DALLAS SEMICONDUCTOR МАХІМ

Программные ресурсы сети 1-WIRE
Описание до уровня прибора

ВВЕДЕНИЕ

В настоящее время существует свыше 30-ти различных приборов для шины 1-Wire®, включая iButton®, которые выпускаются фирмой Dallas Semiconductor. Ориентация среди имеющихся API (Application Program Interface — программный интерфейс приложения), примеров программ и других ресурсов, с помощью которых обеспечивается взаимодействие с этой группой приборов, или поиск соответствующего ресурса для приборов одного типа может оказаться очень сложной задачей. В настоящем документе представлен обзор имеющихся ресурсов, а также руководство по их выбору. Все API, которые описываются в этом документе, можно использовать без каких-либо ограничений, и в большинстве случаев они включают полный исходный код.

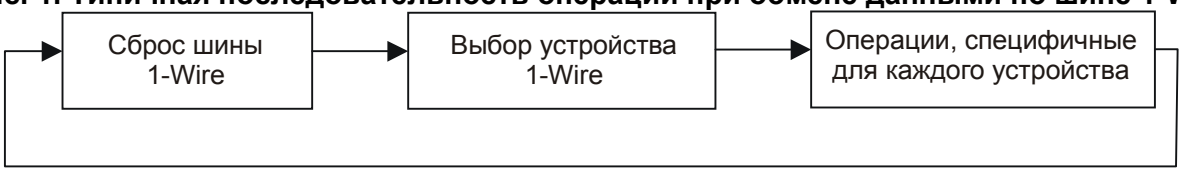
ОБЗОР ШИНЫ 1-WIRE

Шина 1-Wire, разработанная Dallas Semiconductor, представляет собой простую схему сигнализации, которая обеспечивает двустороннюю связь между одним ведущим (мастером) и периферийными приборами (ведомыми) по одному проводу. Важной характеристикой, свойственной всем приборам шины 1-Wire, является то, что каждый прибор, будь он выполнен в виде чипа или в виде iButton, имеет серийный номер, который входит в состав записанного лазером в процессе производства регистрационного номера и никогда не повторяется ни в каком другом приборе, то есть, каждый прибор является уникальным. Это позволяет индивидуально выбрать любой прибор среди многих других, которые могут быть подключены к той же самой шине. Так как к одному проводу для обмена данными могут быть одновременно подсоединены один, два или даже десятки приборов шины 1-Wire, то для того чтобы найти по очереди каждый прибор, используется алгоритм двоичного поиска. После того, как становятся известными серийные номера всех приборов, можно однозначно выбрать для обмена данными любой прибор, используя для адресации его серийный номер.

Первым этапом любого обмена данными является выдача мастером шины импульса «сброса», который синхронизирует всю шину. Затем выбирается ведомый прибор для последующего обмена. Это может осуществляться путем выбора всех ведомых приборов, выбора конкретного ведомого (используя серийный номер прибора) или путем нахождения следующего ведомого прибора с помощью алгоритма двоичного поиска. Все эти команды называют «сетевыми» (командами сетевого уровня) или командами ПЗУ (ROM). После того как определенный прибор выбран, все другие приборы отключаются и игнорируют последующий обмен данными, пока не будет передан следующий импульс сброса.

После того как прибор выбран для обмена данными по шине, мастер может передавать характерные для этого прибора команды, посылать ему или считывать из него данные. Поскольку каждый тип прибора отличается от других типов по выполняемым функциям и назначению, то, будучи выбранным, он имеет свой уникальный протокол. Но несмотря на то что протоколы и технические характеристики разных типов приборов отличаются, все они имеют один и тот же процесс выбора и следуют потоку выполнения команд, представленному на Рис. 1.

Рис. 1. Типичная последовательность операций при обмене данными по шине 1-Wire



Составной частью уникального регистрационного номера в каждом ведомом приборе является 8-битный групповой код. Этот код определен для каждого типа (модели) прибора. Поскольку типы приборов отличаются по выполняемым функциям, на основании этого кода происходит выбор протокола, который будет использоваться для управления или опроса прибора. Соответствие групповых кодов наименованиям (номерам) приборов Dallas Semiconductor см. в Табл. 1.

Таблица 1. Указатель групповых кодов

Групповой код (hex)	**Наименование прибора** () в корпусе iButton	**Описание**
01	(DS1990A), DS2401	Только сетевой адрес 1-Wire (серийный номер)
02	(DS1991), (DS1425)	Мультиключ (Multikey) iButton, защищенная память объемом 1152 бит
04	(DS1994), DS2404	4-Кбит энергонезависимое ОЗУ (RAM) с часами, таймером и сигнальными таймерами (alarms)
05	DS2405	Один адресуемый ключ
06	(DS1993)	4-Кбит энергонезависимое ОЗУ
08	(DS1992)	1-Кбит энергонезависимое ОЗУ
09	(DS1982), DS2502	1-Кбит СППЗУ (EPROM)
0A	(DS1995)	16-Кбит энергонезависимое ОЗУ
0B	(DS1985), DS2505	16-Кбит СППЗУ
0C	(DS1996), (DS1996x2), (DS1996x4)	64…256-Кбит энергонезависимое ОЗУ
0F	(DS1986), DS2506	64-Кбит СППЗУ
10	(DS1920), DS1820, DS18S20	Датчик температуры с программируемыми пороговыми значениями температуры
12	DS2406, DS2407	1-Кбит СППЗУ, двухканальный адресуемый ключ
14	(DS1971), DS2430A	256-бит ЭСППЗУ (EEPROM) и 64-бит однократно программируемый (OTP) регистр
18	(DS1963S)	4-Кбит энергонезависимое ОЗУ и процессор SHA-1
1A	(DS1963L)	4-Кбит энергонезависимое ОЗУ со счетчиками циклов записи
1D	DS2423	4-Кбит энергонезависимое ОЗУ с внешними счетчиками
1F	DS2409	2-канальный адресуемый ответвитель для подсетей
20	DS2450	4-канальный АЦП
21	(DS1921), (DS1921H), (DS1921Z)	Датчик/регистратор температуры Thermochron™
22	DS1822	Недорогой датчик температуры
23	(DS1973), DS2433	4-Кбит ЭСППЗУ
24	(DS1904), DS2415	Часы реального времени (RTC)
26	DS2438	Интеллектуальный монитор батарейного питания с датчиком температуры и АЦП
27	DS2417	Часы реального времени с прерыванием
28	DS18B20	Датчик температуры с регулируемым разрешением
2C	DS2890	Одноканальный цифровой потенциометр
30	DS2760	Высокоточный монитор Li-Ion-батареи с датчиком температуры и тока, встроенный АЦП
33	(DS1961S), DS2432	1-Кбит ЭСППЗУ с процессором SHA-1
91	(DS1981)	512-бит СППЗУ (только по специальному заказу)
96	(DS1955), (DS1957B)	Поддерживающий Java™ криптографический iButton (64-Кбайт ПЗУ, 64…134-Кбайт энергонезависимое ОЗУ)

