Component, Model and Library Concepts

Related Resources
Net Ties and How to Use Them (PDF)

This article explains Altium Designer components, models and libraries, and their relationships. Approaches for identifying and managing component-to-library relationships are explored, as well as the search sequence for locating models and the options that make these searches more effective.

Components form the basic building blocks of electronic products. During the design capture and implementation process a component can be represented in different ways: as logical symbols on the schematic, as footprints on the PCB, as SPICE definitions for simulation, as signal integrity descriptions for analysis, and as three-dimensional descriptions for 3D component and PCB visualization. While not every one of these representations is necessary for a component, there is a minimum definition required before a component can at least be placed into a schematic design. What makes up a starting component is established, and how adding additional properties, parameters, and models make implementing the component in various parts of the design process possible is discussed.

Components - the Basic Building Blocks

A component is the general name given to a constituent part that can be placed into an electronic design whether it is the schematic capture or the PCB layout. Components can have multiple representations for each of the phases of design capture, and thus may be referred to by different contextual names depending on the current implementation. Flexibility is a requirement for component definition so that all of its information and linkage can be easily adapted and transferred from one phase of design to the next.

Because electronic design begins with the schematic capture, a component has minimum requirements for placement in any schematic design - at the very least with its own name in a schematic library. It may also contain pins and graphic symbols in single or multi-part fashion, and even have alternative display options.
Once placed on the schematic design and assigned a graphical representation, the component is then more commonly called a symbol. Because it is graphical, the symbol includes drawing objects that define its physical shape and pins that define the electrical connection points, or logic. You may also hear the symbol referred to as a logical symbol during schematic design capture.

Their initial simple definition makes it easier and more flexible for components to be adapted to represent very complex electronic entities. Certain components, such as a resistor network or relay, can be drawn as a series of separate parts which can be placed independently on the schematic. These are referred to as multi-part components and their individual pieces are simply called parts.

The different phases of design are called domains - these are the specific types, groups or areas of representation. In the Altium Designer environment the valid domains include PCB layout, SPICE simulation, signal integrity analysis, and 3D.

As previously mentioned, there are some distinct contextual terms used during the PCB layout and design. A footprint describes the model that represents the component on the PCB layout. A footprint is thus a grouped set of PCB pads and component overlay shapes that define the space required to mount and connect the component on the board layout. Once mounted on a PCB, the component is then considered to be a physical component.

It's worth noting that in most cases, the logical symbol also represents the physical component. When this is the case, the references for each will be the same. However, one exception is that in the case of components placed from a database library, the record in the external database represents the physical component (and the symbol is effectively a model then).

Let's discuss in more detail the concepts of how components are defined and described, the different types of representation they can have, and how more specialized types are supported in Altium Designer.

**Component Properties**

We've established that the symbol with minimal definitions and linkage is the essential starting point for any component in Altium Designer. This allows for greater flexibility for the component to be represented in different ways later during other parts of the design and capture process. Altium Designer supports different approaches to building components that adapt to whatever your design requirements are.
Various attributes, linkages and descriptions can be easily defined for any component to create unique properties for it through the Component Properties dialog - allowing you to create exactly the type of representation and implementation needed for every phase of your design.

Component Properties dialogs will differ depending on the design editor you are using (schematic, PCB, etc.).

**Component Types**

Components can be built several different ways in Altium Designer. It’s worth discussing some of the standard and non-standard types as well as multi-part components.

**Common Graphic, Different Component**

**One Component Symbol for Each Physical Component**

This type of representation is ideal for any component where the logical symbol is the same as the physical component such as integrated circuits. The component would include the specific representation, or model (described in greater detail below) such as the PCB footprint, simulation or 3D modelling information.
One Symbol for Graphically Equivalent Components

Sometimes components are logically equivalent but have slightly different component specifications. An example would be a logic gate that is available in a variety of logic families, for example a 74ACT32 and a 74HC32. In this case the symbol is drawn once, and then another name, or alias, is defined for each equivalent component required. Component aliases are added in the Schematic Library editor panel. Component aliases can be thought of as one component, with multiple names.

