Altium Designer Summer09.

Практические подходы к организации библиотек и структуры проектов.

Структура и организация библиотеки на основе базы данных

В цикле статей будут сформулированы требования к библиотекам и другим файлам проектов, которые сложились у автора в результате длительного опыта работы в пакете проектирования Altium Designer Summer09. В этой статье рассмотрен вопрос создания библиотек на основе базы данных.

Владимир Пранович, к. т. н.

pranovich@bsu.by

Основным здесь является такая организация библиотек и служебных файлов (именно они используются во всех проектах), которая позволяет легко находить необходимые компоненты, унифицировать их параметры и получать выходные отчеты с минимумом их доработки. Как правило, эта проблема возникает после создания первого десятка проектов, когда разработчики обнаруживают в собственных библиотеках множество компонентов с разным графическим отображением, по-разному названными идентичными параметрами и т. д. При этом каждый раз приходится модифицировать существующие библиотеки. Автор предлагает свое видение данной проблемы и приводит практическое решение ряда вопросов.

Итак, начнем с формулирования требования к библиотекам. Каждый компонент проекта должен иметь:

1. Обязательно полную информацию о его названии или критических параметрах, однозначно определяющих его применение.
2. Обязательно информацию о посадочном месте и виде монтажа.
3. Обязательно информацию о производителе и обозначение компонента по классификации потребителя.
4. Обязательно краткое описание компонента для облегчения поиска по библиотеке.
5. Информацию об альтернативных производителях.
6. Информацию о поставщике комплектующего изделия.
7. Ссылку на модели компонента для трехмерного моделирования.
8. При необходимости ссылку на модели для электронного моделирования и иные ссылки.
9. Желательно информацию о наличии компонента на складе и его местоположении там, рекомендации по применению этого компонента в новых разработках.
10. Иную информацию.

Из перечисленного следует, что компонент должен обладать разнообразными параметрами и широким диапазоном их значений. Учесть и задать значения всего объема параметров непосредственно при создании графических изображений достаточно проблематично, так как этот процесс занимает много времени. Вызывает определенные затруднения и необходимость отслеживания изменений при редактировании параметров компонентов. Единственным выходом в такой ситуации является организация библиотек в виде базы данных, которая легко позволяет вводить и редактировать текстовые параметры, не затрагивая графических объектов компонентов. Как организовать такую базу, автор рассказал в [1], однако там обрисован общий подход к созданию базы данных и приведено описание работы только с библиотекой в виде базы данных. В данной статье рассмотрим эти вопросы в более широком аспекте, включая нюансы работы и в графических редакторах библиотек, и настройки службы подготовки выходных данных.

Структура библиотеки

Структура библиотеки представлена на рис. 1.

Главным элементом является файл **2009.DbLib**, где **2009** — имя файла, **DbLib** — расширение, указывающее, что мы будем создавать библиотеку на основе базы данных. Название этого файла будет использоваться в качестве имени библиотеки. В этом файле содержатся ссылка на базу данных (в нашем случае это база в Access) и свойства синхронизации тех параметров, которые будут добавляться, модифицироваться или удаляться при внесении компонента в схему. Все это более подробно будет рассмотрено в отдельных разделах статьи.

Второй, не менее важной частью является сама база (в нашем случае — файл **DB2009.mdb**). Именно в нее будут введены все параметры компонентов и их значения, а также, ссылки на графические объекты компонентов и их модели.

Сами графические элементы компонентов, как и их модели, сгруппированы по типам и хранятся, как правило, в отдельных тематических библиотеках или в виде отдельных файлов в специальных папках. В частности, на рис. 1 предложен следующий вариант организации файлов библиотеки.

Графические изображения компонента, сгруппированные по типам в нескольких файлах:

1. D_Common.SchLib — библиотека, в которой собраны изображения стандартных элементов микросхем (логика, усилители и т. п.). Такие элементы могут быть использованы для отображения разных компонентов, нумерация и назначение выводов которых совпадают.
2. D_Uni.SchLib — библиотека, в которой собраны изображения специализированных элементов микросхем (микроконтроллеры, драйверы и т. п.), которые могут быть использованы для отображения конкретного компонента определенной фирмы.
3. RLC.PcbLib — библиотека, в которой собраны типовые изображения пассивных компонентов, таких как резисторы, конденсаторы, индуктивности и т. п.
4. Connector.SchLib — библиотека, в которой собраны изображения различного рода соединителей.
5. V.SchLib — библиотека, в которой собраны изображения полупроводниковых элементов.
6. Micelangio.SchLib — библиотека, в которой собраны изображения компонентов, не предназначенных для включения в перечень элементов (например, различного рода контрольные точки, знаки, реперы, радиаторы, просто графические знаки и иные элементы), и другие компоненты, не попавшие в вышеперечисленные типы библиотек.

Естественно, данная градация приведена как пример, и вы вправе добавлять иные библиотеки с другой направленностью по типам.