Thermochron является торговой маркой Dallas Semiconductor.
Java является торговой маркой Sun Microsystems.

ОСНОВЫ API

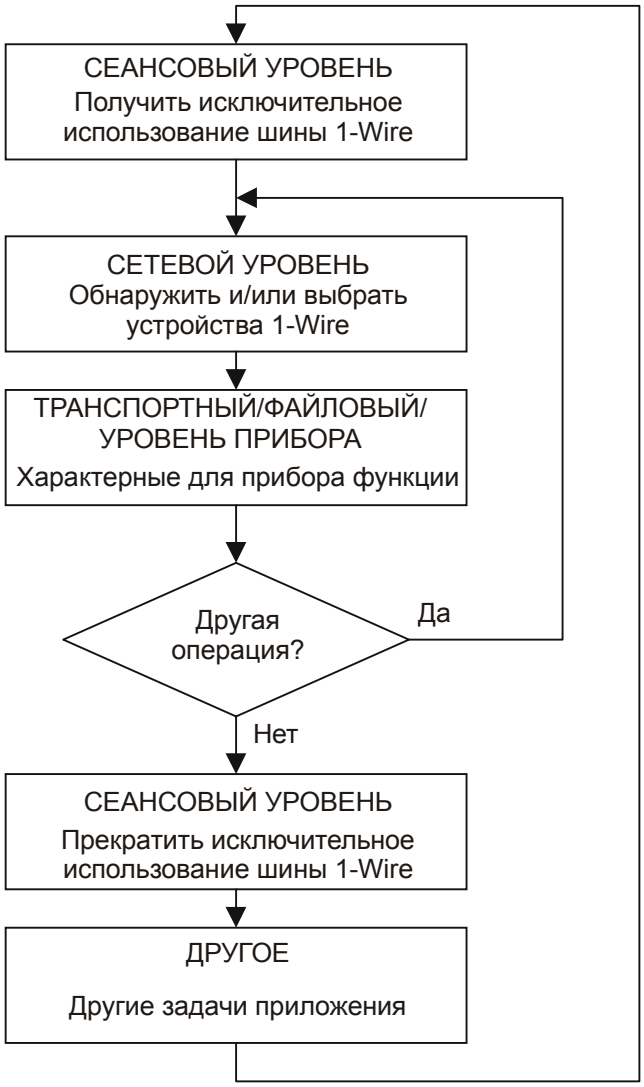
Различные API для взаимодействия с приборами 1-Wire имеют общие характеристики, которые отражают фундаментальные вопросы обмена данными, определяемые протоколом. На Рис. 2 представлено деление функций на группы (уровни), общие для различных API. Поскольку большинство приборов 1-Wire имеют память, функции обмена данными с памятью обрабатываются как общая группа API, хотя эти функции применимы не для всех приборов. Все остальные специальные, не относящиеся к памяти функции собраны в группу УРОВЕНЬ ПРИБОРА.

Рис. 2. Классификация функций API

СЕАНСОВЫЙ УРОВЕНЬ
Реализует исключительное использование шины 1-Wire. Это особенно важно в операционных системах или средах, где нескольким процессам или потокам может одновременно потребоваться доступ к одной и той же шине 1-Wire. Исключительное использование сети необходимо при выполнении ряда последовательных операций над одним прибором, когда эта последовательность не должна прерываться.
КАНАЛЬНЫЙ УРОВЕНЬ
Примитивные (элементарные) функции обмена данными по шине 1-Wire. Весь процесс обмена данными по шине 1-Wire можно свести к передаче «импульса сброса», который сбрасывает все приборы, и записи и считыванию битов. Эта группа может содержать также функции установки электрических параметров шины, например, для обеспечения специальных импульсов программирования СППЗУ (EPROM) или подачи питания.
СЕТЕВОЙ УРОВЕНЬ
Сетевые функции для поиска прибора и его выбора. Уникальный серийный номер, записанный лазером в каждый прибор 1-Wire, используется в качестве его сетевого адреса. Эти функции могут быть построены из функций канального уровня. В спецификациях на приборы 1-Wire сетевые функции называются командами ПЗУ, поскольку серийный номер записан в ПЗУ. Некоторые ведущие (мастера) шины 1-Wire содержат встроенные сетевые функции, которые более эффективны, чем функции, созданные из функций канального уровня.
ТРАНСПОРТНЫЙ УРОВЕНЬ
Функции блочного обмена данными и примитивные (элементарные) функции чтения/записи памяти. Может включать также функции чтения/записи пакетов памяти. Эти функции создаются из функций сетевого и канального уровней.
ФАЙЛОВЫЙ УРОВЕНЬ
Функции уровня файловой памяти, использующие файловую структуру 1-Wire (см. документ *Application Note 114*). Эти функции создаются из функций сетевого и транспортного уровней. Используются только для приборов с многостраничной организацией памяти.
УРОВЕНЬ ПРИБОРА
Характерные для прибора функции «высокого уровня». Эти функции часто создаются из функций сетевого, транспортного и канального уровней, и выполняют, например, такие операции, как считывание значений температуры или установка состояния ключа.

