Component, Model and Library Concepts

Summary

This article defines components, models and libraries, and their relationships. The search sequence for locating models is explained, as well as options that make this search more restrictive for specific models.

Article
AR0104 (v2.0) June 07, 2006

Components are the basic building blocks of an electronic product. During the design capture and implementation processes each component needs to be represented in different ways: as a logical symbol on the schematic, 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 description for a 3D representation of the finished PCB.

Not all of these representations are necessary for every component, but the logical symbol is the starting point. Every component must be defined, at the very least, with its own name in a schematic library. It may contain pins and graphic symbols in single or multi-part fashion, and even have alternative display options. As such, it may be placed in any schematic design. However, until models have been added to the component, it cannot be implemented in any practical sense.

Definitions

Component: general name given to an object which can be placed into a design.

Symbol: general name given to the graphical representation of the component that is placed on the schematic. The symbol can include drawing objects that define the shape, and pins that define the electrical connection points.

Physical Component: representation of the component that will be mounted on the PCB.

Logical Symbol: the symbolic representation of the Physical Component, as it appears on the schematic.

Part: certain components, such as a resistor network or a relay, can be drawn as a series of separate parts, which can be placed independently on the schematic (referred to as a multi-part component).

Model: a representation of the component which is useful in some practical domain.

Footprint: name used for the model that represents the component on the PCB layout. A footprint is a grouped set of PCB pads and component overlay shapes that define the space required to mount and connect the physical component on the board layout.

Logical versus Physical…

For a standard Altium Designer component the logical symbol in the Altium Designer schematic library also represents the physical component, so for this component the Logical Symbol and Physical Component references will be the same.

This is not the case for a component placed from a database library, in this situation the record in the database represents the Physical Component, with the Altium Designer schematic symbol being just the Logical Symbol.

Domain: type, group, or area of representation. In the Altium Designer environment the valid domains include PCB layout, SPICE simulation, Signal Integrity analysis, and 3D.

Library: a file containing a collection of components, or a collection of models, or both.

Model Library: a file containing a collection of component models.

Component Library: a file containing a collection of schematic components.

Integrated Library: a file containing a collection of schematic components, and their associated models.

Database Library: component library where all symbol reference, model linking and parameter information is stored in an ODBC or ADO based database, or an Excel spreadsheet.

Version-Controlled Database Library: extension of the standard database library, but with symbol and model libraries stored under version control (Subversion).

Fundamentals

At the schematic stage, the design is a collection of components which 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, and so on.