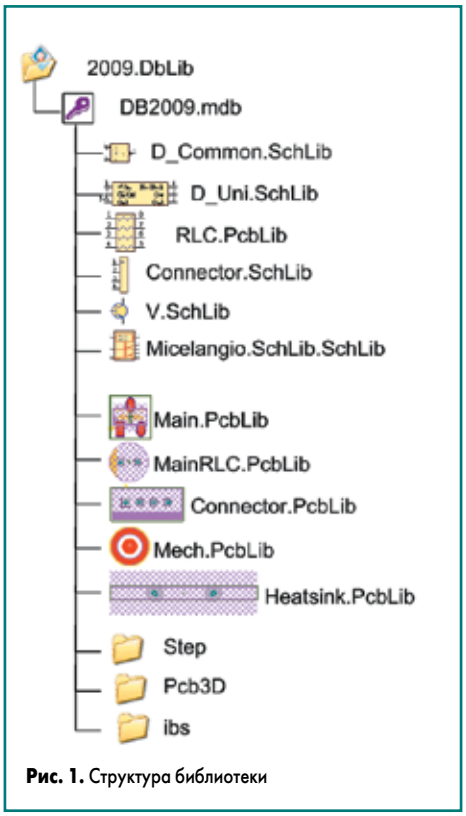

Рис. 1. Структура библиотеки

Графические изображения посадочных мест также рекомендуется разделить по типам, например так, как показано на рис. 1:

1. Main.PcbLib — библиотека, в которой собраны типовые посадочные места для микросхем и прочие компоненты с числом выводов не менее трех.
2. MainRLC.PcbLib — библиотека, в которой собраны типовые посадочные места для резисторов, конденсаторов и других пассивных элементов, как правило, с двумя выводами. Однако сюда включены и специализированные корпуса для многовыводных компонентов указанных типов.
3. Connector.PcbLib — библиотека, в которой собраны посадочные места для различного рода соединителей, держателей карт и т. п.
4. Mech.PcbLib — библиотека, в которой собраны механические элементы для монтажа на плату, такие как штыри контрольных точек, фиксаторы, монтажные отверстия и т. п.
5. Heatsink.PcbLib — данная библиотека выделена отдельно для изображения на печатной плате мест для радиаторов.

Для различного рода моделей отведены отдельные директории, в частности:

1. Step — для хранения трехмерных моделей, полученных в других пакетах или взятых с сайта производителя в формате STEP.
2. Pcb3D — библиотека, в которой собраны трехмерные модели предыдущих версий пакета.
3. Ibs — эта и другие подобные директории предназначены для хранения моделей описания работы компонентов, которые могут быть подгружены к графическому элементу компонента для моделирования работы электрической схемы.

Создание проекта

Для удобства работы вначале создадим отдельный проект, куда внесем ссылки на все файлы, имеющие прямое отношение к создаваемой библиотеке. Это позволяет при редактировании или добавлении компонентов легко ориентироваться в множестве файлов библиотек. Итак (рис. 2):

1. Командой **File=>New=>Project=>Integrated Library** создадим новый проект библиотеки, в нашем случае **2009.LibPkg**, и сохраним его в выбранной папке.
2. В панели **Project** наводим указатель на надпись созданной библиотеки (**2009.LibPkg**), нажатием правой кнопки мыши вызываем команду **Add Existing to Project…** и последовательно добавляем в проект необходимые библиотеки компонентов и посадочных мест. Заметим, если бы библиотеки находились в одной папке, можно было бы сразу сделать проект на основе папки, что объединило бы первый и второй пункты.

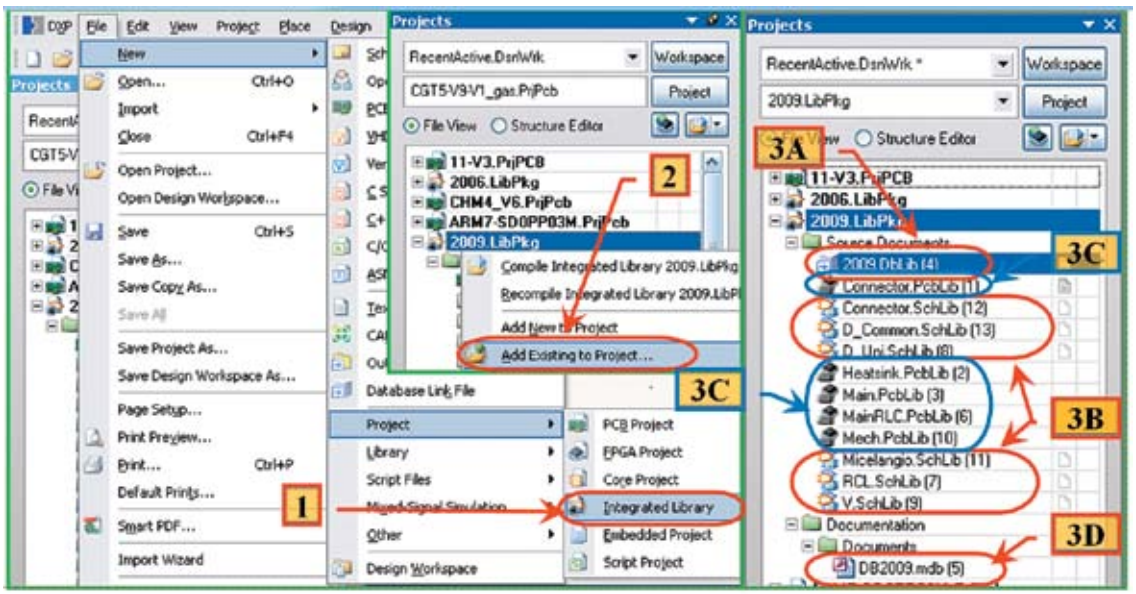

Рис. 2. Создание проекта для файлов новой библиотеки

3. В примере, описываемом в данной статье, используются следующие типы файлов:
   а) **2009.DbLib** — файл описаний передачи параметров из базы данных в проект;
   б) ***.SchLib** — библиотеки графических изображений компонентов;
   в) ***.PcbLib** — библиотеки посадочных мест компонентов;
   г) **DB2009.mdb** — база данных параметров и ссылок в Access.