Типичная последовательность использования этих функций приведена на Рис. 3. Между функциями сеансового уровня располагается цикл обращений к прибору для обмена данными, обычно включающий использование функции сетевого уровня, за которой следует характерная для прибора функция или функция памяти.

Рис. 3. Блок-схема использования API



Характерная особенность прибора iButton заключается в том, что обмен данными происходит посредством касания прибора. Это означает, что контакт с прибором не всегда является надежным. Таблетка iButton, вставленная в считывающее устройство, может сдвигаться во время считывания. Поэтому необходимо строго следовать методике устранения ошибок. Обычно это влечет за собой повторное выполнение операции при кажущемся обнаружении ошибки и использование проверок контрольной суммы CRC при передаче данных. Функции API по обмену данными с файлом используют стандартную файловую структуру, подробно описанную в документе *Application Note 114* «1-Wire File Structure» («Файловая структура сети 1-Wire»). Эта структура использует контрольную сумму CRC16 на каждой странице данных для быстрой проверки достоверности считываемых данных. Большинство функций API 1-Wire имеют небольшое число автоматических циклов повторения или вовсе их не имеют. Эти повторы находятся под управлением приложения. Более подробную информацию о методологии устранения ошибок и оценке рисков при осуществлении обмена данными по шине 1-Wire см. в документе *Application Note 159* «Ultra Reliable 1-Wire Communication» («Сверхнадежный обмен данными в сети 1-Wire»).

ВЫБОР API

Существуют четыре разных программных интерфейса приложения (Application Program Interface — API), которые и рассматриваются в настоящем документе. Они работают на различных платформах, используют различные языки и имеют разные возможности. Эти четыре API с их кратким описанием представлены в Табл. 2.

Таблица 2. Описание API

API	Аббревиатура	Описание
Public Domain для 1-Wire	PD	Полный открытый исходный код на «Си», который поддерживает, главным образом, последовательный адаптер DS9097U на разных платформах
API 1-Wire для Java	OWAPI	Полный открытый исходный код на Java, который поддерживает почти ВСЕ приборы 1-Wire. Из «не родных» (не Java) мастеров шины 1-Wire поддерживается только последовательный адаптер DS9097U
COM для 1-Wire	OWCOM	Упаковщик модели компонентных объектов (Component Object Model — COM) Windows для OWAPI. Доступен из стандартных языков и языков подготовки сценариев типа Java Script и Visual Basic Script
API TMEX	TMEX	Поддерживает все адаптеры мастера шины 1-Wire, работающие на 32-битных платформах Windows. Обеспечивает функции канального уровня и обмена данными с файлом, но не функции уровня прибора. Драйверы являются закрытым источником. Этот API вызывается при помощи других API для получения доступа ко всем типам адаптеров шины 1-Wire

В Табл. 3 приведены операционные системы (ОС) для имеющихся API с разбивкой по языкам программирования.

Таблица 3. API для различных операционных систем и языков программирования

Язык / ОС	Независимо от языка	Си	Java
Windows 2000	TMEX / OWCOM	PD	OWAPI
Windows ME	TMEX / OWCOM	PD	OWAPI
Windows 98	TMEX / OWCOM	PD	OWAPI
Windows 95	TMEX	PD	OWAPI
Windows XP	(TMEX) / (OWCOM)	(PD)	(OWAPI)
Win3.1		PD	
DOS		PD	
PALM		PD	
VISOR		PD	
PocketPC		PD	
Solaris		(PD)	OWAPI
Linux		PD	OWAPI
TINI*		(PD) — без операционной системы TINI	OWAPI
( ) — Поддержка планируется, но пока не завершена.			

* Заметим, что TINI® является встроенной платформой с операционной системой на базе Java производства Dallas Semiconductor http://www.ibutton.com/TINI.

Поддержка отдельных групп (семейств) приборов меняется от API к API. В Табл. 4 перечислены все доступные в настоящий момент приборы 1-Wire, для каждого из них флаги A…I указывают на имеющуюся поддержку в каждом API. Расшифровка флагов приводится внизу таблицы. Заметим, что отсутствие фона означает, что прибор полностью поддерживается API. Светлый фон указывает на частичную, а темный фон — на минимальную поддержку.

TINI является зарегистрированной торговой маркой Dallas Semiconductor.

Таблица 4. Поддержка приборов в API

Прибор	Групповой код	Описание	TMEX	PD	OWAPI	OWCOM
DS1425	02	Мультиключ iButton, защищенная память объемом 1152 бит	AB	AB**I**	ABC	ABC
DS1427	04	4-Кбит ОЗУ, часы, таймер и сигнальные таймеры (alarms)	ABDE	ABCDE**I**	ABCDEFGHI	ABCD**EFGH**I
DS1820	10	Датчик температуры с программируемыми пороговыми значениями температуры	AB	ABI	ABCI	ABCI
DS18B20	28	Датчик температуры с регулируемым разрешением	AB	AB	ABI	ABI
DS18S20	10	Датчик температуры с программируемыми пороговыми значениями температуры	AB	ABI	ABCI	ABCI
DS1981	91	512-бит СППЗУ (только по специальному заказу)	ABCE	AB	AB	AB
DS1982	09	1-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**E**
DS1985	0B	16-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**E**
DS1986	0F	64-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**EFGH**
DS1904	24	Часы реального времени	AB	AB	ABI	ABI
DS1920	10	Датчик температуры с программируемыми пороговыми значениями температуры	AB	ABI	ABCI	ABCI
DS1921/H/Z	21	Датчик/регистратор температуры Thermochron	ABDE	ABCDEI	ABCDEFGHI	ABCD**EFGH**I
DS1955 DS1957	96	Поддерживающий Java криптографический iButton (64-Кбайт ПЗУ, 64…134-Кбайт энергонезависимое ОЗУ)	AB	ABI	ABI	ABI
DS1961S	33	1-Кбит ЭСППЗУ с процессором SHA-1	ABDE	ABCDEI	ABCDEFGHI	ABCD**EFGH**I
DS1963L	1A	4-Кбит энергонезависимое ОЗУ со счетчиками циклов записи	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1963S	18	4-Кбит энергонезависимое ОЗУ с процессором SHA-1	ABDE	ABCDEI	ABCDEFGHI	ABCD**EFGH**I
DS1971	14	256-бит ЭСППЗУ и 64-бит OTP-регистр	ABD	ABCDI	ABCDI	ABCDI

