRING. It can also make a substitution of the whole ID line with STRING;

User Execution Commands:

"print2 [STRING]~[FONT/ID]~(X,Y)~(COLOR)~(ID)", - Print the given STRING using the given FONT/ID, in the given coords X,Y
(default arrayset is 0,0) using the given COLOR. If it will get the ID param then It will replace the old User Execution Data
with new given Data. (that is 'Shot' System);
"imm [NAME/ID]~(X,Y)", - Blit the given previously loaded image recognized as NAME/ID in the given coords X, Y;
"rect [X,Y,W,H]~(COLOR)~(MODE)", - Will draw a rectangle in the given coords X, Y as width W and height H,
using the color COLOR. MODE must be 0 for an empty rect, 1 for filled rect

;
"line [X0,Y0,X1,Y1]~(COLOR)", - Will draw a line in the given coords us
ing the color COLOR;
"res (ID) (ID2)", - Will delete an User Execution recognized as ID group and
as identifier ID2. If it gets no params,
it will provide to reset all the groups, else if it gets only one param, It will
reset only the given ID group; (look up from the ID's list).
"input (ID) (ID2)" - Will give you more information about an User Executio
n recognized as ID group and as identifier ID2.
If It gets no params It will tell you the number of total AB elements (call
ed User Execution in AB Environment),
else if It gets only one param then It will tell you the number of total eleme
nts of the given ID group;
"set [TIME2,TIME1]~(MODE)~[INPUT]" - Will set a new timer (TIME1 is th
e start time, TIME2 the end time),
which will execute the given INPUT at the end of the timer. If MODE is 1 t
hen the PSP System will sleeping awhile;
"time [INPUT]"- Will stop or resume a timer identified by INPUT;

HowTo rip an UMD or capture a screenshot:

"umd (PATH)" - Example: "umd ms0:/ISO/Tomb_Raider.iso" will rip the UM
D currently inserted in this PATH;
If the command gets no param then It will start the UMD currently inserted
after 3 seconds;
"save [PATH]" - Example: "save ms0:/PSP/PHOTO/pic1.PNG" will capture
a screenshot in the given PATH.
The current formats allowed are: JPG, JPEG and PNG.

HowTo play a track or a playlist:

Before playing a track, this last must be previously loaded by using the c
ommand "load [PATH]". You can load only one element per time for format.
The allowed formats are: uni, it, xm, s3m, mod, mtm, stm, dsm, med, far, u
lt, 669, wav, wave, ogg, ogm, oga, ogv, ogx, mp3 and aa3.
After did this, you must type "play (MODE) (MODE2)", where MODE is the ex
tension identifier of the file loaded.
With the 3.2.4 update however, you can directly load and play by typing
the macrocmd: "play2 [PATH]~(MODE)".
It must be 0 for Ogg, 1 for Mp3, 2 for Aa3 and 3 for other low foormats. M

ODE2 is a boolean value which refers
to the repeat mode. If you type 'play' another time with no params then you will respectively pause or resume
the current track. For stop the track or the playlist currently playing you can simply do 'stop'.
Creating a playlist: - First, you must put the file you want to play (remaini ng however within the formats limits)
in the path you want and so, put the name of the files in an AUF (or other rea dable formats) file in a single line,
splitting the respectively paths by the file seperator, that is default-set to "//" .
Remember: you must put the seperator also at the last file declaration (this i s the only case that an AUF File requires this).
After did this, inshell simply type "plist [PATH]" where PATH refers to the path of the AUF File previously created.
You can also modify inshell the playlist by typing "mlist [PATH]".
Somehow be careful to put the correct formats in the AUF File that cause a bug,
AlphaBase Environment cannot do any control of the existance or the extensi on of the files declarated in the AUF.

