Customization Techniques for the Allegro PCB Editor
cdnLive-1285

Frank Farmar

Cadence Design Systems
978-262-6497
fxf@cadence.com

1 Introduction

Will present the options available to users in Allegro PCB and APD to custimize and extend the products. The paper covers topics such as changing environment variables, symbol libraries, menus, scripts, reports and skill code. Configuration can be at both site (company) and user levels. Finally presents techniques for managing multiple Allegro releases.

This paper is of use to those programming to Allegro's PCB Editor, Allegro Package Designer and Allegro SI products. It does not apply to the Allegro Design HDL (formally ConceptHDL) or the PCB Router (formally known as SPECCTRA) environments.

2 Additional Information

Extensive documentation on user configuration options can be found in **cdsdoc** in the following areas:

- Allegro PCB and Package User Guide; Getting Started
- Setting User Variables (same book as above User Guide)
- Allegro PCB and Package Skill Reference Guide.
- SKILL Language User Guide
- SKILL Development Functions Reference

3 What's new in Customization for 15.5

Several new customization options were added in the SPB 15.5 release. They are:

- Allegro journal files can have an optional prefix setable via the environment variable *journal_prefix*. This is useful if multiple users plan on using the same project directory. This variable supports two predefined values, *user* and *host*. If value is *user* the the user's login is substituted and if the value is *host* the system hostname is substituted.

- New OS level variable, *ALLEGRO_PCBENV*, to allow user to specify location for the pcbenv directory. Overrides default <HOME>/pcbenv location. This variable must be set at the operating system level and is ignored if put into an allegro environment file.

- If your company has written a setup of reports these can now be included into the new html report dialog via the *axlReportRegister* Skill API. See the Allegro Skill reference manual for more information.

- A new Windows tool to allow switching between SPB releases; available from the 15.5 Cadence start menu (switchversion.exe).

4 Customization Methods

The Design Editor products provides multiple levels for customization:

- At the prroject and user level
- Site (customer wide) customizations via CDS_SITE
- Default Cadence environmen

Several methods can be mixed and matched for customization. These include;

- Function key bindings (aliases and “funckeys”)
- Allegro scripts and macros
- Skill Programming La nguage
- Allegro environment variables
- Menu replacement

5 User Level Customization

Many of the user level customization can be managed via dialogs provided in the SPB tool suite. The main tool is the Allegro environment editor (*enved*), that is used view and set Allegro environment variables. Other customization are done automatically during normal tool operation and saved to files (example the position and size of the tool windows) at tool exit

5.1 Your pcbenv Directory

The local pcbenv directory is used by the back-end tools to store the user preferences and other startup information:

- *env*: local environment file. Stores Allegro environment variables, alias and funckey definitions. Settings in this file typically superseed variables provided in the Cadence master and *CDS_SITE* files. Use the **enved** tool to manage the variable settings. Currently alias and funckey definitions must be managed via your favorite text editor.

  If editng the env file via a text editor, you should always put your edits above line with the comment "### User Preferences section". Anything below this line is reserved for the environment editor.

  Never copy the Cadence master env file to your local env file. This will cause compatibility issues when you switch to a new SPB release.

- *allegro*.ini: File used by tools to store main windows size and location, last directory (project) and control panel dock state and size. Also used to store default plotter configuration settings. This file is dynamically updated on tool exit. While you can set this file to read-only to disable updating information on exit, we provide the *nolast_directory* to prevent using the ini file for storing the last directory.

- <*program name*>.exe.ini: Obsolete, prior to 15.0, was used by programs to store toolbar configuration state. In the 15.0, release this data was moved to the Microsoft registry. Its location in the registry is (format of entries is not documented at this time by Cadence):

  [HKEY_CURRENT_USER\Software\Cadence Design Systems\<version>\<toolname>]

  Example 15.5 Allegro is:

  [HKEY_CURRENT_USER\Software\Cadence Design Systems\15.5\Allegro]

- <*program name*>*.geo*: Stores secondary window position and sizes. Secondary windows to the main program are forms (dialogs) and text windows (examples "show element", log file viewer and reports). This data is updated on program exit. The Allegro executable stores this data in

allegro.geo. In 15.5, you can reset the these locations by invoking the dialog *View->Customization->Display* and selecting the "Restore windows positions" button. Setting the file to read-only will prevent it from being updated.