Отметим, что нет необходимости держать все библиотеки в проекте. Сюда рекомендуется добавлять только те из них, которые будут редактироваться в процессе использования. Связь между библиотеками определяется ссылками в **DB2009.mdb** и не требует внедрения конкретной библиотеки в проект. Также в проект примера не включены файлы (кроме основной базы **DB2009.mdb**), редактирование которых производится средствами иных программ.

Описание библиотеки DbLib

Перейдем к описанию порядка работы с файлами библиотеки. Начнем с основного — с файла, создаваемого программой Altium Designer (расширение имени файла DbLib), который, собственно, и является библиотекой на основе базы данных. Подробно работа с данным файлом описана автором в [1]. Здесь повторим только основные моменты в контексте организации построения и взаимосвязи всех библиотек и файлов, объединяемых базой. Далее будет приведен процесс создания базы на основе существующих библиотек компонентов (*.SchLib), библиотек посадочных мест (*.PcbLib), непосредственно параметров компонентов из базы Access. Впоследствии будет показан процесс добавления ссылки на другие модели компонентов и их описания. Отметим, автор базируется на собственных библиотеках, уже оптимизированных для такого подхода, особенности организации которых будут описаны далее.

Откроем файл (в примере — **2009.DbLib**) и установим в нем следующие настройки (рис. 3):

1. Подключим базу данных в формате **Access**. Первым делом устанавливаем флаг **Select Database Type**. После этого в выпадающем меню, находящемся под данным флагом, выбираем тип базы данных. В нашем примере это база **Microsoft Access**. Наконец, нажимаем кнопку **Browse** и указываем путь, где хранится база данных. Заметим, что можно установить флаг **Store Path Relative to Database Library**. Это позволит указать относительный путь нахождения базы данных. Такая функция удобна, если вы работаете на разных компьютерах и постоянно синхронизируете файлы между ними. В этом случае достаточно, чтобы весь проект находился на одном диске, и нет необходимости каждый раз при смене компьютера указывать новое положение базы данных.

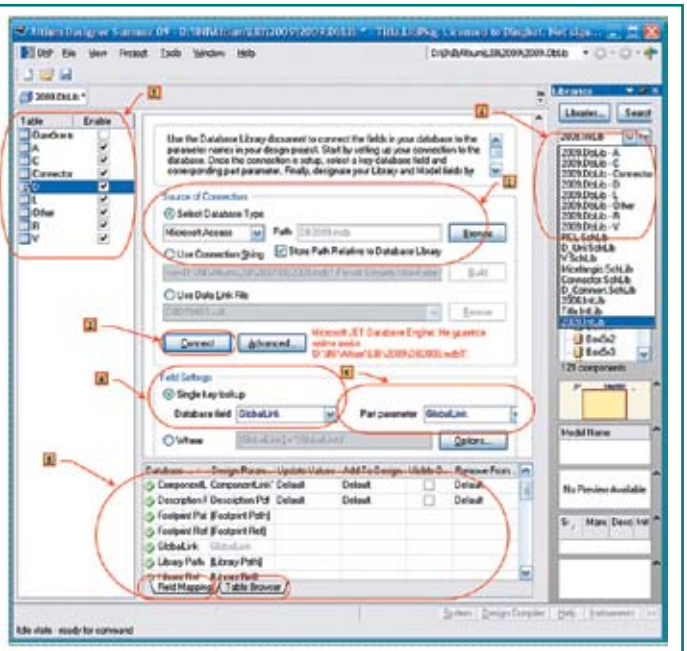

Рис. 3. Подключение базы данных

2. Теперь, после указания базы данных, следует нажать кнопку **Connected** для открытия непосредственно базы данных.
3. Заметим, база данных у нас уже существует. Как ее создавать, описано автором в [1], однако здесь рассмотрим более подробно ее модификацию для примера, описываемого в настоящей статье. Если у вас нет собственной базы данных, для начала вы можете загрузить любую базу из папки **…\Program Files\Altium Designer Summer 09\Examples\Cis\Example database.mdb**. После открытия базы данных станет доступным список таблиц. Следует установить флаг **Enable**, чтобы выбранная таблица выступала как библиотека компонентов.
4. Именно по названию файла **2009.DbLib** и по именам таблиц образуются названия библиотек компонентов после инсталляции библиотеки **2009.DbLib**. В нашем примере это:
   а) **2009.DbLib-A** — библиотека законченных устройств, таких как индикатор, световые панели, клавиатура и т. п.;
   б) **2009.DbLib-C** — библиотека для элементов типа «Конденсатор»;
   в) **2009.DbLib-Connector** — библиотека соединителей;
   г) **2009.DbLib-D** — библиотека интегральных микросхем;
   д) **2009.DbLib-L** — библиотека для компонентов типа «Катушка индуктивности»;
   е) **2009.DbLib-Other** — библиотека для компонентов, не попадающих под выделенные категории;
   ж) **2009.DbLib-R** — библиотека для элементов типа «Резистор»;
   з) **2009.DbLib-V** — библиотека для полупроводниковых элементов.