DS1973	23	4-Кбит ЭСППЗУ	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1990A	01	Только сетевой адрес 1-Wire	AB	AB	AB	AB
DS1991	02	Мультиключ iButton, защищенная память объемом 1152 бит	AB	AB**C**	ABC	ABC
DS1992	08	1-Кбит энергонезависимое ОЗУ	ABDE	ABCDE	ABCDEFGH	**ABCD**EFGH
DS1993	06	4-Кбит энергонезависимое ОЗУ	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1994	04	4-Кбит энергонезависимое ОЗУ, а также часы, таймер, сигнальные таймеры	ABDE	ABCDE**I**	ABCDEFGHI	ABCD**EFGH**I
DS1995	0A	16-Кбит энергонезависимое ОЗУ	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1996	0C	64-Кбит энергонезависимое ОЗУ	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1996x2	0C	64-Кбит × 2 (128 Кбит) энергонезависимое ОЗУ	ABDE	ABCDE	ABCDEFGH	ABCD**EFGH**
DS1996x4		64-Кбит × 4 (256 Кбит) энергонезависимое ОЗУ				
DS2401	01	Только сетевой адрес 1-Wire	AB	AB	AB	AB
DS2405	05	Один ключ	AB	ABI	ABI	ABI
DS2404	04	4-Кбит энергонезависимое ОЗУ, а также часы, таймер и сигнальные таймеры	ABDE	ABCDE**I**	ABCDEFGHI	ABCD**EFGH**I
DS2406	12	1-Кбит СППЗУ, 2-канальный адресуемый ключ	ABCE	ABCDEI	ABCDEI	ABCD**E**I
DS2407						
DS2409	1F	Сдвоенный ключ, ответвитель	AB	ABI	ABI	ABI
DS2415	24	Часы реального времени	AB	AB	ABI	ABI
DS2417	27	Часы реального времени с прерыванием	AB	AB	ABI	ABI
DS2423	1D	4-Кбит энергонезависимое ОЗУ с внешними счетчиками	ABDE	ABCDEI	ABCDEFGHI	ABCD**EFGH**I
DS2430A	14	256-бит СППЗУ и 64-бит OTP-регистр	ABD	ABCDI	ABCDI	ABCDI
DS2432	33	1-Кбит ЭСППЗУ с процессором SHA-1	ABDE	ABCDEI	ABCDEFGHI	ABCD**EFGH**I
DS2450	20	Четыре АЦП	AB	ABI	ABI	ABI

DS2438	26	Интеллектуальный монитор батарейного питания с датчиком температуры и АЦП	AB	ABI	ABCI	**ABCI**
DS2502	09	1-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**E**
DS2505	0B	16-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**E**
DS2506	0F	64-Кбит СППЗУ	ABCE	ABCDE	ABCDE	ABCD**EFGH**
DS2760	30	Высокоточный монитор Li-Ion-батареи с датчиком температуры и тока, встроенный АЦП	AB	AB	ABI	ABI
DS2890	2C	Одноканальный цифровой потенциометр	AB	AB	ABI	ABI

Расшифровка фона поддержки

Полная поддержка
Частичная поддержка
Минимальная поддержка

Флаги поддержки

A. Поддержка примитивов (элементарных функций) 1-Wire на канальном уровне
B. Поддержка 1-Wire на сетевом уровне
C. Поддержка чтения/записи байта памяти на транспортном уровне
D. Поддержка чтения/записи пакета памяти на транспортном уровне
E. Поддержка файловой структуры 1-Wire типа AA (типы файловых структур см. в документе *Application Note 114*)
F. Поддержка файловой структуры 1-Wire типа AB
G. Поддержка файловой структуры 1-Wire типа BA
H. Поддержка файловой структуры 1-Wire типа BB
I. Поддержка других, характерных для прибора функций

Флаги, выделенные жирным шрифтом

Указывают, что текущая, предварительная версия этого API пока еще не обеспечивает данный вид поддержки, но в окончательной версии будет его поддерживать.

ОБЗОР PUBLIC DOMAIN (PD) ДЛЯ 1-WIRE

Функции, которые реализованы в API PD для 1-Wire, написаны полностью на языке «Си» и предназначены для использования на платформах, не поддерживаемых API TMEX. «Сеть 1-Wire» (или MicroLan) — это сеть, имеющая один провод данных (сигнальный) и общий провод (земля), с одним ведущим (мастером) и одним или более ведомыми приборами. Рассматриваемый API создает мастера шины 1-Wire, который может использоваться для идентификации и обмена данными с ведомыми приборами. Он полностью обеспечивает работу на сетевом, транспортном и файловом уровне для обмена данными со всеми приборами 1-Wire производства Dallas Semiconductor, включая приборы iButton.

Найти этот комплект API и примеры построения платформы можно на сайте:
http://www.ibutton.com/software/1wire/wirekit.html.

Исходный код на языке «Си» для данного API является переносимым. Чтобы обеспечить его выполнение для определенной платформы, предусматриваются шаблоны TODO. На сайте представлено несколько примеров, реализованных на следующих платформах: 32-бит Windows, Linux, Visor, Palm и PocketPC. Приведено также несколько примеров приложений, в которых используются реализации API на этих платформах.

