Р10-2006-27

Т. Б. Петухова, А. С. Кирилов

СОЗДАНИЕ МАСТЕРА Visual Studio.Net ДЛЯ РАЗРАБОТКИ МОДУЛЕЙ УПРАВЛЕНИЯ УСТРОЙСТВАМИ В КОМПЛЕКСЕ Sonix+

Петухова Т. Б., Кирилов А. С. Р10-2006-27

Создание мастера Visual Studio.Net для разработки модулей управления устройствами в комплексе Sonix+

Рассмотрено применение технологии построения мастеров Visual Studio.Net на основе Wizard Engine для разработки модулей управления устройствами в комплексе Sonix+. Описаны этапы создания модуля и пользовательские интерфейсы мастеров «Sonix+ Device Driver» и «Sonix+ Command».

Работа выполнена в Лаборатории нейтронной физики им. И. М. Франка ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2006

Petukhova T. B., Kirilov A. S. P10-2006-27

Creation of Visual Studio.Net Wizard for Development of Modules of Management by Devices for Software Complex Sonix+

This work is devoted to application of the Wizard Engine technique of the Visual Studio.Net to driver development for software complex Sonix+. Stages of driver creation as well as the user interfaces of masters «Sonix+ Device Driver» and «Sonix+ Command» are described.

The investigation has been performed at the Frank Laboratory of Neutron Physics, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2006

Современные системы программирования в значительной степени оцениваются с точки зрения удобства использования среды разработки (*IDE — integrated development environment*). При этом существенными являются такие характеристики IDE, как оснащенность ее различными мастерами (*wizards*) и возможность расширения функциональности среды разработки.

По обоим параметрам Microsoft Visual Studio.Net может быть оценена как весьма качественно реализованная среда разработки. Встроенные в нее мастера позволяют автоматизировать или, по крайней мере, значительно упростить выполнение широкого спектра рутинных операций разработки, начиная от генерации Windows-форм и заканчивая добавлением к классам полей, методов и свойств. Весь набор готовых мастеров можно классифицировать по трем типам:

- мастер создания нового проекта (*New Project wizard*);
- мастер добавления нового элемента к проекту (*Add New Item wizard*);
- пользовательский мастер (*Custom Wizard*), который позволяет разрабатывать свой собственный мастер.

Настоящая работа посвящена применению технологии создания пользовательского мастера [1].

ПРЕДПОСЫЛКИ И НЕОБХОДИМОСТЬ СОЗДАНИЯ МАСТЕРА ДЛЯ КОМПЛЕКСА Sonix+

Комплекс Sonix+ [2] имеет модульную организацию. Часть модулей, относящихся преимущественно к пользовательскому интерфейсу, не рассматривается в данной работе, поскольку на их структуру и запуск нет ограничений. Другая часть модулей регламентирована более строго. Это принято для того, чтобы дать возможность построения структур управления с более чем одним уровнем иерархии модулей. Например, модуль *s_expo* управляет накоплением спектров для всей установки, посылая необходимые команды модулям отдельных детекторов. Как правило, эти модули предназначены для управления аппаратурой, но есть и исключения.

Модули часто могут ассоциироваться с контроллером, с помощью которого обслуживается одно или несколько устройств. Например, модуль *d_motor* обслуживает все шаговые двигатели установки, управляемые с помощью контроллера БУШД-2. Однако существуют и модули, обслуживающие чисто программные устройства, не связанные с какой-либо аппаратурой. Примерами таких модулей служат модуль интерпретатора скрипта *si* и модуль списка параметров *list_device*.

Образно говоря, можно рассматривать модуль Sonix+ как синтез формальной структуры модуля и его оригинального функционального наполне-

ния. Формализм заключается в стандартизации структуры самого модуля, а также его интерфейса и протокола межмодульного взаимодействия.

Структура модуля должна быть составлена строго по принятой схеме с использованием набора заранее подготовленных процедур, а именно:

- чтение/генерация структур файла конфигурации,
- подключение к механизму межмодульного взаимодействия, создание почтовых ящиков,
- инициирование внутренних структур данных и объявление команд и сигналов,
- запуск основного цикла.

Каждое устройство характеризуется

- набором команд,
- состоянием,
- параметрами конфигурации, которые инициируются вместе с инициированием обслуживающего модуля.