   Заметим, что названия таблиц не совпадают с соответствующими названиями библиотек графических изображений компонентов и их посадочных мест. И это обсудим далее в примерах организации соответствующих библиотек.
5. В данном окне отображены список **Field Mapping** параметров базы данных и свойства их синхронизации с проектом. Здесь же можно посмотреть и непосредственно значения параметров базы, открыв таблицу **Table Browser**. Но эту опцию мы рассматривать не будем, так как и просмотр базы данных, и ее заполнение удобнее проводить посредством **Microsoft Access**, с помощью которого база и создается.
6. Установим связь между проектом и базой данных. Так как мы создаем и базу, и библиотеку сами, то проще такую связь произвести по одному параметру, который имеет в своей таблице уникальный буквенно-цифровой номер. В нашей базе имя такого параметра для всех таблиц едино и имеет значение **GlobalLink**, однако для каждой таблицы оно может быть и индивидуальным. Отметим, что связь проекта устанавливается для каждой таблицы базы данных в отдельности и нижеперечисленные операции следует провести для всех таблиц (на рисунке показано для таблицы **D**). Итак, в этом случае:
   а) устанавливаем флаг **Single key lookup**;
   б) в выпадающем меню **Database field** выбираем из всего списка параметров базы данных параметр **GlobalLink**;
   в) параметр автоматически перенесется в поле **Part Parameter**.

Мы не будем усложнять систему синхронизации проекта и базы данных и поэтому не станем изменять значение в последнем поле. Однако имейте в виду: вы можете сопоставить имя уникального поля базы данных (в примере **GlobalLink**) с любым другим имеющимся полем библиотеки графического изображения. Более того, применив опцию **Where**, вы можете построить запрос и получить уникальный номер по нескольким полям базы данных. Однако такой подход хорош только для случая, если вы подключаете готовую базу, взятую с сайта производителя или поставщика компонентов, которую нежелательно редактировать.

Рассмотрим более подробно настройку параметров синхронизации данных между базой и проектом. Отметим, что Altium Designer позволяет при формировании отчетов брать как параметры, так и их значения непосредственно из базы данных. Поэтому нет необходимости держать все параметры непосредственно в компоненте (в параметрах графических элементов на схеме), чтобы не увеличивать размер файлов. Непосредственно в параметры графического изображения компонента следует передавать только те, которые явно отображаются на схеме. Синхронизацию остальных следует ограничить и передать только те, которые необходимы для однозначного определения компонента, и те, что использует Altium Designer для иных целей. Рассмотрим список **Field Mapping** параметров подробней и настроим свойства синхронизации (рис. 4):

1. Настроим свойства синхронизации по умолчанию. Отметим, эти свойства будут действовать при синхронизации для всех параметров базы данных, за исключением тех, для которых мы укажем индивидуальные свойства. Командой **Tool=>Option** откроем окно настроек Default Action.
2. В данном окне установим следующие свойства синхронизации:
   а) **Update Values => Update**. Заменять значения параметров в проекте значениями одноименных параметров из базы данных. Действительно, значения параметров в базе постоянно контролируются и являются более свежими.
   б) **Add to Design => No Add**. Не добавлять в проект новые параметры из базы данных. В базе может быть много параметров, не имеющих прямого отношения к параметрам компонентов в проекте, и их не следует вносить в проект.
   в) **Remove from Design => Do not remove**. Не удалять из проекта параметры, значения которых отсутствуют у одноименных параметров базы данных. Действительно, отдельные параметры и их значения могут быть добавлены непосредственно в проекте и иметь отношение только к конкретному проекту, и нет смысла, в таком случае, их удалять.
3. Укажем пути нахождения библиотек и моделей. Автор для этого использует одну директорию, однако можно иметь и несколько. Для добавления новой директории следует указать путь к ней (на рис. 4 — **D:/INI/Altium/LIB/2009**) и нажать кнопку **Add**. В частности, для указания относительного пути следует установить флаг **Add/Update As Relative Patch**. В этом случае запись в соответствующем окне примет вид «**../2009**» и ссылка будет действовать на всех компьютерах с вашими настройками.
4. Обратимся теперь к тем параметрам, на которые следует обратить особое внимание. Подробное описание их со ссылками автор приводил в [1], здесь отметим только их настройку. Вначале укажем те параметры (см. стр. 8 документа AP0133 Using Components Directly from Your Company Database.pdf), названия которых зарезервированы и используются для специальных целей. Отметим, что свойства синхронизации для них недоступны, так как они дают ссылки на графическое изображение компонента, его посадочное место и иные модели компонента. Такие параметры в столбце **Design Parameter** взяты в квадратные скобки. Эти ссылки всегда передаются в проект, и при поиске компонентов в библиотеке именно по ним вам будет представлен вид графического элемента, посадочного места и модели в панели **Library**. В приведенной базе используются следующие параметры:
   а) **Description**. Специальный параметр описания графического изображения компонента.
   б) **Footprint Path**. Имя и путь к библиотеке посадочного места.
   в) **Footprint Ref**. Имя посадочного места в библиотеке.
   г) **Library Path**. Имя и путь к библиотеке графического изображения компонента.
   д) **Library Ref**. Имя графического изображения компонента.
   е) **PCB3D Path**. Имя и путь к библиотеке трехмерной графики посадочного места компонента.
   ж) **PCB3D Ref**. Имя модели трехмерной графики посадочного места компонента графического изображения компонента. Отметим, что в последних версиях пакета есть возможность внедрять данную графику непосредственно в посадочное место и нет необходимости передачи ее из базы данных.
   з) **GlobalLink**. Этот параметр — уникальный номер, который мы определили выше. И он не может быть сопоставлен для синхронизации ни с одним специальным или другим параметром из проекта.
5. Особо отметим, что можно любой параметр базы сопоставить со значением специального параметра в проекте, и сделать это можно в столбце **Design Parameter**, для чего необходимо выбрать нужное значение специального параметра в выпадающем меню.
6. Укажем теперь те параметры из нашей базы данных, которые следует добавить к проекту и обновлять там:
   а) Параметр **ValueSCH**. Он введен с целью отображения надписи возле графического изображения. Это может быть как наименование микросхемы, так и значение определяющей величины компонента. Например, значение сопротивления резистора или емкости конденсатора. Автор рекомендует не использовать для этого типовой параметр **Value**, так как он применяется в моделях для обозначения идентичных величин, но при этом требует иного формата записи. Впрочем, если вы намереваетесь использовать данный параметр только в графическом редакторе схем, можете игнорировать эту рекомендацию. Для параметра **ValueSCH** установим следующие свойства синхронизации. Во-первых, в столбце **Update Value** установим значение **Update**, что означает: этот параметр всегда будет обновляться из базы данных. ВНИМАНИЕ! Для входа в режим редактирования свойств синхронизации параметра следует навести указатель на нужную ячейку в таблице, сделать клик левой кнопкой мыши и выбрать нужное значение из выпадающего меню. Во-вторых, в ячейке столбца **Add To Design** в соответствующей строке установим значение **Add**. Так мы установим режим, когда этот параметр всегда будет добавлен к компоненту проекта, даже если его значение не определено в базе данных. В-третьих, данный параметр всегда будем отображать на схеме и для этого установим флаг в столбце **Visible On Add**. В-четвертых, в столбце Remove **From Design** установим свойство **Not Remove**. Это позволит сохранить значение данного параметра, определенное в библиотеке типа ***.SCHLib** или непосредственно в проекте, если это значение не определено в базе данных.
   б) Параметр **NoteSCH**. Данный параметр введен для отображения дополнительной надписи (если это требуется) возле графического изображения. Как правило, он будет использоваться для указания значений вспомогательных величин параметров компонентов. Этот параметр следует всегда обновлять и добавлять к свойствам компонента в проекте только в случае необходимости, а именно, если значение данного параметра определено в базе данных. Соответственно, для него настройки синхронизации установим следующие:

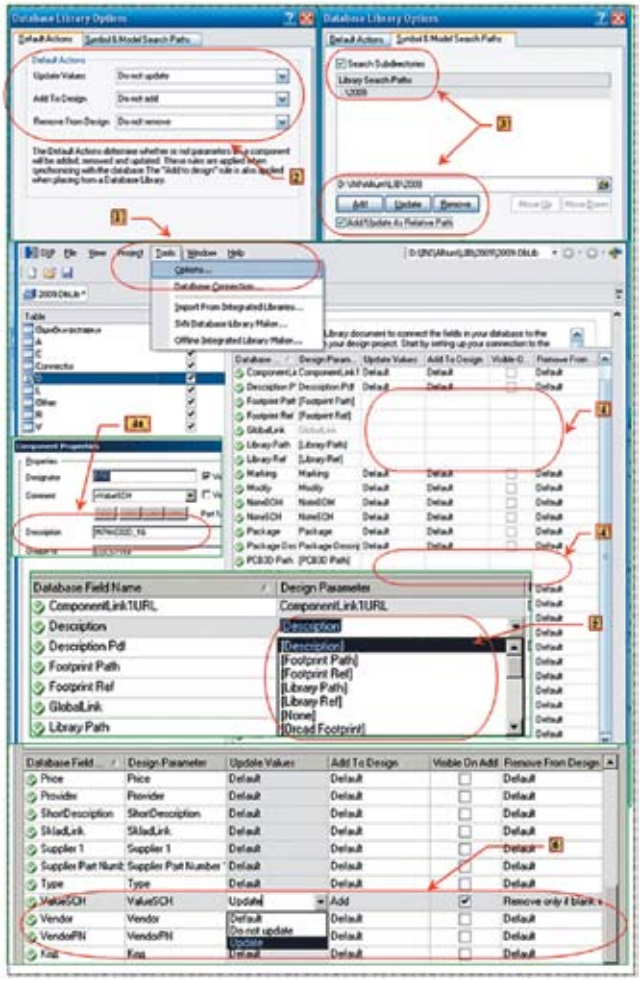

Рис. 4. Настройка свойств синхронизации значений параметров с данными в базе

- **Update Value → Update**;
- **Add To Design → Add only if not blank in database**;
- **Visible On Add → флаг снят**;
- **Add To Design → Add only if not blank in database**;
- **Remove From Design → Do Not Remove**.

в) Параметр **Marking**. Данный параметр введен автором для указания маркировочной надписи на корпусе компонента. Как правило, это актуально для маленьких корпусов, где мало места и на корпусе наносится только маркировочный цифро-буквенный код. По данному коду легко определить, тот ли компонент установлен на плате. Более того, его можно использовать при формировании различного рода сборочных чертежей. Установки свойств синхронизации сделаем такими же, как в пункте «б», и далее, если не будет оговорено особо, будем поступать так же.