One Symbol for a Type of Component

PCB components have some special considerations with respect to component types where the component may need to be linked to multiple PCB footprints depending on the physical layout requirements of the design. An example of this would be discrete components such as resistors where the component has a value attribute that is defined when the component is placed on the schematic, rather than in the library.

Common Component, Different Graphics

Altium Designer supports multiple symbols for the same component. For example, you may have one client that requires their symbols drawn using traditional drawing shapes while another requires the symbols drawn in accordance with a specific standard. Or you may require many different symbols for the same component. You can define additional graphical representations for a symbol that are stored with the component by adding a new mode, either from the schematic library editor Tools menu, or by using the Mode toolbar (Figure 3).

Multi-part Components
In some instances it is more appropriate to represent the one physical component using multiple symbols, for example each resistor in a resistor network, or the coil and contacts of a relay.

Additional parts are added or removed using the commands in the library editor Tools menu. Each part is then drawn individually, and pins are added accordingly.

**Non-standard Component Types**

Not all components are destined to be mounted on the assembled PCB, not all components are required in the Bill of Materials (BOM), and not all items that are mounted on the PCB need to be represented on the schematic. Altium Designer supports non-standard component types through the Component Type property, set in the Component Properties dialog in the library or schematic editor.

For example, the presentation and readability of your schematic might be enhanced by including a chassis-mounted component that is wired to the PCB. If this component was not required in the PCB BOM, then the component type can be set to Graphical. A graphical component is not included during schematic electrical verification, it is not included in the BOM, nor is it checked during schematic to PCB synchronization. In this case the Component Type is set to Graphical.

Another special class of component would be a test point - this component is required on both the schematic and the PCB, it should be checked during design synchronization, but is not required in the BOM. In this case the Component Type is set to Standard (No BOM).

Another example of a special component kind would be a heat sink - typically it is not shown on the schematic and is not required to be checked during schematic electrical verification, but must be included in the BOM. In this case the component type is set to Mechanical.

**Component Parameters**

Parameters are a way of defining and associating additional textual information to the component. This can include electrical specifications (i.e., wattage or tolerance), purchasing or stock details, designer notes, or references to component datasheets. This information is included by adding parameters to the component either during component creation in the library editor; once the component has been placed on the schematic (using a DBLink file); or automatically during placement when placing from a database library (DBLib or SVNDBLib).

Adding a component parameter to an individual component is easily done by going directly through the Component Properties dialog for that component:

**Adding Parameters to an Individual Component**
parameter information appears in the workspace through Component Properties.

Any parameters defined in the Parameters section are also made available in the Match By Parameters region of the Annotate dialog. This is particularly useful if you later wish to group specific parts of a multi-part component, using a unique parameter that you have defined and included for those parts.

**Adding Parameters to a Component Library**

While the example above shows the manual addition of parameters to an individual library component using the Component Properties dialog (Figure 6), you may require a more streamlined approach to add parameters to a library of components. In such instances the Parameter Manager dialog would be the better option.

![Parameter Manager dialog](image)

**Referencing Datasheets as Parameters**

There may be times when you need to access your own reference material from within a design project using a component datasheet. Altium Designer provides two options for linking from a component on the schematic sheet to reference datasheets which is established through the addition of component parameters. The first option allows you to use the F1 button to access a specific referenced document. The second option allows for multiple references and uses the right-click context menu.

If a component includes a parameter using the system-reserved name of HelpURL, then the URL will be accessed when the F1 button is pressed while the cursor is hovering over the component or its entry in the Libraries panel. The URL can actually be a web address, a text file, or a PDF file.

The parameter’s value can point to a document and even include a specific page number in a PDF (Figure 7).