- *<program name>*.mru: Stores list of most recently used files for a program. This is displayed in the program's File menu. Number of files is controlled by *recentFileList* environment variable. File is updated on program exit. File can be set to read-only to freeze the initial MRU file set.

- *allegro.strokes* stores user defined strokes. You can manage mouse shortcuts within Allegro by *"Tools->Utilities->Stroke Editor"*. Alternatively use the stand-alone **stroke_editor** tool.

- *allegro.ilinit*: Initial user Skill code loaded at program startup. Covered in last year's Allegro Skill paper at the given at the user group. A little known fact is if you create a file called apd.ilinit that can load a different set of Skill code when running the **apd** executable.

  This file is loaded early in Allegro's startup process before the database or menu is loaded. If you need to perform an operation when a database is open register a callback trigger via the **axlTriggerSet** API.

5.2 Additional Variables on UNIX

On UNIX, several additional directories and files are present under the user's home directory.

The *.mw* directory is used to store Mainsoft Microsoft Windows setup data. Prior to 15.0 this directory was named *windows*. Data stored in this location is the Microsoft registry database (*hklm* files), font caches, printer setup and other Windows mapping infomation.

When SPB tools migrate to a new version of Mainsoft, Mainsoft insures compatiblity with older releases (where necessary) by changing the filenames. OS environment variables of interest to override our default Mainwin setup are:

- *ALLEGRO_MWUSER_DIR* (introduced in 15.2 via an ISR), allows the specify a different location for the Mainsoft .mw directory. Useful if running Mainsoft based tools from different vendors.

- *MWUSER_DIR*, allows specifing different .mw user directory. Ignored if ALLEGRO_MWUSER_DIR is set.

- *MWSET_INPUT* (default followmouse). Alternative is to set "select" which uses a Windows convention for mouse focus.

- *MWSYSTEM_FONT_HEIGHT* (default 14), control size of font for menus.

If you delete the *mw* directory, it will be recreated the next time you run a Mainsoft based program but you may lose preferences. For example, Allegro stores any user modifications to the toolbar in the regristry file in this directory.

The *.cdsdoc* directory is used to store configuration data for the Cadence on-line help and documentation system (cdsdoc). None of this data is publically documented. If you delete the directory it will be recreated the next time cdsdoc is accessed. This directory also exists on Windows under *<my documents>/<user login>*.

The *.cdsplotinit* file is used to store Cadence plot serve information about plotters. See Cadence Plot Server documentation for file format.

6 Site (Company) Level Customization

6.1 CDS_SITE

Allegro looks at for configuration files in the following default locations:

- Cadence provided default configuration files: *<cdsroot>/share/pcb*
- Site (company) provided location (this is called *CDS_SITE*): *<cdsroot>/share/local/pcb*
- User preferences, typically: *<HOME>/pcbenv*
- Certain design files: *<current project directory>/<data type>*

By default, you can use the Cadence supplied location *(<cdsroot>/share/local*) for *CDS_SITE* or set this operating system variable to specify an alternative location. Do **not** set the CDS_SITE via the Allegro environment file. The default Cadence location provides an empty hierarchy for your configuration data so if you specify a different location, I suggest you copy the Cadence supplied hierarchy to seed your alternative location.

In addition to the *CDS_SITE* variable, you can use the variable, *ALLEGRO_SITE* to override the location of Allegro site data. *ALLEGRO_SITE* lets you locate the Allegro configuration files outside the standard Cadence site location, <CDS_SITE>/pcb. Site level customization does not require any changes to the Cadence installation hierarchy and minimizes the setup required to the users local environment. As with *CDS_SITE*, do not set the *ALLERGO_SITE* via the Allegro environment file.

The *CDS_SITE* gives larger sites the ability to provide there end users to their end users without requiring a special set-up for each user.

6.2 Default CDS_SITE PCB Hierarchy

Directory	File Extension	Comment
assembly	.arl, .rle	Design For Manufactoring rule files. This is the rule based DFx interface not the 15.5 on-line DFA checker (design data).
devices	.txt	Third party device files. Used by import logic program netin (design data).