г) Параметры **Vendor** и **VendorPN**. Данные параметры введены для хранения сведений о производителе комплектующего изделия и кода компонента. Эти параметры используются для однозначного определения типа компонента в схеме и будут дополнительно применяться при формировании специального параметра, соответствующего графе «Наименование» перечня элементов.

д) Параметры **ComponentLink1Description** и **ComponentLink1URL**. Данные параметры будут содержать название типа документа и гиперссылку на этот документ. Это удобная функция для быстрого открытия, например, описания на данный компонент. Отметим, можно ввести при необходимости не одну, а несколько ссылок на разные виды документов. При этом следует добавить к компоненту аналогичные параметры, изменив в их названии только цифру.

е) Параметры **Description Pdf** и **ShortDescription**. Первый используется для указания краткого описания компонента, так как это приведено в документе на комплектующее изделие. Второй — этот формализованное краткое описание, которое будет использоваться для записи компонента в графе «Наименование» перечня элементов.

Примечание. *Все указанные настройки индивидуальны для каждой таблицы базы данных, и их следует настраивать отдельно.*

Вы вправе добавлять к параметрам компонента и иные параметры из базы данных, однако в нашем примере мы ограничимся только приведенным списком.

Инсталляция библиотеки

Итак, после настройки свойств синхронизации следует проинсталлировать новую библиотеку и настроить параметры отображения в панели **Library**. Сделаем это следующим образом (рис. 5):

1. Откроем панель **System/Library**.

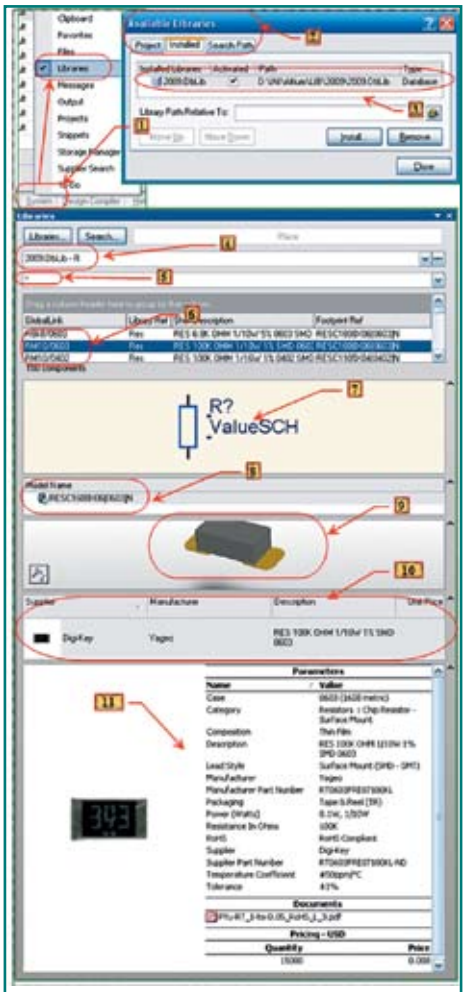

Рис. 5. Инсталляция и настройка отображения параметров компонентов в библиотеке

2. В панели **Library** нажатием кнопки **Library** вызовем панель **Available Library**, а в нем окно **Installed**.
3. В окне **Installed** нажимаем кнопку **Install** и указываем путь к файлу ссылки на базу (в нашем примере — **2009.DbLib**). В случае если библиотека не одна, следует, используя кнопки **Move Up** и **Move Doun**, определить нашей библиотеке более высокий приоритет, переместив ссылку на нее вверх по списку. Таким образом, мы обеспечим доступность нашей библиотеки для всех проектов.
4. После этого на панели **Library** будет выведена информация, которая важна при поиске новых компонентов для вставки в проект. Это имя библиотеки. На рисунке указано **2009-R**, что означает: выбрана таблица под именем **R** из базы **2009.MBD**, то есть таблица резисторов.
5. Здесь настроим фильтр и таким образом ограничим список компонентов, отображаемых в таблице. Это очень удобный инструмент, так как он работает непосредственно по набору символов. Как правило, набор из характерных двух-трех символов достаточен для получения списка, полностью отображаемого в нижнем окне. Работу с фильтром опишем ниже.
6. Здесь находится окно со списком доступных компонентов, которые выбраны из таблицы в соответствии с настройками фильтра.
7. Поле графического отображения компонента. Отметим, что если в библиотеке имеется не одно, а несколько видов графического изображения, то отображается последнее открытое. К сожалению, нет механизма выбора и просмотра текущего отображения графического изображения компонента на этапе выбора его в библиотеке. Но это можно сделать уже непосредственно после вставки в редакторе документов SCHDOC.
8. Здесь показан список подключенных с помощью базы данных к графическому изображению компонента иных моделей, включая посадочное место.
9. Внешний вид модели. В частности, в качестве модели на рис. 5 показано посадочное место.
10. Строка из таблицы поиска данного компонента по каталогу компании-поставщика. На данный момент подключены такие поставщики компонентов по каталогам, как: **www.digikey.com**, **www.farnell.com**, **www.newark.com**. Это удобно, поскольку берется текущая информация и можно видеть доступность тех или иных компонентов.
11. Более подробная информация о самом компоненте, поставщике, производителе, взятая с сайта компании-поставщика. Эта информация полезна схемотехникам при выборе компонента для применения в той или иной схеме.