D.T.E. Commands
(Directly Template Executing):

"exec [LINE]", - Will directly execute the given LINE, writing this in a temp .lua temporary file.
LINE must be any LUA sintactically-correct and executable LINE, so It can be also any statement.
"asrt [EXPRESSION]~(MODE)~(STRING)" - Will assert the given EXPR ESSION.
if the EXPRESSION returns false if MODE is 0 then nothing will happen, wi ll be displayed only an info message;
else if MODE is 1 then will be displayed a LUA error and then Environment w ill have to restart to work;
"calc [EXPRESSION]" - Will calculate the given math EXPRESSION. You ca n use also all the LUA math library functions,
cause this command is perfectly equivalent to: "exec printf(\"RESULT: \" ..val(EXPRESSION))".

F.M.S. (File Manipulation System):

F.M.S. in AlphaBase environment is easily than the correspective LUA metho ds, even if is powerless.
"write [PATH]~[DATA]~(MODE)~(WHENCE,OFFSET)", - Will write the given DATA on the PATH File.
MODE must be 0 for entering in writemode, 1 for entering in appendmode.
WHENCE, OFFSET params are optionally to change the current file manipula tion pointer position.
"read [PATH]~(LINE)" - Will read the whole PATH file if LINE param is em pty, otherwise will read only that LINE.
Tip: you can temporary file manipulate a file (by don't closing immediatly the file pointer), by doing this:
"exec local hi_txt = io.open("folder/hi.txt",'w')" Sample: "exec hi_txt:write("Hello, World!\n") hi_txt:close()"

Login/Register, Users and E.S.M.
(Environment Security Management):

Login/Register system was implemented in AB Environment just for the a-c md.auf system, which refers to the permissions of same commands (see Nex t Chapter).
You can register any nickname which doesnt already exist in the use rs folder by typing "reg [NICKNAME]~[PASSWORD]~(PASSWORD)".
The third param is required if mode is Normal and the convalidation system is enabled, for enforcing the E.S.M..
(you must repeat the PASSWORD). If mode is Normal you will automaticall y logged in.
For log in, type "log [NICKNAME]~[PASSWORD]". For log out, just type th e (obviously after logged in)
'login' command without params. The level of a new user refers to the 'leve l' vars which may be set in the SETTINGS.ini
or directly inshell by typing "lvl (LEVEL)"; obviously to do this you must b e logged as the main level
(the high level of all the existant users). You can also set directly the level of an user by typing
"slvl [NICKNAME]~(LEVEL)". You can also change your password b y typing "pass [OLDPASS]~[PASSWORD]~(PASSWORD)".
You can suspend an user by typing "susp [NICKNAME]~(MODE)~(STRING)", where NICKNAME is the users; MODE must be 0 for
null, 1 for kicking the user if he tries to enter. STRING refers to the possibl e reason;

"adm [NICKNAME]" - Will set as admin the given NICKNAME (by setting hi
s level as the max level). If the command
is called without parameter will show you the admin list;

Use of a-marks.auf, a-cmd.auf and macrofiles:

a-marks.auf - You can create new commands by mixing the existant or simp
ly assigning to a new command an user input.
For example, writing into a-marks.auf "load intheend.mp3#rplay 1//p//info#
 rcmd//x"
will assign to the new commands 'p' and 'x' the respective inputlines which p
recede it.
The line up-quoted is perfectly equivalent to: "load intheend.mp3//p//play 1//
p//info//x//cmd//x" and this is a bit more readable.
You can directly add new markers inshell by typing "marks [NAME]~[INPUT
]", where NAME refers to the new input environment has to replace (iden
tifier);
a-cmd.auf - You can set the permission of a command by typing "ctrl [CM
D]~(LEVEL)", where CMD refers to the command you want to scan at the c
alling,
while LEVEL refers to the level required of the current user for do that c
ommand. You can also set outshell the commands permission,
by opening the a-cmd.auf and by declaring "CMD//LEVEL//ecc.".
a-pos.auf - You can assign to a given screen area, which is identified
by the screen bounds (XMIN, YMIN, XMAX, YMAX).
You can do it inshell by typing "pos [INPUT]~[XMIN, YMIN, XMAX, YMAX]", o
r outshell by setupping the current poslist file (that is default set: a-
cmd.auf).
HowTo setup a poslist: write "XMIN,YMIN,XMAX,YMAX//INPUT//...//" int
o poslist;
Macro System - You can create alias of any command by loading a macro.
The macro is temporary if you load it by typing "mac [CMD]~[REPLACE]",
static if
you load it from a macrofile (that must be 'AUF' or other readable forma
ts) by typing "fmac (PATH)", where PATH is the PATH
of the macrofile.
HowTo setup a macrofile: - If you write, for example in a-mac.auf (the defau
lt macrofile):