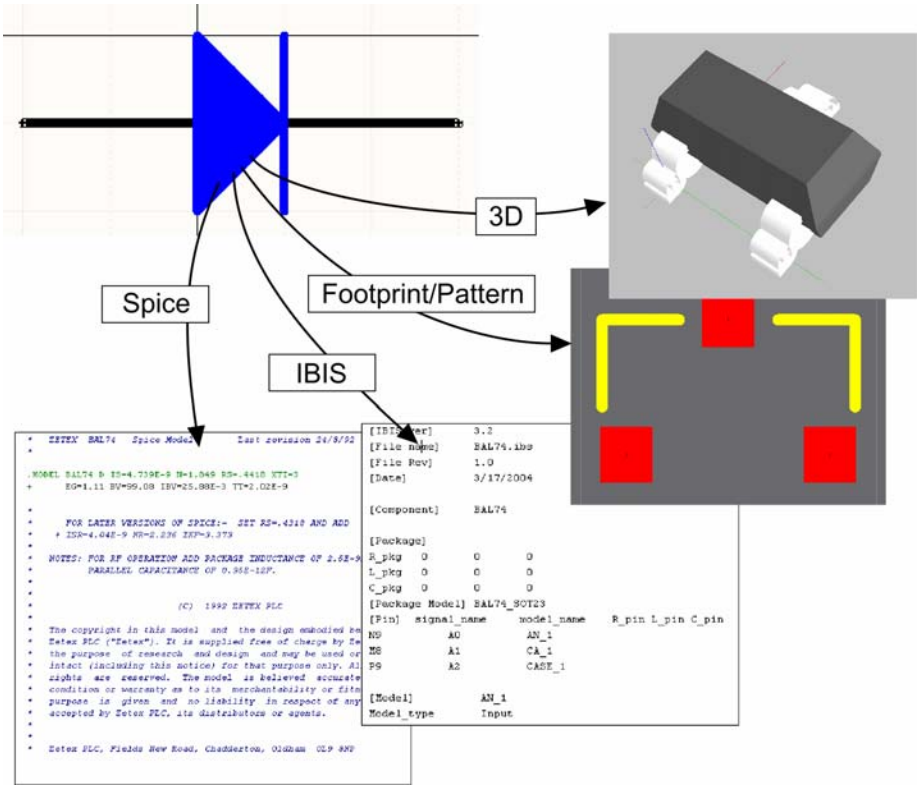


Figure 1. Information on how to model the component in each domain is stored in the model files.

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 pre-defined. 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 netlisting data must be stored and managed by the system.

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.

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.

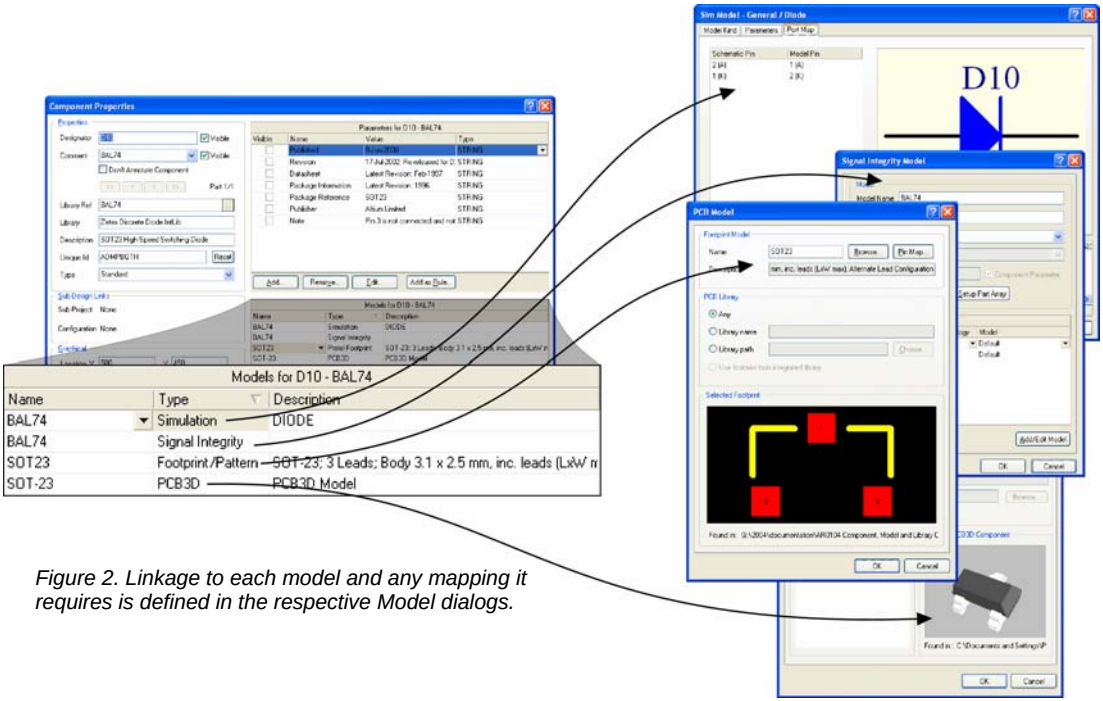


Figure 2. Linkage to each model and any mapping it requires is defined in the respective Model dialogs.