Однако все же более полезна информация, указанная в п. 6, так как по ней настраивается фильтр (п. 5). И мы можем искать как по номиналу, так и по посадочному месту или по другим характерным признакам в параметрах компонента. Поэтому еще раз повторим и покажем на нашем примере, как выбрать и настроить столбцы таблицы в данном окне панели **Library** (рис. 6):

1. Наведем указатель на строку с названиями столбцов, затем, нажав правую кнопку мыши, вызовем контекстное меню и выберем в нем команду **Select Columns**.
2. В окне **Select Parameter Columns** у нас есть возможность добавить или удалить столбцы, содержимое которых будет отображаться в панели **Library**, и, соответственно, поиск компонентов можно будет проводить и по значениям этих параметров. Однако не следует выбирать все доступные параметры, так как это затруднит их отображение в таблице на панели **Library**, необходимо остановиться только на значимых столбцах. Итак, в нашем примере выбраны следующие столбцы:

а) **GlobalLink** — имя компонента в библиотеке. Приняв собственные правила присваивания имен, вам легко будет по названию судить о компоненте и определять параметры настройки фильтра ограничения списка отображаемых компонентов.

б) **LibraryRef** — имя графического отображения компонента.

в) **FootprintRef** — имя посадочного места компонента.

г) **SortDescription** — краткое формализованное описание компонента.

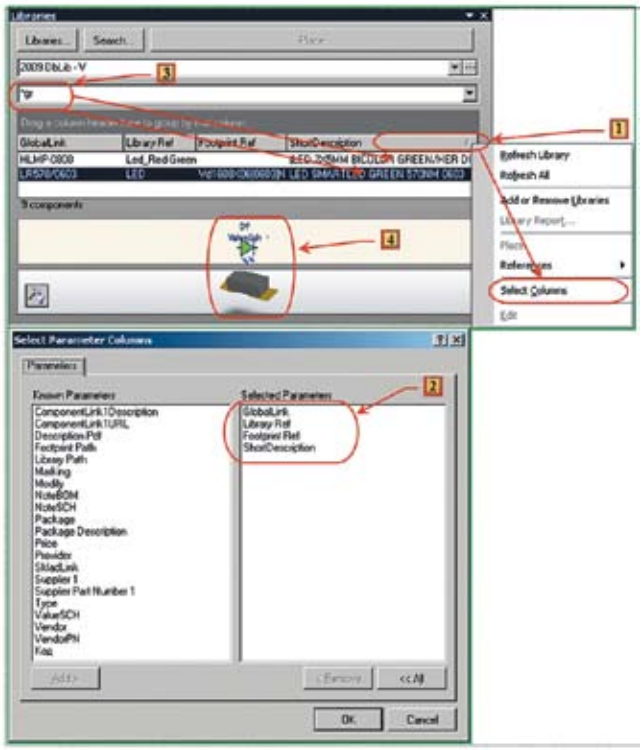

Рис. 6. Настройка отображаемых значений параметров в панели Library

Практика показала, что этих параметров вполне достаточно для быстрого поиска нужных для проекта компонентов.

3. В качестве примера на рис. 6 представлен поиск светодиодов (зеленая область спектра излучения) по первым двум буквам от слова **Green** (зеленый), в результате чего список в таблице для библиотеки нашего примера ограничился двумя позициями. В обеих строках слово **Green** встречается в параметре **SortDescription**.
4. Графическое изображение компонента и вид его посадочного места облегчают подбор комплектующего изделия.

Команды работы с базой данных

Укажем теперь те команды, которые работают с базой данных. Команды редактора графических изображений компонентов здесь не приводим, так как параметры в данной библиотеке хранить не станем. Этому редактору посвятим отдельный раздел.

Начнем с редактора схем. Опустим команды вставки новых компонентов из базы и рассмотрим только команды синхронизации уже существующих в схеме компонентов. Работу в редакторе схем автор описывал в ряде статей [3], поэтому здесь мы подробно говорить об этом не будем.

Замена компонента на новый из базы данных (рис. 7).

1. Откроем лист схемы, подведем указатель на графическое изображение выбранного компонента и двойным нажатием левой кнопки мыши вызовем окно свойств данного компонента.
2. Здесь отмечено, что данный компонент в схему был введен из библиотеки **2009.DbLib** (это и есть библиотека, описанная выше), а именно из ее таблицы с именем **C**, и наименование компонента в базе данных — **CCu10/0402/6V3/X7R**. Последнее и есть значение параметра **GlobalLink**, определенное в базе данных (этот параметр мы задали в качестве главного, и по нему будут синхронизироваться остальные параметры компонента).
3. Здесь приведены все параметры компонента, которые были добавлены из базы данных или введены дополнительно непосредственно в схеме.
4. Здесь отображаются модели и посадочные места, которые введены через базу, были определены в библиотеке графических изображений компонентов или добавлены непосредственно в схеме.
5. Заметим, что элемент в схему может быть помещен не из базы, а вообще из другой библиотеки. Чтобы проверить наличие компонента в библиотеке, на которую дана ссылка, следует нажать на кнопку **Validate**. В частности, для этого элемента указано, что он найден в библиотеке **D:/INIAltium/LIB/2009/2009.DbLib/C**.
6. Если мы хотим заменить компонент (например, требуется замена номинала или посадочного места), следует нажать кнопку **Choose**, указать новую библиотеку, таблицу в ней и там выбрать новый компонент. При замене компонента из базы полностью изменятся все значения параметров, модели, посадочные места — на новые, соответствующие новому компоненту.
7. Рекомендуем именно таким образом менять компоненты, поскольку это избавит вас от проверки соответствия всех параметров друг другу.