**Single Linked Document - F1 Access**
Figure 7. Here a HelpURL parameter has been added to a schematic symbol from the Component Properties dialog. Given the value of \\Help\CR0118 FPGA Generic Library Guide.pdf#page=93 results in the referenced PDF file being opened at page 93 when the F1 button is pressed when the cursor is over the placed component.

**Multiple Linked Documents - Right-click Access**

This second technique enables you to define and support multiple links to one or more reference documents in a right-click context menu by pairing parameters and using the system-reserved name of ComponentLinknURL:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Example Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st parameter</td>
<td>ComponentLink1URL</td>
</tr>
<tr>
<td></td>
<td>C:\MyDatasheets\XYZDatasheet.pdf</td>
</tr>
<tr>
<td>2nd parameter</td>
<td>ComponentLink1Description</td>
</tr>
<tr>
<td></td>
<td>Datasheet for XYZ</td>
</tr>
<tr>
<td>1st parameter</td>
<td>ComponentLink2URL</td>
</tr>
<tr>
<td></td>
<td>C:\MyDatasheets\AlternateXYZDatasheet.pdf</td>
</tr>
<tr>
<td>2nd parameter</td>
<td>ComponentLink2Description</td>
</tr>
<tr>
<td></td>
<td>Datasheet for Alternate XYZ</td>
</tr>
</tbody>
</table>

Any number of links can be defined using the same parameter pair, except with the number incremented. When you right-click on a component that uses datasheet linking, a **Reference** menu entry will appear in the Context menu, in it you will find an entry for each component link, as shown in Figure 8.

Component-to-datasheet linkage can also be used when you are browsing components in the Libraries panel - press **F1** or right-click on the component name in the panel to access the linked documents/URLs.
For more information about adding other types of component parameters, refer to the Creating Library Components Tutorial.

Figure 8. Right-click on the placed symbol to access the datasheet links.

Altium Designer 14.3.12 introduced support for clicking on a ComponentLink or HelpURL parameter in the generated PDF, and opening the referenced PDF. For this to work, the link to the file must be specified using the Adobe Device Independent Path syntax. An example of this format is: the Windows path C:\MyFolder\MyFile.pdf, is specified as /C/MyFolder/MyFile.pdf. The leading forward slash indicates that this is an absolute path, if there is no leading forward slash then the path is relative. You may also need to adjust the Protected Mode (disable) or the Protected View settings for this to work (go to the Security (Enhanced) page in the Adobe Acrobat Preferences dialog to configure these options).

Models - Specialized Component Representations

Remember that domains are the type, group, or area of component representation that can be captured as part of the design process in Altium Designer. A model thus is the implementation representation of the component that is useful for a particular domain. This could be as a footprint on the PCB, as a SPICE definition for simulation, as a suitable signal integrity description for signal integrity analysis, or as a three-dimensional model for 3D visualization in either or both the legacy 3D viewer and the DirectX-based 3D visualization engine. While a component is not required to have a model attached to it in order to be placed in a schematic alone, it cannot be implemented in any other domains until it does.
Model Libraries are a collection of component representations, and are described in further detail in the library section below. It’s worth explaining the fundamentals of how model mapping information is stored with the component.

**Fundamentals of Model Mapping Information**

At the schematic stage, the design is a collection of components that have been connected logically. To test or implement the design it needs to be transferred to another modeling domain, such as simulation, PCB layout, signal integrity analysis, etc.

Each domain needs some information about the component, and also some way to map that information to the pins of the schematic symbol. Some of this domain information resides in model files, the format of which is typically predefined. Examples of these include IBIS, MDL and CKT. Some of the information does not reside in the model files, for example the SPICE pin-mapping and net listing data must be stored and managed by the system.

All of the necessary domain information is contained within the schematic component, which stores a separate interface to each model that has been added to it. In effect, the complete model is the combination of the model mapping information stored in the component, and the domain modeling information stored in the model library.