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.

Library Types

There are four types of libraries that can be used in the Altium Designer environment.

Model libraries – the models for each domain are stored in “model containers”, typically called model libraries. In some domains, such as SPICE, where the storage is typically one model per file, they are also referred to as model files (*.MDL, *.CKT). In other domains, the 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 – these contain schematic components and their model interface definitions (*.SchLib). Each model interface definition contains a link to its respective model library.

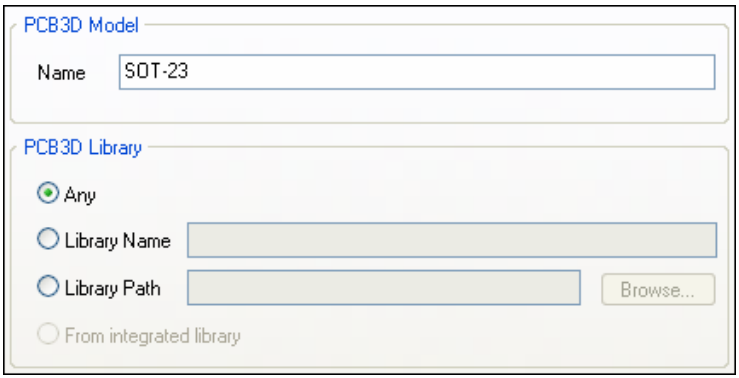
Integrated libraries – these are a set of schematic libraries, which, together with their linked model libraries, are 'compiled' into one file – the integrated library (*.IntLib). The advantage of compiling into an integrated library is that all component information is available in a single portable file. Integrated libraries cannot be edited without extracting the sources, and recompiling.

Database Libraries – a library where all symbol reference, model linking and parameter information is stored in an ODBC or ADO based database, or an Excel spreadsheet. Each record in the database represents a component, storing all of the parameters, along with links to the models. The record can include links to inventory or other corporate component data. There are two types of database library, depending on whether the symbol and model files are stored on a local or network drive (Database Library (`*.DBLib`)), or stored under version control in a Subversion repository (SVN Database Library (`*.SVNDBLib`)).

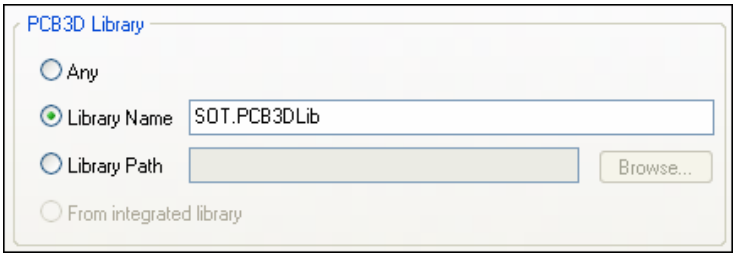
Referencing a Model

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:

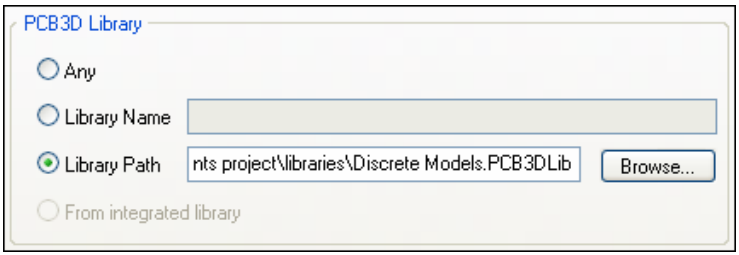
Any – searches all available libraries for a matching model.