Команды описываются специальными шаблонами на C++. Их описания сводятся к заданию структур входных/выходных параметров, идентификаторов команд и процедур их реализации.

Состояние устройства хранится в специализированной базе данных Varman [3]. Оно образовано из двух частей:

- статической — для редко изменяемых параметров (параметры),
- динамической — для часто изменяемых параметров (статус).

И параметры, и статус содержат как обязательные для каждого устройства поля, так и специфические поля.

Параметры конфигурации также содержат обязательные компоненты (имя и идентификатор устройства) и произвольные компоненты, однако способ хранения всех данных в файле конфигурации стандартизирован.

Выполнение основного цикла, реакция на команды и сигналы осуществляются по строго определенному алгоритму. Уникальными должны быть только имена модулей, устройств, структур параметров конфигурации и состояния, набор команд и их реализация.

Необходимость выдерживать жесткие правила при создании модуля в комплексе Sonix+ создает настоятельную необходимость (и предпосылки) для разработки специального мастера Visual Studio.

МЕТОДИКА ПОСТРОЕНИЯ МАСТЕРА Visual Studio.Net

Типовой мастер должен содержать одно или несколько окон, обеспечивающих взаимодействие с пользователем, логику навигации (если используется несколько окон), а также основную функциональную часть — некоторые осмысленные действия, например, генерацию файлов, а также, возможно,

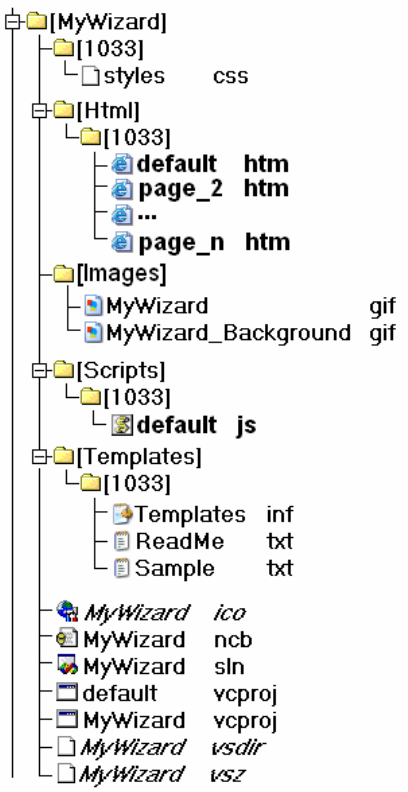
определение состояния IDE в момент вызова мастера и модификацию этого состояния по завершении работы.

С точки зрения разработчика, создание мастера для VS.Net мало чем отличается от создания обычного Windows-приложения. Тем не менее трудоемкость этого процесса оказывается достаточно высокой.

Для реализации мастеров Microsoft Visual Studio.Net содержит довольно гибкий и в то же время мощный «движок» (*Wizard Engine*). На его основе работает большинство встроенных мастеров VS.Net. Этот «движок» выполнен в виде NET-библиотеки *EnvDTE* и реализует специальную объектную модель *Visual C++ Wizard Model* [4].

Первый шаг при создании любого мастера — написание и отладка шаблонного приложения. Приложение содержит все необходимые структуры, переменные и процедуры и представляет собой работоспособную программу, не выполняющую никаких действий.

Затем с помощью мастера *Custom Wizard* выполняется генерация основы нового мастера. В папке проекта создаются необходимые каталоги и файлы:



Относительно серьезной доработке достаточно подвергнуть только выделенные файлы.

Файл *default.htm* представляет собой HTML-страницу, определяющую внешний вид пользовательского интерфейса нашего мастера и некоторую минимальную функциональность, которую можно реализовать в контексте этой страницы. Внешний вид страницы корректируется в дизайнере HTML-страниц VS.Net путем удаления ненужных и добавления необходимых элементов управления из палитры *Toolbox – HTML*. Если пользовательский интерфейс содержит одну страницу, файлы *page_2.htm – page_n.htm* отсутствуют. Имена элементов управления, размещенные на странице, формируются в таблицу символов, которые мастер использует для подстановок в исходный код приложения. Как правило, этого набора символов недостаточно, поэтому в таблицу можно добавить дополнительные символы.

Логика работы мастера обычно базируется на языке сценария Jscript [5] в файле *default.js*. По окончании работы мастера вызывается функция *OnFinish*, которая и выполняет основную работу.