![Figure 9. Information on how to model the component in each domain is stored in the model files. Here we see how the symbol hooks to the individual implementation models.](image)

Components may have models for multiple domains, and can also have multiple models per domain, one of which will be set as the current model.
Note that IBIS signal integrity models and VRML or IGES 3D models must be imported into Altium Designer format models before they can be used. IBIS models are imported directly in the Signal Integrity Model dialog, which opens when you add an SI model to a component. VRML and IGES models must be imported into a PCB3DLib before they can be added to a schematic component.

**Options for referencing models**

PCB3DLib is a legacy library type, it is not used for Altium Designer 10/12, Altium Designer 2013.

Whenever you add a model to a component you have the option of defining how tightly you want to control where the model is searched for. Although they vary slightly from one model type to another, the model editor dialogs generally include these options: Any, Library Name, Library Path, and From integrated library.

- **Any** - searches all available libraries for a matching model.
- **Library name** - only searches valid libraries of this name for a matching model.
Library path - only searches a valid library of this name in this location for a matching model.

Integrated library - draws the model directly from the integrated library used to place this component. The integrated library must be available in a valid location.

Libraries - Collections of Components, Models, or Both

Components and/or their models are almost always stored as collections, or libraries, for more efficient use during the design process. A library may only contain the constituent elements that can make up the component and not necessarily the complete definition. For example, a descriptive name such as model library indicates that the library contains only models whereas a schematic library contains only schematic symbols.

Sometimes the constituent elements of a library are stored in a format other than an Altium Designer file or under version control. This is more common when more than one engineer may be working on a design, or in different locations. A database library is one where all symbol references, model linking and parameter information is stored in an ODBC or ADO-compliant database, or an Excel spreadsheet. A version-controlled database library is simply an extension of the database library where the symbol and models are stored under version control (like Subversion).

Each of these libraries and the appropriate file extension is described in more detail below.

Library Types

Model Libraries (*.MDL,* CKT,* PcbLib)

The representative models for each domain type are stored in model libraries, and can also be referred to as a model container. The grouping and organization of models may vary between domains. In some domains, such as SPICE (.MDL,* .CKT), the storage is typically one model per file. In other domains however, models are normally grouped into library files according to a user-defined categorization, such as PCB footprints grouped into package-type libraries (*.PcbLib).

Schematic Libraries (*.SchLib)

These libraries contain the schematic symbols and their respective links (can be thought of as a pointer only but not the actual model itself) to any model definitions that reside in separate model libraries.

Creating a Schematic Source Library of an Active Project

Very useful if you want to create an exact working library or archive of your finished design, this ability creates a schematic source library of all the components that have been placed in the source schematic documents of the active project. After launching Design » Make Schematic Library all schematic source documents for the active project will be opened, if not already open, and a library document, ProjectName.SCHLIB, will be automatically created and added to the project.
**Integrated Libraries (*.IntLib)**

An integrated library in Altium Designer is one where the source symbol, footprint, and all other information (e.g. SPICE and other model files) are compiled into a single file. The advantage of compiling into an integrated library is that all component information is available in a single portable file.

During compilation checks are made to see how relationships are defined, to validate the relationship between the models and the symbols and to bundle them into a single integrated library. Components and models in an integrated library are not available for editing, unless the library is decompiled (opening the*.IntLib to extract the sources), offering portability and security.

All of Altium Designer's 70,000+ components are supplied in integrated libraries, from which the source libraries can be extracted at any time if required.

Since all models are packaged into the integrated library, only one file needs to be available to the project, or moved when the project is relocated.

**Creating and Reusing a Working Library or Archive of a Finished Design**

You can create an integrated library of all the components and their linked models that have been placed in the schematic documents of the active project - very handy for creating project libraries or libraries sent to you from a third party. It is also accessible from the Design » Make Integrated Library when a schematic design is active.

**Database Libraries (*.DBLib,* SVNDBLib)**

These libraries store all symbol information, model links, models and parameters in a format external to Altium Designer such as ODBC, ADO, or an Excel spreadsheet.