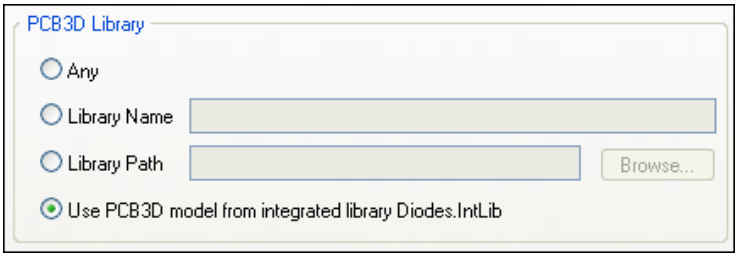
Library name – only searches valid libraries of this name for a matching model.



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.



Locating the Model – Valid Search Locations

Each time you perform an operation that requires a model, the system will search for it in the valid locations, according to the reference criteria defined in the previous section. 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 the design from schematic capture to PCB layout. During this process, the footprint for each component must be retrieved and placed on the PCB.

The valid locations of models that can be searched include:

Project Model/Library – models and model libraries can be added into the project. Like all project files, model/library files are only linked to the project, so you can link a model/library to many projects. The advantage of this approach is that whenever the project is opened the models will be available. The disadvantage is that models/libraries that are not stored in the project folder structure can be forgotten if the project files are moved from one PC to another. Project models/libraries are only available when you are editing a document that belongs to the current project.

Installed Library – these libraries are associated with the Altium Designer environment. Components in installed libraries are available to all open projects.

Project Search Path – models/libraries can be made available to a project by defining a search path in the *Options for Project* 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 is a large number of files in the search path folder(s).

Checking the Available Models/Libraries

The set of models/libraries found in these locations are referred to as the *available libraries* – meaning the set of models/libraries available for use in this project.

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.

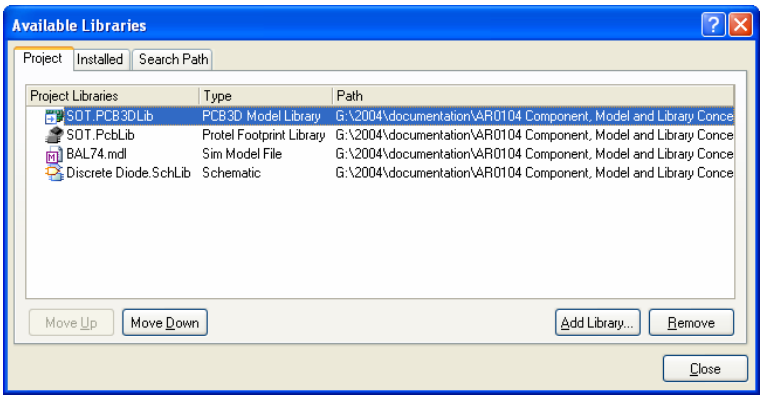


Figure 3. The Available Libraries dialog shows all models/libraries that are available to the active project.

General Search Order

The rule is to stop searching for a model as soon as a match is found. For all models not tied to an integrated library, the search will proceed in this order:

- Project Libraries
- Installed libraries
- Search Paths

You will notice that this order is followed from left to right through the *Available Libraries* dialog. 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.

While the Altium Designer environment offers flexibility and control over your model 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.

Integrated Libraries

Integrated libraries are a special class of library, where the schematic symbol library and all referenced models are compiled into a single file. The advantage of an integrated library is portability and security. 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. Components and models in an integrated library are not available for editing, unless the library is decompiled (open the IntLib to extract the sources).

An Integrated library is built by:

- Creating an integrated library package
- Adding schematic symbol libraries
- Referencing the required models from each component symbol
- Setting the integrated library project options, including the location of the compiled output
- Compiling the integrated library package to produce the integrated library (`Name.IntLib`)

If a component symbol is placed from an integrated library then the system will only attempt to retrieve the referenced models from that integrated library. Note that the integrated library must be available in the valid search locations when the system is attempting to retrieve a model.


For more information on integrated libraries see the *Enhanced Library Management Using Integrated Libraries* article.


For step by step instructions on creating a component library, creating component symbols, and referencing models, see the *Creating Library Components* tutorial.