Поскольку в обязанность мастера входит присвоение шаблонным именам простых понятных имен, следующим шагом в создании мастера является изменение шаблонного кода с учетом подстановок.

Кроме шаблонного кода мастер должен знать, какие файлы добавлять в создаваемый проект приложения. Такой список помещается в специальный текстовый файл *Templates.inf*.

Последним шагом в создании мастера является указание Visual Studio.NET о существовании нового мастера. Среди файлов, созданных мастером Custom Wizard, есть файл ядра мастера, в котором содержится информация о параметрах мастера (*VSZ-файл*), и файл маршрутизации между оболочкой Visual Studio.NET и элементами в проекте мастера (*VSDIR-файл*). Эти файлы необходимо разместить в определенном каталоге пакета Visual Studio.NET.

Очевидно, что методика построения мастеров на основе *Wizard Engine* не является единственно возможной. Создание прототипа мастера с помощью генератора *Visual C++ Projects – Custom Wizard* вовсе необязательно – можно вручную создать соответствующие HTML-страницы и файл обработки.

Естественно, что использование *Wizard Engine* для разработки мастеров не может обеспечить абсолютную универсальность. Тем не менее для построения относительно несложных мастеров использование *Wizard Engine* способно значительно сократить затраты на разработку.

МАСТЕР ДЛЯ РАЗРАБОТКИ МОДУЛЯ УПРАВЛЕНИЯ УСТРОЙСТВАМИ В КОМПЛЕКСЕ Sonix+

Чтобы облегчить процесс разработки драйвера устройств в комплексе Sonix+, был создан мастер, который включает в себя два типа мастеров:

• мастер создания нового проекта драйвера устройства – «*Sonix+ Device Driver»;*

• мастер добавления команды устройства – «*Sonix+ Command»*.

Таким образом, создание модуля выполняется в два этапа.

I этап — создание нового проекта драйвера устройства. Мастер создает код приложения, который включает в себя не только основной цикл работы драйвера, но и шаблонные коды всех необходимых структур данных, процедур и функций с простыми, понятными разработчику именами. Для этого необходимо выбрать «*New Project*», тип «*Sonix+ Device Driver*» (рис. 1). Откроется

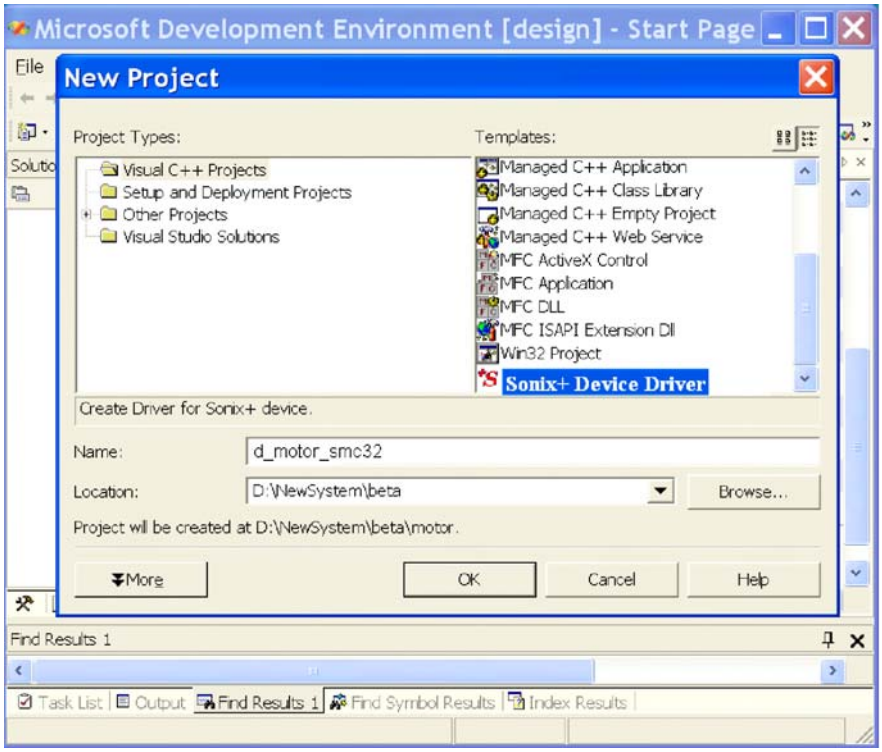


Рис. 1. Создание нового проекта драйвера устройства. Выбор типа проекта

окно пользовательского интерфейса мастера (рис. 2). Пользовательский интерфейс состоит из одной страницы, имеет текстовые поля, которые являются ключевыми символами и используются мастером при подстановке в исходные файлы. Так, например, имя класса устройства присутствует во всех именах структур данных и именах процедур, имеющих непосредственное отношение к этому устройству.