Directory	File Extension	Comment
dfa	.dfa	Package spacing technology file. Feature introduced in 15.5 to perform on-line package to package spacing checks (design data).
extracta	.txt	Extract view files.
forms	.form	Allegro form files. Used with Skill programming environment.
icons	.bmp	Bitmaps that can be loaded into Allegro form files. Used with Skill programming environment.
menus	.men	Allegro menu files. Menu files present in this directory will superseed Cadence provided menus.
nclegend	.txt	NcDrill legend template files (design data)
padstacks	.pad	Allegro padstack databases (design data).
scripts	.scr	Allegro scripts
signal	.dat, .wave, .ibs, .mod, .ctl	Signoise models and control files (design data).
skill	.il, cxt, ils	Used to store skill programs. The file *example.ilinit* can be renamed to *allegro.ilinit* to automatically load all .il or .ils files in this directory.
symbols	.psm, .bsm, .osm, .ssm, .fsm	Symbols: package, mechanical, board, shape and flash symbols (design data).
tech	.tech	Technology files used to store constraints, user properties and stackup settings (design data).
views	.color	Visibility setting files.
xtalk	.xtb	Crosstalk table files (design data).

7 Customization by Operation

7.1 Allegro Scripts

A method for capturing repetivitve operations. Scripts can be used across projects if you store the files in one of the locations referenced via the SCRIPTPATH environment varaiable.

Scripting is a command language. There is no real logical constucts exist (e.g. looping, subroutines or conditionals) so it is not targeted for operations that require decisions. If you need logical constructs use the Skill language. You can call Allegro commands from Skill or call Skill from within a script.

Scripts also support a "macro" mode. When a macro is replayed the script will request an origin pick from the user and all coordinates the script enters into Allegro will be relative to that origin. If you examine a macro script you will notice a XXX command (which is what requests the origin pick from the user) and then all picks in the script will be of the "*ipick*" type. The *ipick* (or *ipick_to_grid*) are the commands Allegro uses to perform a relative pick.

The easiest method to create a script is to record a typical operation. You can then edit the script in a text editor to clean up the sequence of operations.

The following commands are useful in either developing a script or in tuning its operation:

- *scriptmode*: controls a set of script options. See the documentation for a full set. If scriptmode used in a script then it restores its original settings when the script terminates. The following are the most useful options:
    - *+invisible*: Any forms invoked by the script are not visible. This makes the script run faster and looks more professional. If the script ends and the form, opened by the script, is still active then it will be made visible.
    - *+echo*: Echos all script commands to the Allegro type-in area. Useful in debugging the script
- *replay* <*script*>: Start replaying script with filename "*script*". You can bind a script to a alias, put it in a menu file or call it from Skill via the *axlShell* API. You can also call a script from another script (scripts can be nested up to five levels deep).
- *record* <*script*>: Starts recording a script to filename "*script*".
- *stop*: Stops recording a script.
- *echo* <*message*>: Display a message to user in Allegro's type-in area.

- *prompt <message>*: Same as the *echo* command put displays a blocking confirmer window to the user with the message.

- *set noconfirm*: Prevents Allegro from displaying most blocking confirmers to user. This can help if you find your script sometimes needs to deal with a confirmer window. If you use this environment variable, you should remove all of the "*fillin yes*" lines in your script. It is also important

- *;* (the semicolon): This is not an Allegro command but part of the scripting language. You can chain several commands together on a single line, each command seperated by a semicolon. If you have a short sequence of commands that you wish to bind to a function key, you can use this method instead of binding a script to the key. For example (note the presence of the double quotes):

  alias F2 "command1; command2"

- *helpcmd*: Command that displays all Allegro commands.

7.2 Visibility Views

This is accessible via the "Views" dropdown under the Find Filter tab of the Control panel. It provides the ability to quickly toggle between a set of visibility settings. By default it automatically uses the artwork films defined in the database but you can use the *Color View Save* (under *View* menu) command to create external view files.

Views files are just Allegro scripts with the .color extension and located using the VIEWPATH environment variable instead of the SCRIPTPATH variable. This means you can deposit script files (using the .color extension) in the *<cds_site>/pcb/views* directory and have them show up in the View drop-down.

7.3 Environment Variables

Allegro environment variables are managed and documented via the *enved* tool. Many of these variables offer the user the ability to modify the default behavorior for various aspects of the tool.

Allegro also supports a set of dynamic variabless (shown in following table). These variables are automatically set at startup or changed when appropriate during tool operation. Users can take advantage of these variables but should not change them (exceptions to this rule are *allegro_site* and *cds_site*).