Рассмотрим обновление параметров компонентов из базы данных (рис. 8). Такая необходимость существует, если значения параметров были модифицированы по тем или иным причинам в самой базе данных. Итак:

1. Командой **Tool>>Update Parameter From Database** вызовем одноименное окно.
2. В левой части окна можно ограничить список листов схем проекта, для которых будет сделано обновление из базы.
3. В правой части окна можно указать конкретно, параметры каких компонентов следует обновить.
4. В случае если найдены параметры, значения которых отличны от аналогичных в базе данных, будет открыто окно **Select Parameter Change**. В этом окне параметры, которые будут обновлены, помечены синим уголком, которые добавлены — зеленым знаком

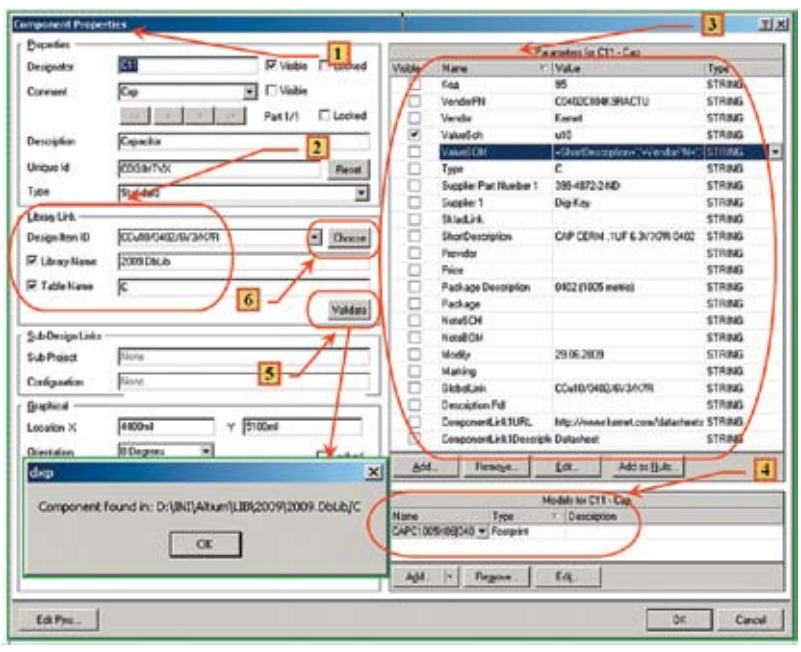

Рис. 7. Замена графического изображения на другой компонент из базы данных

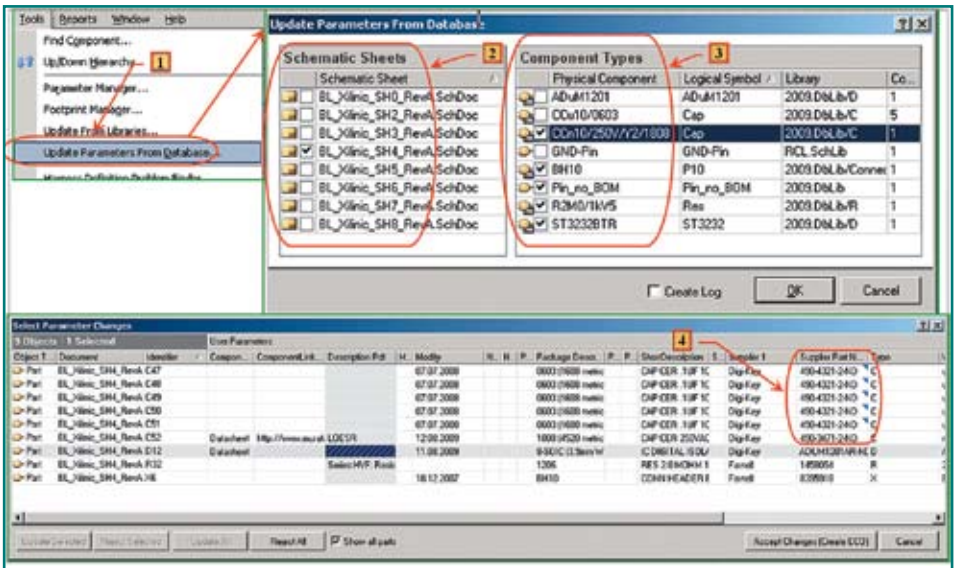

Рис. 8. Обновление параметров из базы данных

«плюс». При выделении одной из строк таблицы будет открыта схема и тот ее участок, где находится компонент, параметры которого были отображены в этой строке.

Таким образом, создание и подключение базы данных завершено. В следующей части статьи мы рассмотрим порядок формирования графических отображений компонентов для библиотеки на основе базы данных.

Продолжение следует

Литература

1. Пранович В. Altium Designer 8. Создание библиотеки на основе базы данных // Технологии в электронной промышленности. 2008. № 5.
2. AP0133 Using Components Directly from Your Company Database.pdf.
3. Пранович В. Цикл статей по Altium Designer // Технологии в электронной промышленности. 2006–2009.