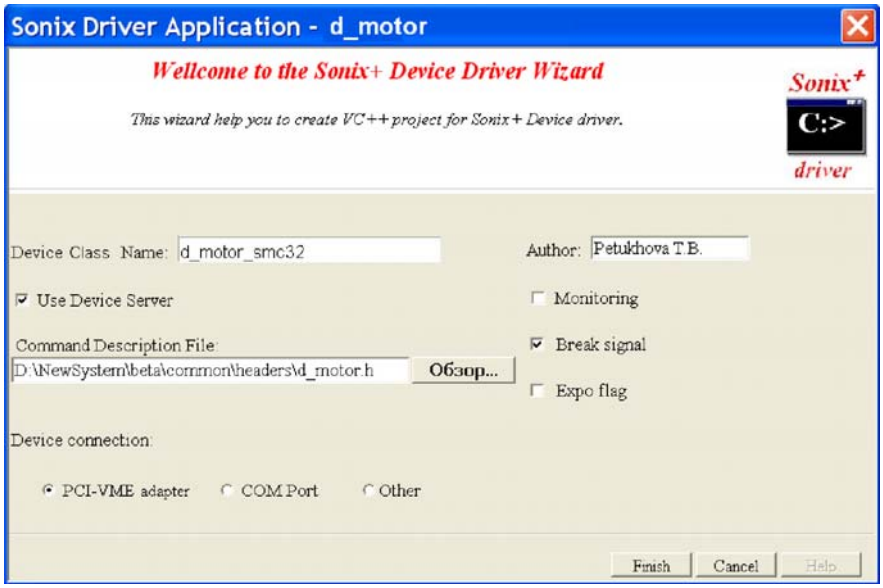


Рис. 2. Окно мастера «*Sonix+ Device Driver*»

Список файлов, добавляемых в проект приложения

Файлы	Описание
d_xxx.h	Объявления типов устройств и набора команд данного модуля.
xxxImpl.h	Описания команд модуля и структур, требуемых для конфигурации.
xxx.h	Объявления процедур, необходимых для «общения» с аппаратурой.
d_xxx.cpp	Основной цикл работы модуля. Процедуры создания и обновления конфигурационных данных и информации о данном модуле в базе данных.
xxxImpl.cpp	Процедуры, реализующие команды модуля.
xxx.cpp	Процедуры, реализующие «общение» с аппаратурой.
xxx_def.py	Описание класса устройства и команд, выполняемых устройством (для управления устройством из скрипта).
Примечание. xxx — название модуля.	

Флажки, помещенные на пользовательском интерфейсе, позволяют мастеру добавлять или отбрасывать отдельные части исходного кода. Если установлен флажок «*PCI-VME adapter*», мастер добавит в проект файл библиотеки для работы с PCI-VME-адаптером [6], флажок «*COM-port*» укажет мастеру на то, что устройство работает через последовательный порт и необходимо создать библиотечный файл для «общения» с этим устройством. Если

среди команд, выполняемых устройством, есть длинные команды, выполнение которых можно прервать, необходимо в исходный текст приложения включить описание и процедуру работы с сигналом *Break*. Если устройство управляет DAQ-экспозицией, необходимо установить «*Expo flag*».

Нажатие кнопки «*Finish*» приведет к тому, что приложение будет создано, в него добавятся «рабочие» файлы, список которых приведен в таблице.

Последний шаг в создании нового проекта — заполнение структур конфигурации и состояния модуля, написание процедур инициализации программы, создания и обновления конфигурационных данных и информации об устройстве в базе данных.

II этап — добавление команд устройства. На этом этапе возможны два варианта.