For instructions on creating an integrated library, see the *Building an Integrated Library* tutorial.

Database Libraries

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.

Each record in the database details the symbol and model names and locations, as well as information that is to be included in the placed component as parameters, such as a reference to a datasheet, or other company component data. The Altium Designer symbol in the schematic library (which is referenced from the database) is just a symbol, the graphical representation. The database record describes the physical component that is mounted on the board.

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.

The database library document (`*.DBLib` or `*.SVNDBLib` respectively) is where the connection to the external database is defined. For a version-controlled database library, you also specify the link to the version control repository.

The database library document behaves as an intermediary link file between the component records in the database and the Altium Designer components placed in a design. It handles the mapping from database entries to Altium Designer symbols, models and parameters. It is also where you define parameter key field linking, which can be a single key field (for example a part number), or multiple key fields (by defining a **Where** clause). After placement, it is this key field linking that facilitates the synchronizable link back to the source database record.

Locating Symbols and Models

The symbol and model names are stored as part of the record for each component, in the external database. You can also include information as to where the symbol and model libraries are located, either by entering an absolute path or a relative path. Typically, you would only specify the name of the library in which the required symbol or model resides or, better yet, not specify location information at all.

In these last two scenarios, you can specify – within the database library document – paths along which the symbols and models may be searched for. For a DBLib, you specify search paths to a directory on your hard disk for example. For an SVNDBLib, you specify root directories for the symbol and footprint model libraries within the version control repository.

Component Placement

A database library is added to the Available Libraries, presents in the **Libraries** panel, and is used in the same way as any other library. Each table in the database presents as a separate 'library' in the **Libraries** panel, with each component entry corresponding to a physical component record in that table. The panel essentially behaves as a viewing portal into the database. Figure 4 illustrates the correlation between a record for a component in a database table and the entry for that component seen from the **Libraries** panel, once the database library has been added to the list of Available Libraries.

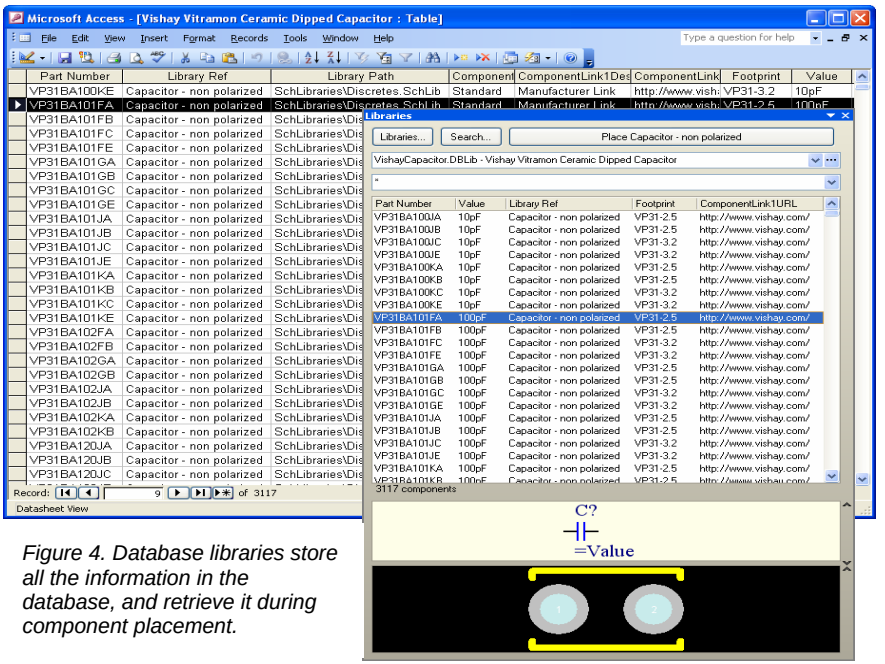


Figure 4. Database libraries store all the information in the database, and retrieve it during component placement.