Each record in the database thus represents a component, storing all of the parameters along with model links, datasheet references, or other component information. The record can include links to inventory or other corporate component data.

Database libraries come in two flavors - non-version-controlled (database library) and version-controlled (SVN database library). The only difference between the two is the location of the symbol and model libraries, containing the referenced symbols and models. The difference can be summarized as follows:

- Database Library (*.DBLib) - symbol and model libraries are stored in a directory on your hard disk or other local/network medium.
- SVN Database Library (*.SVNDBLib) - symbol and model libraries are stored under version control in a Subversion repository.
- Database Libraries provide the ability to place components directly from an external company database. In a database library, all the detail that makes the component complete is stored in the database itself - in this sense the symbol is only a graphical representation or another model.
Figure 11. Database libraries store all the information in the database, and retrieve it during component placement.

When a component is placed from a database library, its parameter and model information is created on-the-fly, using the corresponding fields in the matched database record and in accordance with defined mapping in the intermediary link file. One or more of these parameters will then be used to maintain an ongoing link back to the database, enabling future design synchronization after placement.

**Searching For and Identifying Your Components**

Having portability of designs and libraries has become a common work requirement - it's more the norm that libraries are in separate locations from the design itself or there may be alternate sets of libraries used for a single design. Sometimes the designer may just wish to take his work home where he's using a local copy of the company library. Because such a flexible way to reference source libraries is needed, it's critical to be able to control the source of the components and identify that they are the right ones.

Altium Designer offers both the flexibility and control to be able to easily switch between locations of reference libraries, and identify and validate that you are using the correct components from the
The reason for this is that when you place a component from a library you need to remember which library (or table in the case of a DBLib) that component came from. Knowing where a component came from then becomes very valuable as a record for design management. If you have ever tried updating components and didn't have the original libraries and had to work within a restrictive design environment, then you'll appreciate how frustrating this problem can quickly become! A solution that thus provides both definable levels of control that can suit any work environment's configuration and help maintain the integrity of the design components is needed.

![Figure 12. Here we see that Component A is the same component referenced in both Library A and Library B. You can change the Library Path, Library Name for a component, or Table Name for a DBLib to switch between the source components.](image)

**Valid Search Locations**

As part of the validation process for identifying components, Altium Designer searches in specific locations for models and libraries anytime that you perform an operation that requires them. The search will also take into account any options that you have specified for controlling how you reference models through a model editor (as discussed earlier). For example, when you perform a circuit simulation, the SPICE model linked to each component is retrieved and used by the XSPICE simulation engine. Another example would be transferring a design from schematic capture to PCB layout. During this process, the footprint for each component is retrieved and placed onto the PCB. Any models/libraries found in these locations are referred to as the available libraries - meaning the set of models/libraries are available for use in a project or the design environment.

You can review the models/libraries available to the current project in the Available Libraries dialog. Click the Libraries button in the Libraries panel, or select **Design » Add/Remove Library** to open the dialog (Figure 13).

**Available Libraries Dialog**
For all models not tied to an integrated library, the search order proceeds from left to right through the tabs of the Available Libraries dialog (Project, Installed, and Search Path). In fact, since the available libraries can be ordered within this dialog from top to bottom, the entire search sequence is intuitive and easy to set up.

Valid locations of available models/libraries that can be searched therefore include:

- **Project** - these models/libraries are linked and available only to the project and its documents. The advantage of this approach is that whenever the project is opened the model/libraries will be available. The disadvantage is if the models/libraries are not stored in the project folder structure, they can be forgotten if the project files are moved (like from one PC to another).

Tip:

While the Altium Designer environment offers flexibility and control over your model/library locations, it does require you to use the correct file extension for each model type. For example, a footprint cannot be found unless it is in a file with a *.lib or *.pcblib extension. Similarly, a SPICE .SUBCKT will not be found unless it is in a *.ckt file, nor will a SPICE .MODEL if it is not in a *.mdl file. Whenever a model search does not yield a match, an error will appear in the Messages panel.