Variables	**Default**	**Comment**
allegro_install_dir	<cdsroot>/share/pcb	Location of the Allegro platform independant files. Set at tool startup.

Variables	Default	Comment
allegro_install_root	<cdsroot>	Root location of the Allegro install hierarchy. Set at tool startup.
allegro_install_tools	<cdsroot>/tools/pcb	Location of the Allegro platform files (the "bin" directories). Set at tool startup.
allegro_site	<cdssite>/share/local/pcb	Site specific files. Set at program startup unless user as already set variable. – User can set. --
allegro_type	pcb	Always pcb. Indicates the sub-directory under <cdsroot> where Allegro tools/data is located.
base	<starting directory>	Directory where tool was started.
cds_site	<cdsroot>/share/local	Location for site configuration files. – User can set. --
class	<active class>	Current active class shown in *Options* panel. Updated whenever this class changes.
global	<cdsroot>/share/pcb/text	Cadence base configuration files. Set at program startup.
globalpath	. <global>	Same as global variable but prefixes current working directory.
home	<user's home directory>	Obtained from the OS level *HOME* variable. If not set uses /tmp (UNIX) or user's registry setting for the user's "*Documents and Settings*" (Windows). The Windows registry fallback is new for SPB 15.5.
localenv	<home>/pcbenv	The location of the user's configuration directory. In 15.5 this can be overridden with the *ALLEGRO_PCBENV* variable.
localpath	. <localenv> <cdssite>/pcb <global>	Resolution path for configuration files. Set at program startup

Variables	Default	Comment
menuload	<current menu>	Current menu file in use by tool. Updated whenever menu changed.
module	<current design>	Current design name including extension. Legacy variable, replaced by *_module*. Updated on open and save of designs.
subclass	<active subclass>	Current active subclass shown in *Options* panel. Updated whenever this class changes.
telenv	<global>/env	Legacy variable for location of the Cadence master env file. User should never set this variable as it will cause issues when migrating to a new Cadence release.
tmp	<temporary directory>	Loction for temporary files. Inherited from OS settings but some users override the default by setting it their local environent file. In 15.5 on Windows, we will also fallback on the user's "*Documents and Settings*" temporary location if the variable is not set at either the env file or as an OS variable.
viewlog	<last log filename>	Name of last logfile generated. Can utilize with "*viewlog –last*" command.
__unix	set if UNIX; unset Windows	Set if running on UNIX. Used in the menu files
_module	<current design>	Current design name including extension. Updated on open and save of designs.
_module_base	<current design minus extension>	Current design name excluding file extension. Updated on open and save of designs.
_program	<current program>	Current program name. Set at program startup.

Variables	Default	Comment
sx1, sy1, sxlo, sylo, sxhi. syhi, sxn, syn	<cursor locations>	Used in conjunction with the stroke editor. Dynamically set when a stroke is done. Reflects upper left and lower right coordinates.

7.4 Keyboard Bindings

Allegro offers two keyboard binding styles; *aliases* and *funckeys*. The *funckeys* (function keys) command was a 15.2 enhancement. The two binding styles offer the same capability for function keys and control modified alpha-number keys where pressing the key immediately executes the bound operation. The difference between the two bindings is for the alphanumeric portion of the keyboard, you must press the Enter key invoke an aliased operation while the funckey binding does not require the pressing of the Enter key.

The funckey option only applies when the keyboard focus is in the Allegro graphics area. If the keyboard focus is in the Allegro type-in area the funckeys bindings operate the same as the aliases. This allows you to move the cursor to the type-in area if you want to type an Allegro command.

Example difference between alias and funckey:

alias r redraw you need to press 'r' then 'Enter' to redraw display

funckey r redraw pressing the 'r' key would redraw the display

Bindings also supports modifier keys, Shift, Control and Alt. The following prefix is used for these modifiers:

- **C** – control key modifier used with a function key (example CF2)
- **~** - Control key modifier used with an alpha-numeric key (ex. ~F or ~1)
- **S** – Shift key modifier used with a function key (example SF2)
- **CS** – Control-shift key modifier used with a function key (example CSF2)
- **A** – Alt modifier key. Note this is only supported with certain function keys. Windows reservers this modifier key when used with other keys such as the alpha-number keys and F4 (ex. AF2).
- The Alt key can also be used in conjunction with the Control and Shift keys.