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 synchronization after placement.

Components placed from a database library can be fully updated using the standard **Update From Libraries** command.

Database information that is not included in the Altium Designer component can be included directly in the Bill Of Materials.

For more information on database libraries refer to the *Using Components Directly from Your Company Database* application note.

For more information on using an SVN Database Library, refer to the *Working with Version-Controlled Database Libraries* application note.

For more information on ensuring synchronicity between placed components and a source database, refer to the *Keeping Components Up-To-Date* application note.

Properties of the Component

The basis of a component is the symbol. Rather than requiring each component symbol represent a unique component, the system supports different approaches to building components.

Common graphic, different component

One component symbol for each physical component – ideal for components such as integrated circuits, where each component symbol represents an actual physical component. The component would include suitable models, such as the PCB footprint, the signal integrity model and the 3D model.

One symbol for graphically equivalent components – useful when components are logically equivalent, but have a different component specification. 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 component symbol is drawn once, and a *component 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 – useful for discrete components, such as resistors. The component could link to multiple PCB footprints. The component value is defined when the component is placed on the schematic, rather than in the library.

Common component, different graphics

The system also supports multiple graphics for the same component. For example, you may have one client that requires their component symbols drawn using component traditional drawing shapes and another that requires the component symbols to be drawn in accordance with a specific standard. To define additional graphical representations for the one component, add a new component Mode, either from the library editor **Tools** menu, or using the **Mode** toolbar.

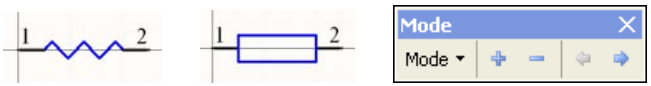


Figure 6. Use the Mode feature to define multiple graphical representations of the one component.

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, and not all items that are mounted on the PCB need to be represented on the schematic. The system supports this through the **Component Type** property, set in the *Component Properties* dialog (in the library or on the schematic).

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

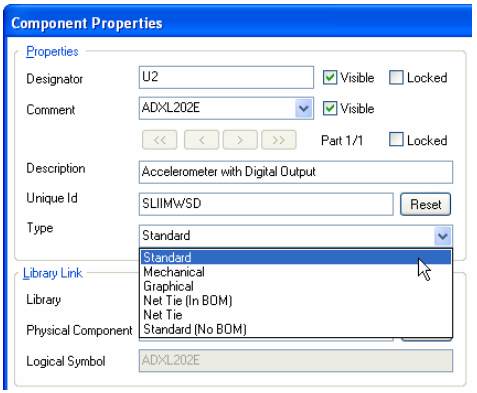


Figure 5. Set the component type for special component requirements.

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 heatsink – typically it is not shown on the schematic (but could be), 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**.

Use the What's This help for details on the various component types, or press F1 when the *Component Properties* dialog is open.

Component parameters

Typically there is a variety of text information that must be included with a component. This could include such things as component electrical specifications (eg wattage or tolerance), component 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).

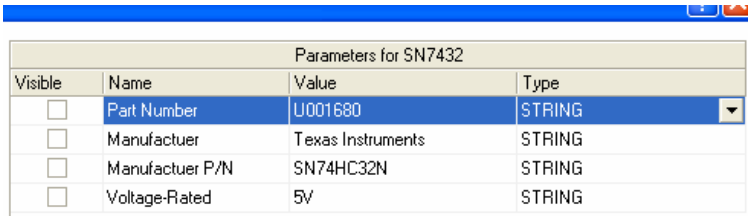


Figure 7. Add component parameters to include required component detail.

For more information on linking from placed components to a company database, see the *Linking Existing Components to Your Company Database* application note.

For more information on database libraries refer to the *Using Components Directly from Your Company Database* application note. For information on using a database library where the symbol and model libraries are under version control, refer to the *Working with Version-Controlled Database Libraries* application note.