"
info//psp//informazioni;

macro//alias//also;

"
NOTE: // refers to the current F.P.S. (File Parameter Seperator);

, if you type 'psp' or 'also', then the results will be the same as if you type re
spectively 'info' or 'macro'.
This is NOT the same thing as the User Input Replacer (looks a-marks.auf usag
e), in fact the U.I.I. replace the
entire input you enter to another one, while the Macro System directly defi
nes a new command identifier
that refers as the same id as the main CMD identifier;
To check whether a given macro is valid or not, just type "mfind [CMD]".
If you have to replace a command which doesnt require any param,
it's highly advisable to use the Macro System, instead of the U.I.R..

HowTo use mouse and set its sensibility:

Before switch mouse system on, you must set an image previously loaded as
the respective cursor,
by using the command "mouse [NAME/ID]", where NAME/ID is the image. Af
ter did this, if mode is normal then
mouse system will automatically switch on, otherwise you have to do 'switc
h'. You can switch the mouse system status
also by pressing triangle inshell. For set its sensibility, simply type "sens
(x, Y)". If the command gets no params
then It will automatically assign the sensibility as the WIDTH/HEIGHT of t
he cursor image.

Multi-Environments stuffs:

From the v3.60 Version, you can work with Multi-Environments.
So, all the User Execution except the systemprints are linked by the current
environment.
The C.E. (Current Environment) refers to the visible interface which contain
all the systemprints
of the suddenly environment; while the W.E. (Working Environment), refers
to the environment in which
the User Execution will be linked. To create a new environment, just type
"envs (ENV)". ENV refers to

the optional name you want to give it. To eliminate an environment, just ty
pe "envs (ENV/ID)".
To set the C.E. or the W.E., type "cenv (ENV/ID)~(MODE)", where MODE m
ust be 0 for the Current,
1 for the Working. L and R keys are default sysbinded to switching the Cur
rent Environment respectively
on the Left or Right. It's also possible to set the M.E.V. (c Environment V
ariable) by typing "cenv (MAX)",
where MAX is the limit the environments number can't trespass. So, if Al
phaBase already reached the MEV
and you're trying to add a new environment, the 'envs' command will show
you an error message.
The MEV can be also set by SETTINGS.INI as "max_env" identifier. It's defa
ult-set to ~999999 (infinite envs);

Environment Commands:

"dir (PATH)", - Will set the Current Directory, which is default-set to Alp
haBase HM7/;
"file [PATH]", - Will set the Current File. So, in the commands which suppor
t current file system,
if you do for example 'eboot' and curfile is 'example.pbp' (compatible) then
AB will automatically assign
the first parameter as the current file;
"tree (PATH), - Will show you all the subdirs of the given PATH, which is de
fault-set to: ms0:/;
"get (STRING/POS) - Gets a snapshot string in the snaplist. If the given STR
ING already exists in the snaplist,
If mode is Normal then this STRING will set as the current snapshot string,
otherwise the command returns an error;
If you enter a number (POS), then the command will set as the current snaps
hot string
the string which is indexed by POS in the snaplist;
"unget (STRING/POS)~(STRING2)" - Will remove the snapshot string STR
ING or the one indexed by POS If STRING2 isnt given;
Otherwise, It will replace the snapshot string given in the first param with
STRING2;
"find [ELEMENT]~(MODE)~(MODE2)" - Will check whether the given ELEMEN
T is loaded or not in the itemslist or in the macrolist
(which are identified by MODE, 0 for the first one, 1 for the second one).
If you use MODE 0, then you have to use