Существует два набора переносимых исходных файлов. Первый — это набор общего назначения general, который предназначен для платформ, уже имеющих примитивные (элементарные) функции канального уровня для обмена данными по шине 1-Wire. Это самый низкий уровень, который зависит от аппаратных средств. Другой набор переносимых исходных файлов userial предназначается для пользователя, который имеет последовательный порт (RS232) и желает использовать «Мастер драйвера универсальной последовательной шины 1-Wire DS2480B». Эта микросхема принимает команды через последовательный порт, выполняет операции 1-Wire и затем посылает результаты обратно на последовательный порт. Исходная программа преобразует заданные операции 1-Wire в последовательные пакеты для обмена данными с DS2480B. В этом случае для платформы необходим только программный модуль, реализующий элементарные функции (примитивы) чтения/записи последовательного порта. Прибор DS2480B представляет собой микросхему интерфейса, которая используется во всех последовательных адаптерах серии DS9097U.

Эти два набора переносимых файлов исходных кодов реализуют одни и те же функции API 1-Wire и являются взаимозаменяемыми. На Рис. 4 представлен API для версии 3.00 базового кода PD 1-Wire. Заметим, что характерные для прибора функции, не связанные с памятью, подробно не приводятся из-за их большого количества. На Рис. 5 представлены файлы исходных кодов, которые обеспечивают функции и требуемые модули для новых платформ.

В дополнение к переносимым модулям на «Си» в загружаемом комплекте PD также имеется ограниченное число примеров микропроцессорных устройств для осуществления обмена данными по шине 1-Wire.

Функции файлового уровня в рассматриваемом API реализуют файловую структуру 1-Wire типа AA, как это определено в документе *Application Note 114*, который можно найти на сайте: http://pdfserv.maxim-ic.com/arpdf/AppNotes/app114.pdf.

Как и подразумевает название этого API, лицензия на представленный исходный код программ делает их практически общедоступными. Разработчики могут свободно использовать этот код в своих приложениях без каких-либо ограничений.

Рис. 4. Функции API для PD

СЕАНСОВЫЙ УРОВЕНЬ
owAcquire — Запрашивает исключительное использование сети 1-Wire. *owRelease* — Освобождает сеть 1-Wire от исключительного использования.
КАНАЛЬНЫЙ УРОВЕНЬ
owHasOverDrive — Указывает, имеет ли адаптер возможности ускорения. *owHasPowerDelivery* — Указывает, может ли адаптер обеспечить подачу питания. *owHasProgramPulse* — Указывает, доступно ли напряжение программирования СППЗУ. *owLevel* — Устанавливает уровень (режим) линии 1-Wire: Обычный (5 В, слабая подтяжка), Подача питания (5 В, мощная подтяжка) или Программирование (12 В, уровень программирования СППЗУ). *owProgramPulse* — Посылает синхронизированный программирующий импульс для записи СППЗУ прибора 1-Wire. *owReadBitPower* — Считывает 1 бит, а затем может обеспечить подачу питания. *owReadByte* — Принимает 8 бит из сети 1-Wire, посылая все «единицы» (0xFF). *owSpeed* — Устанавливает скорость в сети 1-Wire, соответствующую обычному (16 Кбит в секунду) или ускоренному режиму (142 Кбит в секунду). *owTouchBit* — Посылает и принимает 1 бит из сети 1-Wire. *owTouchByte* — Посылает и принимает 8 бит из сети 1-Wire. *owTouchReset* — Сбрасывает все приборы, подключенные к сети 1-Wire, и возвращает результат операции. *owWriteByte* — Посылает 8 бит в сеть 1-Wire и проверяет совпадение принятого ответного сигнала. *owWriteBytePower* — Посылает 8 бит данных в сеть 1-Wire, а затем подает питание.
СЕТЕВОЙ УРОВЕНЬ
owAccess — Выбирает текущий прибор и готовит его к характерной для данного прибора команде. *owFamilySearchSetup* — Устанавливает параметры следующего поиска (owNext), для того чтобы найти прибор определенной группы (семейства). *owFirst* — Ведет поиск «первого» прибора 1-Wire в сети 1-Wire. *owNext* — Ведет поиск «следующего» прибора 1-Wire в сети 1-Wire. *owOverdriveAccess* — Выбирает текущий прибор и переводит его в ускоренный режим работы. *owSerialNum* — Извлекает или задает серийный номер выбранного прибора (номер ПЗУ). *owSkipFamily* — Пропускает все приборы 1-Wire одной группы (семейства), которые были найдены в процессе последнего поиска. *owVerify* — Выбирает и проверяет, присутствует ли текущий прибор (и, как опция, произошло ли у него срабатывание сигнального таймера — alarming).
ТРАНСПОРТНЫЙ УРОВЕНЬ
owBlock — Посылает в сеть 1-Wire и принимает из сети блок данных, возможно, со сбросом. *owCanLockPage* — Проверяет, имеет ли заданный банк памяти страницы, которые могут быть заблокированы. *owCanLockRedirectPage* — Проверяет, имеет ли заданный банк памяти страницы, которые могут быть заблокированы от переадресации. *owGetAlternateName* — Получает альтернативные наименования (номера) приборов. *owGetBankDescription* — Получает строку описания банка памяти. *owGetDescription* — Получает краткое описание типа прибора 1-Wire. *owGetExtraInfoDesc* — Получает описание содержимого дополнительной информации. *owGetExtraInfoLength* — Получает длину (в байтах) дополнительной информации, содержащейся в этом банке памяти. *owGetMaxPacketDataLength* — Получает максимальную длину данных (в байтах) для пакета. *owGetName* — Получает наименование (номер) прибора 1-Wire в виде строки. *owGetNumberBanks* — Получает число банков памяти для определенной группы (семейства) 1-Wire. *owGetNumberPages* — Получает количество страниц в заданном банке памяти. *owGetPageLength* — Получает длину исходной (необработанной) страницы в байтах для заданного банка памяти. *owGetSize* — Получает размер заданного банка памяти в байтах**.** *owGetStartingAddress* — Получает физический начальный адрес заданного банка памяти. *owHasExtraInfo* — Проверяет, предоставляют ли страницы этого банка памяти дополнительную