Referencing a datasheet from a component

There may be times where you want (need, or prefer) to access your own reference material from within a design project, for example datasheets. To facilitate this, Altium Designer provides two methods for linking from a component on the schematic sheet to reference information. Linkage is established through the addition of specific component parameters.

Single linked document - F1 access

To reference a single document such as a datasheet from a component, using the F1 key as the method of access, include the parameter *HelpURL*. The parameter value points to the document, and

can include a page number for a PDF. For example, the HelpURL value of *CR0118 FPGA Generic Library Guide.pdf#page=93* would result in the referenced PDF file being opened at page 93, when the F1 button is pressed when the cursor is over either the placed component, or its entry in the Libraries panel. The reference can be to a local document (include the path if the document is not placed in the \Help folder of the installation), or the reference can be to an internet URL. If the component does not include a HelpURL parameter the default component help topic will appear.

Multiple linked documents – Right-click access

This feature enables you to define and present named links to one or more target reference documents in a part's right-click context menu, using the ComponentLink parameter pairing. Multiple ComponentLink parameter pairings can be defined. To use this feature, simply add and configure the two parameters for each pairing as follows:

First parameter - used to define the target document:

```
Name = ComponentLinknURL
```

```
Value = target document name
```

Again, the parameter's value must include the full path if the document does not reside in the \Help folder of the installation. Specify the page number for a PDF document as required.

Second parameter - used to define the link caption that appears in the menu:

```
Name = ComponentLinknDescription
```

```
Value = any meaningful description
```

Note: For both parameters in the pairing, n is an integer value, allowing you to define multiple parameter pairings. For each ComponentLink pairing, ensure that n is the same for both constituent parameters.

To access the defined link, simply right-click on the placed part in the main workspace. The entry for the link will appear in the **References** sub-menu.

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 drawn individually, and pins are added.

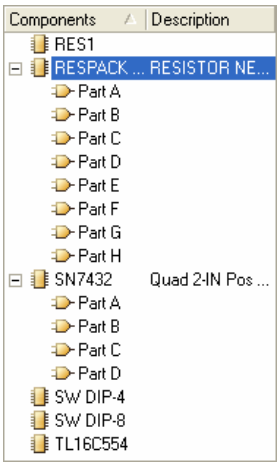


Figure 8. A component can have one or more component parts.

Power pins

Component power pins can be shown visibly, like any other component pin, or they can be hidden. Hidden pins, that have their electrical type set to Power, are automatically connected to the net defined by the **Connect To** property in the *Pin Properties* dialog.

To display hidden power pins during component design, enable the **Show Hidden Pins** option in the library editor **View** menu.

For step by step instructions on creating a component library, creating component symbols, and referencing models, see the *Creating Library Components* tutorial.

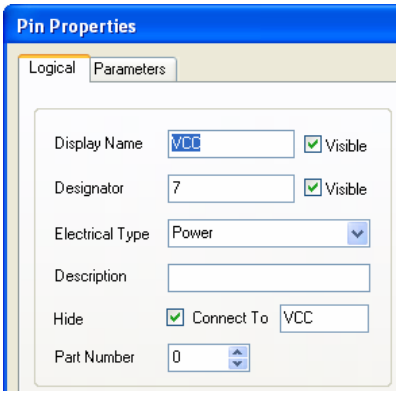


Figure 9. For hidden power pins the Connect To property defines the net that this pin must connect to.

Revision History

Date	Version No.	Revision
9-Dec-2003	1.0	New product release
28-Jul-04	1.1	Added images and extra detail to text
01-Jul-2005	1.2	Updated for Altium Designer SP4
30-Nov-2005	1.3	Updated for Altium Designer 6.0, database libraries added
07-Jun-2006	2.0	Updated for Altium Designer 6.3, database libraries section rewritten to incorporate version-controlled database libraries.