also MODE2 in order to identify the type of Element. It must be: 0 for Fonts
, 1 for Images, 2 for Sockets or 3 for Colors;
"key (KEY)~[INPUT]" - Will assign the given INPUT execution to the given
KEY (that in the command is default set to Circle);
"slist [PATH]~(MODE)" - Will set the Commands List (for the levelcheck),

the Macrolist or the Poslist, which are identified by MODE, as the specifi
ed PATH;
"env [STRING]~(MODE)" - Will set the main programme variables, like the
Name, the Author and the Version,
which are identified by MODE, as STRING;
"sys (MODE)~(COLOR)" - If MODE is 0, It will set the current System Color
which is default-set to white
otherwise It will set the Current Bacjground Color, which is default-set to
the R,G,B system-color
opposite values if mode is Normal, otherwise to black. The Color is identi
fied by COLOR/ID, obviously.
"goto (LINE)", - Go to the specified environment LINE;
"hlpr (MODE)" - Designed for coders. It will display you the name and the st
atus of the variables declared
in the script.lua between the comments: "-- Initializing Helper vars " and "
-- End" and also the SETTINGS.ini vars;
For view only the Normal Helper Vars, type "helper 0", else if you want to
view only the SETTINGS.ini vars then
type "helper 1". Otherwise, to see all, just type 'helper'.
Highly useful for debugging inshell homemade functions that require more v
ariables;
"usage [CMD]" - Will show you the usageprint of the given CMD command
;
"dbg" - Will process a simply but powerful debug, which consists in a defau
lt linelinking (by using 'goto' command)
and a nilcheck for all the Helper and the User Variables. If an Helper Var
is nil then a WARNING will be print on the screen;
"hlp (MODE)" - Will print you a Commands List if MODE is 0, else if a LUA
Functions List. The List are basic
if the Programme Mode is Normal, otherwise they are extended (with the pa
rameters viewer);
"cred" - Will show the Programme Credits;

AB Lexic and Basic Errors:

A.E.S.S.S. - (AlphaBase Environment Subscripts Support System) - refers to the
Alpha:Init, Alpha:setProgram, Alpha:setAuthor and Alpha:setVersion LUA functions, defined into "a-utils.lua".
So, All the subscripts are recognized into any AlphaBase Environment;
C.E.U. - (Commands Elaboration Unit) - refers to the "sendInput(str, prv)" function, which is defined into "a-utils.lua";
C.P.S. - (Complex Parameter Seperator) - refers to the seperator symbol to split Complex Functions params;
D.T.E. - (Directly Template Executing) - refers to the command which use the loadstring LUA's native function;
E.M.T. - (Environment Main Tree) - refers to the structure of dirs, subdirs of the AlphaBase HM7/ dir;
E.S.M. - (Environment Security Management) - refers to all the controls AlphaBase do whenever it's necessary. For example, to check
if a certain file to manipulate with the 'write' or 'read' command is AUF or not (and this case, cannot be manipulated),
to enable the repeating PASSWORD param system check in all the commands in which
you have to insert a new Password ('reg' and 'pass' for now);
F.P.S. - (File Parameter Seperator) - refers to the seperator symbol to split AUF Files (or other supported formats) params;
I.P.R. - (Input Param Require) - refers to the commands that require the INPUT forced param;
M.E.V. - (Max Environment Variable) - refers to the max environments allowed into the entire AlphaBase System;
P.S.E.R. - (Path Single Extension Require) - refers to the commands that support only an extension in one of their params;
C.S.M. - (Color Management System) - refers to the 'col' command and the main array which contain the colors.
- ERRORS:
100 -> "AlphaBase HM7/CEU.LUA/" doesnt exist
(However, you can in any case run the script without it,
but this will be really unworkable);
101 -> "AlphaBase HM7/a-utils.lua" doesnt exist;
102 -> "AlphaBase HM7/danzeff.lua" doesnt exist;
103 -> "AlphaBase HM7/danzeff/" doesnt exist;
104 -> "AlphaBase HM7/cmds/" doesnt exist;
105 -> "AlphaBase HM7/users/" doesnt exist;
106 or 99 ~> GENERIC.

Regards, Wesker.