информацию при считывании.
owHasPageAutoCRC — Проверяет, имеет ли банк памяти проверку сгенерированной прибором контрольной суммы CRC при считывании страницы.
owIsGeneralPurposeMemory — Проверяет, является ли банк памяти пользовательской памятью общего назначения.
owIsNonvolatile — Проверяет, является ли текущий банк памяти энергонезависимым.
owIsReadOnly — Проверяет, является ли этот банк памяти доступным только для чтения.
owIsReadWrite — Проверяет, имеет ли этот банк памяти доступ для чтения/записи.
owIsWriteOnce — Проверяет, является ли этот банк памятью с однократной записью, например, как СППЗУ.
owNeedsPowerDelivery — Проверяет, требуется ли для записи в этот банк памяти подача питания.
owNeedsProgramPulse — Проверяет, требуется ли для записи в этот банк памяти импульс программирования.
owRead — Считывает часть банка памяти в режиме «исходных данных» (пакетов нет, CRC).
owReadPage — Считывает всю страницу банка памяти в режиме «исходных данных» (пакетов нет, CRC).
owReadPageCRC — Считывает всю страницу банка памяти с проверкой сгенерированной прибором контрольной суммы CRC.
owReadPageExtra — Считывает всю исходную (необработанную) страницу банка памяти, включая любую «дополнительную» информацию (пакетов нет, CRC).
owReadPageExtraCRC — Считывает всю исходную (необработанную) страницу банка памяти, включая любую «дополнительную» информацию, с проверкой сгенерированной прибором контрольной суммы CRC.
owReadPagePacket — Считывает универсальный пакет данных (Universal Packet Data) из страницы в банке памяти (описание структуры универсального пакета данных см. в *Application Note 114*).
owReadPagePacketExtra — Считывает универсальный пакет данных из страницы в банке памяти с «дополнительной» информацией.
owRedirectPage — Проверяет, имеет ли банк памяти страницы, которые могут быть переадресованы.
owWrite — Записывает часть банка памяти в режиме «исходных данных».
owWritePagePacket — Записывает универсальный пакет данных на страницу в банке памяти.

ФАЙЛОВЫЙ УРОВЕНЬ

owAttribute — Изменяет атрибуты файла.
owChangeDirectory — Изменяет текущую директорию.
owCloseFile — Закрывает файл.
owCreateDir — Создает директорию.
owCreateFile — Создает файл для записи.
owCreateProgramJob — Создает буфер записи для регистрации незавершенных заданий программирования СППЗУ.
owDeleteFile — Удаляет файл.
owDoProgramJob — Записывает незавершенные задания программирования СППЗУ.
owFirstFile — Находит первый файл в текущей директории.
owFormat — Форматирует файловую систему файловой структуры 1-Wire.
owGetCurrentDir — Получает текущую директорию.
owNextFile — Находит следующий файл в текущей директории.
owOpenFile — Открывает файл для чтения.
owReadFile — Читает открытый файл.
owReadFile — Считывает данные из файла.
owRemoveDir — Удаляет директорию.
owReNameFile — Изменяет имя файла.
owWriteFile — Записывает в файл, который был создан.

УРОВЕНЬ ПРИБОРА

DoAtoDConversion — Выполняет аналого-цифровое преобразование в приборе DS2450.
ReadSwitch12 — Считывает состояние ключа DS2406.
GetFirmwareVersionString — Получает строку версии встроенной программы для прибора iButton, поддерживающего Java. слишком много, чтобы перечислять все характерные для приборов функции.

В Примере 1, приведенном ниже, показан фрагмент кода PD, который соответствует блок-схеме использования API, изображенной на Рис. 3. Для простоты предполагается, что обнаружение каждого прибора в сети 1-Wire происходит при каждом прохождении через рабочий цикл. Более сложное приложение, например, могло бы осуществлять поиск приборов только одного типа или, возможно, выбирать прибор, найденный в предыдущем поиске.

Пример 1. ПРИМЕР КОДА PD

```
int rslt, portnum=0, doing_work=1;
char portString[50]; // установить соответствующую платформе строку порта

// рабочий цикл
while (doing_work)
{
// запросить исключительное использование сети 1-Wire (СЕАНСОВЫЙ УРОВЕНЬ)
if (owAcquire(portnum, portString))
{
// найти все приборы (СЕТЕВОЙ УРОВЕНЬ)
rslt = owFirst(portnum, TRUE, FALSE);
while (rslt)
{
// выполнить КАКУЮ-ТО ОПЕРАЦИЮ с найденным прибором (УРОВНИ
// ТРАНСПОРТНЫЙ/ФАЙЛОВЫЙ/ПРИБОРА)
// . . .

// найти следующий прибор (СЕТЕВОЙ УРОВЕНЬ)
rslt = owNext(portnum, TRUE, FALSE);
}
// освободить сеть 1-Wire от исключительного использования (СЕАНСОВЫЙ
// УРОВЕНЬ)
owRelease(portnum);
}
else
{
// не смог получить исключительное использование сети 1-Wire
// . . .
}

// выполнить другую задачу приложения
// . . .
}
```

На Рис. 5 (а и б) перечисляются модули на языке «Си», которые составляют каждый из двух наборов библиотек PD для 1-Wire. Отображены также функции TODO, которые необходимы для переноса библиотеки на новую платформу. Заметим, что в комплекте предоставляется несколько примеров файлов связи для платформы, реализующих функции TODO.