- If the Shift key is used with alpha-numeric keyboard then the binding is to the upper case character and not to the ‘S’ prefix. For example binding the Shift-R to redraw would be ‘R’ not ‘Sr”.

The following keys and modifiers are reserved by the operating system and are not available for binding: F1, AF4, AF6, Ctrl-C, Ctrl-V, Ctrl-X and Alt keys.

The Page keys (home, end, etc.), Esc (Escape) and arrow keys are treated as function keys with respect to binding operations.

By default the arrow keys are bound to roam. You can change the roam increment with this binding by setting the *roaminc* variable. It comes with a default of 96 pixels. Settings, for best performance, should be in multiples of 16 pixels.

If you take advantage of funckeys to bind multi-key operations then you must remember that we will match the shortest key binding. For example, the following two funckey bindings:

```
funckey s  save

funckey sa save_as
```

will only result in the ‘s’ binding be availble. The ‘s’ binding hides the ‘sa’ binding. No such restriction exists with aliases since the ‘Enter’ key terminates the binding.

Allegro provides a default set of aliases in our master environment file. Currently we do not provide a user interface for modifing our definitions so you should use a text editor to add your preferred bindings either your local or site environment file.

It is also possible to remap the bindings depending upon where you are in your design cycle. To accomplish this create a separate script file with the set of bindings to accelerate each major design phase. For example, you could create three binding files for placement, routing and post-process. You could then modify the menu to allow replaying the approproiate binding script.

7.5 Menus and Toolbars

Users can supply their own Allegro menu, overriding the menu provided by Cadence. The menu files have a “.men” extension and are located via the *MENUPATH* environment variable.

The current tool has no ability for a user to just do an append or delete from the standard Cadence supplied menu. You must replace the menu. Thus, you must intergrate your menu changes with those provided by Cadence when adoption a new Allegro release.

All Allegro board products use the *allegro.men* file found at *<cdsroot>/share/pcb/text/cuimenus*. Other menus for programs such as the symbol

editor, APD, etc. can be found in the same directory. The menu file syntax is documented in the *Allegro Skill Reference Manual.*

Toolbar customization is only provided at the user level. We currently do not support the abilitity to customize toolbars at the site level. Also we do not currently allow the user to supply their own icons and commands. Toolbar perferences are saved on a per-release basis so you need to redo your toolbar settings for each new release.

7.6 Reports

Starting in 15.5, you can add your own reports to the standard Allegro reports dialog. In the Allegro Skill manual, see the documentation associated with axlReportRegister.

8 Managing Multiple Allegro Releases

You typically face two problems in managing multiple Allegro releases; accessing the correct software version and managing incompatibilties in your configuration files.

8.1 Switching Releases on Windows

Starting in 15.5, SPB provides a switchversion.exe program. It is accessible from a Windows Start Menu. It allows easy retargeting of the Windows PATH variable to a support different SPB releases.

In 15.5, the Cadence installer adds Cadence executable directories to the Windows PATH variable using a %CDSROOT% to abstract the Cadence root directory.

Even if you do not plan on immediately adopting the 15.5 release you can obtain this program to manage switching older releases.

Appendix A Example To Change Grids

The following example shows have to use an Allegro script, variables and funckeys bindings to quickly change Allegro's Etch grid settings. Place the script in a one of the directories present in *SCRIPTPATH* environment variable.

The script (name is grid.scr):

```
# Script filename: grid.scr
# a '#' indicates a comment
# This script provides the ability to change grid settings of etch
# layers. Requires variable gridvalue to be set for desired grid
#
# The scriptmode command keeps the form invisible
scritpmode +invisible
setwindow pcb
define grid
setwindow form.grid
FORM grid all_etch all_etch_x_grids $gridvalue
FORM grid all_etch all_etch_y_grids $gridvalue
FORM grid done
# end script
```

Add following function key bindings to your local environment file:

```
funckey g5 "set gridvalue = 5; replay grid"
funckey g0 "set gridvalue = 10; replay grid"
funckey g12 "set gridvalue = 12; replay grid"
```

A key item to remember of funckeys is that the first funckey to match is what is executed. For example, if you define a *g1* funckey, the *g12* in above the example would not be accessible. The same holds true, defining a *g* funckey would disable all of the above keys.