1. Набор команд определяет разработчик («самостоятельный» модуль). Разработчик, используя мастера добавления нового элемента («*Add New Item*» – «*Sonix+ Command*»), вручную описывает команды устройства. Пользовательский интерфейс мастера «*Sonix+ Command*» приведен на рис. 3. После заполнения полей формы мастера разработчику необходимо заполнить структуры входных и выходных параметров команды и написать процедуру, реализующую выполнение команды.

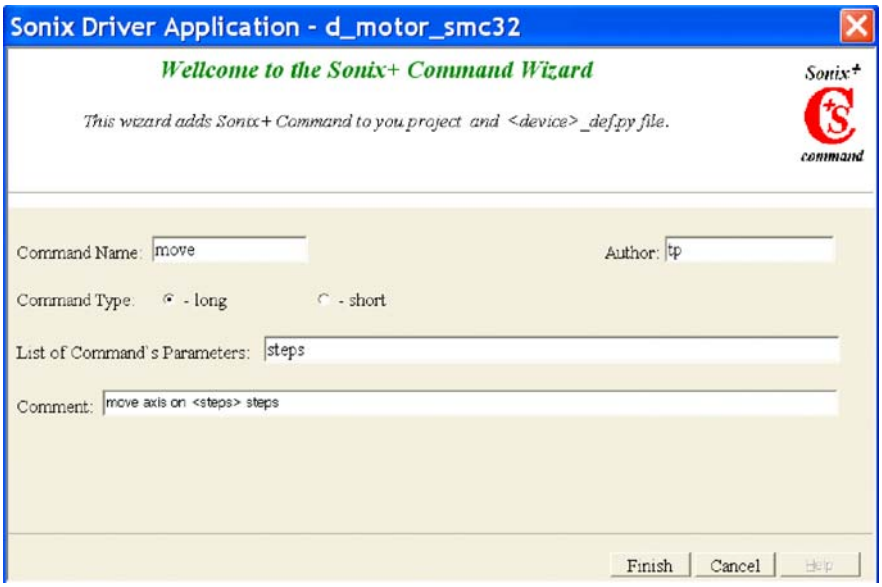


Рис. 3. Пользовательский интерфейс мастера «*Sonix+ Command*»

2. Управление устройством осуществляет сервер, набор команд фиксирован. В этом случае работает мастер «*Sonix+ Device Driver*». Он использует файл описания команд, указанный в поле «*Command Description File*» (см. рис. 2). Разработчику же остается заполнить структуры входных и выходных параметров команды и написать процедуру, реализующую выполнение команды. Имя команды присутствует в именах структур и процедуры, что облегчает поиск их в исходном тексте.

Оба мастера используют комментарий «*// здесь вставляется код ...*», чтобы указать на ту часть исходного текста, в которую надо добавить свой текст.

ЗАКЛЮЧЕНИЕ

Оба типа мастеров («*Sonix+ Device Driver*» и «*Sonix+ Command*») были разработаны и практически опробованы на ряде устройств в комплексе Sonix+. Использование мастеров действительно существенно облегчает выполнение рутинной части при разработке модулей. К сожалению, методика составления мастера в доступной для авторов литературе описана достаточно схематично, что подчас вызывало необходимость изучать предмет методом проб и ошибок.

ЛИТЕРАТУРА

1. *Шефер Дж.* Программирование на Microsoft Visual C++ .NET / Пер. с англ. М.: Торгово-издательский дом «Русская Редакция», 2005. 892 с.

2. *Kirilov A. S. et al.* Sonix+ — the new instrument control at the IBR-2 reactor // NOBUGS 2004. http://lns00.psi.ch/nobugs2004/

3. *Кирилов А. С., Юдин В. Е.* Реализация базы данных реального времени для управления экспериментом в среде MS Windows. Препринт ОИЯИ Р13-2003-11. Дубна, 2003.

4. http://msdn.microsoft.com/library/rus/

5. *Аллен В. и др.* JavaScript. Энциклопедия пользователя: Пер. с англ./Аллен Вайк. К.: Торгово-издательский дом «Dia Soft», 2001. 480 с.

6. http://www.sbs.com/computer/products/cp_pci_vme_hp.shtml

Получено 6 марта 2006 г.

Редактор *Е. В. Сабаева*

Подписано в печать 15.05.2006.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 0,50. Уч.-изд. л. 0,60. Тираж 290 экз. Заказ № 55333.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@pds.jinr.ru
www.jinr.ru/publish/