Рис. 5a. Реализация USERIAL для PD

СЕАНСОВЫЙ УРОВЕНЬ					
owsesu.c					
КАНАЛЬНЫЙ УРОВЕНЬ					
owllu.c	ds2480ut.c	ds2480.h			
СЕТЕВОЙ УРОВЕНЬ					
ownetu.c	crcutil.c				
ТРАНСПОРТНЫЙ УРОВЕНЬ					
mbappreg.c	mbappreg.h	mbee.c	mbee.h	mbeprom.c	mbeprom.h
mbnv.c	mbnv.h	mbnvcrc.c	mbnvcrc.h	mbscr.c	mbscr.h
mbscrcrc.c	mbscrcrc.h	mbscree.c	mbscree.h	mbscrex.c	mbscrex.h
mbsha.c	mbsha.h	mbshaee.c	mbshaee.h	owtrnu.c	rawmem.c
rawmem.h					
ФАЙЛОВЫЙ УРОВЕНЬ					
owcache.c	owfile.c	owfile.h	owpgrw.c	owprgm.c	
УРОВЕНЬ ПРИБОРА					
ad26.c	ad26.h	atod20.c	cnt1d.c	jib96.c	jib96o.c
jib96.h	sha18.c	sha33.c	shadebit.c	shadbtvm.c	shaib.c
shaib.h	swt05.c	swt12.c	swt12.h	swt1f.c	temp10.c
thermo21.c	thermo21.h	weather.c	weather.h		
УТИЛИТА MISC					
ioutil.c	owerr.c	findtype.c	ownet.h	screenio.c	sprintf.c

TODO

Предоставляется интерфейсный модуль SERIAL (последовательный), который реализует следующие функции:

*BreakCOM** — Посылает команду «РАЗОРВАТЬ» (BREAK) на последовательный порт в течение не менее 2 мс.

CloseCOM — Закрывает ранее открытый последовательный порт (опция для некоторых платформ).

*FlushCOM** — Позволяет завершить любую незавершенную операцию записи и очистить входной буфер.

*msDelay** — Задерживает не менее чем на заданное число миллисекунд.

msGettick — Возвращает значение суммирующего счетчика миллисекунд (опция для некоторых примеров).

OpenCOM — Открывает заданный последовательный порт для обмена данными (опция для некоторых платформ).

*ReadCOM** — Считывает определенное число байтов из последовательного порта.

SetCOMBaud — Изменяет скорость последовательной передачи данных на заданную (опция, если необходимо ускорение).

*WriteCOM** — Записывает определенное число байтов в последовательный порт.

* Минимальный набор функций, требуемых для работы.

Рис. 5б. Реализация GENERAL для PD

СЕАНСОВЫЙ УРОВЕНЬ
(см. TODO)
КАНАЛЬНЫЙ УРОВЕНЬ
(см. TODO)
СЕТЕВОЙ УРОВЕНЬ
ownet.c crcutil.c
ТРАНСПОРТНЫЙ УРОВЕНЬ
Те же, что и в реализации USERIAL, за исключением owtrnu.c, который заменяется на owtran.c
ФАЙЛОВЫЙ УРОВЕНЬ
Те же, что и в реализации USERIAL
УРОВЕНЬ ПРИБОРА
Те же, что и в реализации USERIAL
УТИЛИТА MISC
Те же, что и в реализации USERIAL
TODO
Предоставляется интерфейсный модуль канального и сеансового уровней, который реализует следующие функции: *owAcquire* — Запрашивает исключительное использование сети 1-Wire. *owRelease* — Освобождает сеть 1-Wire от исключительного использования. *owHasOverDrive* — Указывает, имеет ли адаптер возможности ускорения. *owHasPowerDelivery* — Указывает, может ли адаптер подавать питание. *owHasProgramPulse* — Указывает, доступно ли напряжение программирования СППЗУ. *owLevel* — Устанавливает уровень (режим) линии 1-Wire: Обычный (5 В, слабая подтяжка), Подача питания (5 В, мощная подтяжка) или Программирование (12 В, уровень программирования СППЗУ). *owProgramPulse* — Посылает синхронизированный программирующий импульс для записи СППЗУ прибора 1-Wire. *owReadBitPower* — Считывает 1 бит, а затем может обеспечить подачу питания. *owReadByte* — Принимает 8 бит из сети 1-Wire, посылая все «единицы» (0xFF). *owSpeed* — Устанавливает скорость в сети 1-Wire, соответствующую обычному (16 Кбит в секунду) или ускоренному режиму (142 Кбит в секунду). *owTouchBit** — Посылает и принимает 1 бит из сети 1-Wire. *owTouchByte* — Посылает и принимает 8 бит из сети 1-Wire. *owTouchReset** — Сбрасывает все приборы в сети 1-Wire и возвращает результат операции. *owWriteByte* — Посылает 8 бит в сеть 1-Wire и проверяет совпадение принятого ответного сигнала. *owWriteBytePower* — Посылает 8 бит данных в сеть 1-Wire, а затем подает питание. * Минимальный набор функций, требуемых для работы.

УСТАНОВКА

API для Public Domain (PD) 1-Wire представляет собой набор модулей на языке «Си», поэтому какая-либо формальная процедура установки отсутствует. Как показано в примерах «построения», требуемые модули компилируются прямо в приложения. Однако это не мешает разработчику объединять модули в загружаемую библиотеку, такую как Windows DLL.

ОБЗОР API 1-WIRE ДЛЯ JAVA (OWAPI)

API 1-Wire для Java с самого начала был разработан как очень устойчивая, объектно-ориентированная база для построения приложений 1-Wire в Java. Он расширяет возможности программистов по разработке переносимого, межплатформенного программного обеспечения и сокращает время выхода на рынок для разработанных ими интегрированных продуктов 1-Wire.

API состоит из многих классов Java и интерфейсов. Специальную группу классов Java в API 1-Wire является контейнер (класс OneWireContainer). Посредством контейнеров обеспечивается поддержка конкретных приборов 1-Wire, включая iButton. API имеет более 30-ти различных типов контейнеров, представляющих большинство приборов 1-Wire. Каждый контейнер заключает в себе (инкапсулирует) и реализует функциональные возможности отдельного прибора.