- **Installed** - these models/libraries are associated with the Altium Designer environment and so components are available to all open projects. Any model/library added to the Installed Libraries list can now be installed relative to a nominated path, described in greater detail below.

- **Search Path** - these models/libraries are made available to the project by defining a search path in the Search Path tab of Options for Project dialog (also conveniently accessed through the Paths button of the Search Path tab in the Available Libraries dialog). Each search path defines a folder, and can include sub-folders if the Recursive option is enabled. All model and library files found down the search path will be valid. Note that retrieving models using search paths can be slow if there are a large number of files in the search path folder(s).

While the Altium Designer environment offers flexibility and control over your model/library locations, it does require you to use the correct file extension for each model type. For example, a footprint cannot be found unless it is in a file with a *.lib or *.pcblib extension. Similarly, a SPICE .SUBCKT will not be found unless it is in a *.ckt file, nor will a SPICE .MODEL if it is not in a *.mdl file. Whenever a model search does not yield a match, an error will appear in the Messages panel.

**Relative Path Installation for Libraries**

Any libraries added to the Installed Libraries list can now be installed relative to a nominated path.
Available in the Installed tab of the Available Libraries dialog, this makes it easy to switch between different sets of libraries and control the source of components in your design. Changing the entry for the path will automatically reload those existing libraries in the list that are found at the new location.

**Library Activation and Deactivation**

![Available Libraries dialog](image)

Each library currently added to the Installed Libraries list (accessed on the Installed tab of the Available Libraries dialog) can also be 'Activated' or 'Deactivated'. This allows you to visually identify quickly which sets of libraries you are using with your design. Simply toggle the associated Activated option accordingly (as shown in Figure 14).

A Deactivated library is treated as though it had been uninstalled, but remains in the list so that it may quickly be activated, based on your design requirements. A library that is not found along the specified relative path cannot be activated.

**Component Properties Dialog**

In addition to being able to search for and identify components through the Available Libraries dialog, additional levels of control are also found through the Component Properties dialog (Figure 15).

**Changing the Library Name at Component level**

The next level down for identifying your components is being able to change the library name for a component itself. Done within the Component Properties dialog there are three levels of component identifications:

- **Design Item ID** - the first library component found, within the current set of activated libraries
the first library component found, within an activated library whose name matches that of the specified library name, and whose component name matches that of the design component on the sheet, will be used.

- **Use Database Table Name** - the first library component found, within an activated library whose name matches that of the specified library name, within a table within that library whose name matches the specified table name, and whose component name matches that of the design component on the sheet, will be used.

Figure 15. Controlling the different levels of component identification through the Component Properties dialog.

- Component name (Design Item ID), Use Library Name and Use Database Table Name can all be specified in the Library Link region of the properties dialog for any placed component. The use of Library Name and Table Name can then be selectively enabled/disabled using available options.

**Changing Table Names from a DBLib**

For design components that were originally placed from integrated libraries, you can change the base reference library to that of a newly converted DBlib or SVNDBlib simply by selecting all components in the design and:

- Disabling the **Use Library Name** option (you would need to ensure that the new DBlib/SVNDBlib has been added to the Installed Libraries list and made active, and that the previous integrated libraries are removed or deactivated.
- Leaving the **Use Library Name** option enabled, but entering the name of the DBlib/SVNDBlib instead. The Table Name could be specified if all selected components belong to that same table, or could be left blank/disabled, meaning the first match found in any table in the
database would be used in each case.

- To verify that the correct library is indeed being used as reference for a design component, click the Validate button - found in the **Library Link** region of the Component Properties dialog. A dialog will appear displaying the path and library in which the first match for the design component has been found.

Source URL: https://techdocs.altium.com/display/ADOH/Component,+Model+and+Library+Concepts