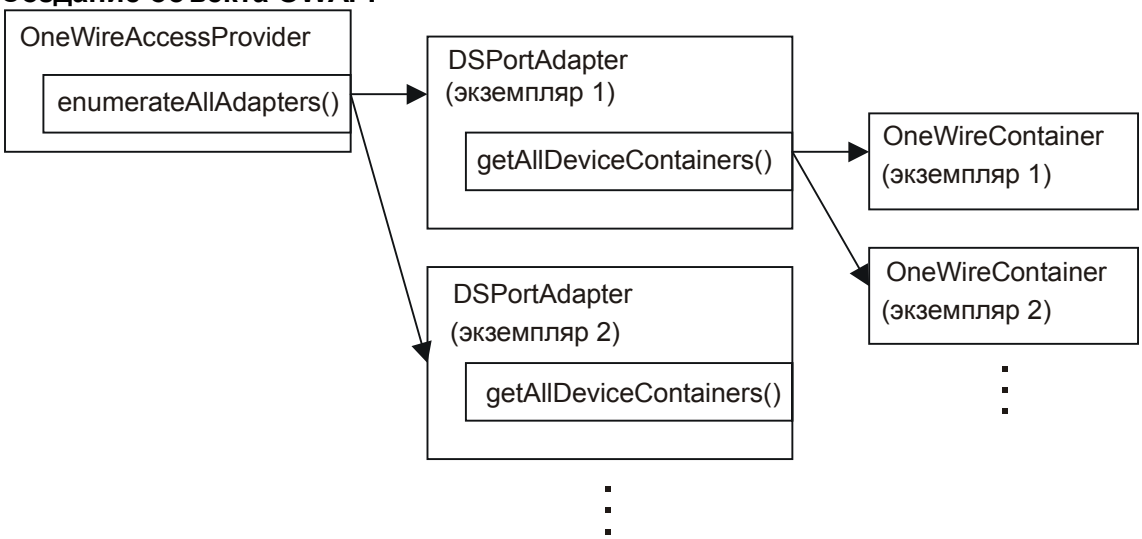
«Контейнер» взаимодействует с прибором 1-Wire через класс адаптеров 1-Wire, который представляет собой физический адаптер 1-Wire (класс DSPortAdapter). Экземпляр адаптера создается из класса провайдера (класс OneWireAccessProvider). Фактическая реализация адаптеров 1-Wire меняется от платформы к платформе, но все они имеют один и тот же интерфейс. Некоторые платформы используют собственные драйверы, но большинство поддерживают, по меньшей мере, последовательные адаптеры DS9097U-XXX с помощью API обмена данными для Java (Java™ Communications API) или эквивалентного. Этот API имеется на сайте компании Sun: http://java.sun.com/products/javacomm.

API 1-Wire для комплекта Java находится на сайте:
http://www.ibutton.com/software/1wire/1wire_api.html.

Как и в случае комплекта PD для 1-Wire, полные исходные коды на Java для OWAPI предоставляются согласно общедоступной лицензии.

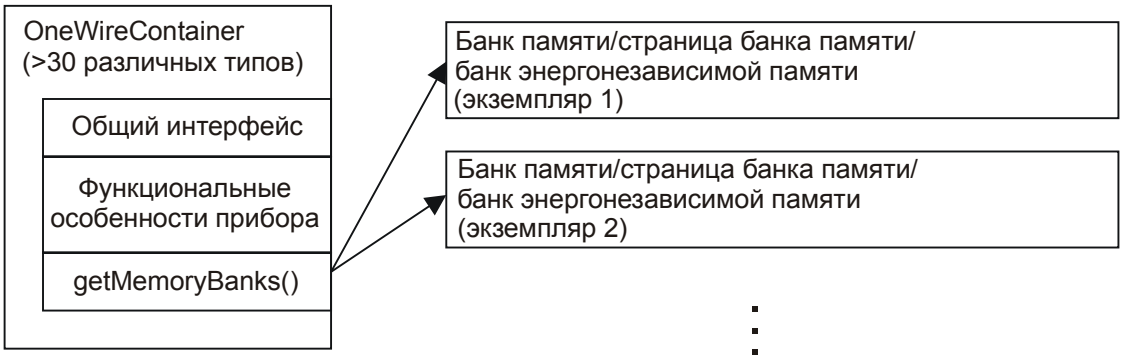
На Рис. 6 показана типичная последовательность создания объекта для этого API. «Провайдер» создает экземпляр «адаптера» (один или несколько), который, в свою очередь, может создавать экземпляры «контейнеров» прибора. Далее обмен данными с прибором осуществляется в основном посредством «контейнера».

Рис. 6. Создание объекта OWAPI



На Рис. 7 показаны общие характеристики «контейнера». Прибор, который содержит память, создает экземпляр «банка памяти» для каждого банка памяти. Память разделяется на банки в зависимости от набора функциональных возможностей банка. Например, один банк может быть энергонезависимым, в то время как другой не энергонезависимым. Или же банк может быть памятью общего назначения либо «с распределением памяти» для изменения функциональных возможностей прибора.

Рис. 7. Характеристики OneWireContainer OWAPI



На Рис. 8 приведены методы, которые предусматриваются базовыми классами OWAPI. Класс или пакет отображается **жирным шрифтом**. Заметим, что, поскольку каждый контейнер имеет высокоуровневые методы (функции) для управления каждым типом прибора, в адаптере функции канального уровня напрямую обычно не вызываются.

Рис. 8. Функции OWAPI

СЕАНСОВЫЙ УРОВЕНЬ
com.dalsemi.onewire.adapter.DSPortAdapter *beginExclusive* — Запрашивает исключительное использование сети 1-Wire. *endExclusive* — Освобождает сеть 1-Wire от исключительного использования.
КАНАЛЬНЫЙ УРОВЕНЬ
com.dalsemi.onewire.adapter.DSPortAdapter *canBreak* — Проверяет, поддерживается ли адаптером операция «разрыва» («break») 1-Wire (длительный НИЗКИЙ уровень). *canDeliverPower* — Проверяет, поддерживается ли адаптером подача питания через схему «мощной подтяжки». *canDeliverSmartPower* — Проверяет, поддерживает ли адаптер «интеллектуальную» подачу питания. «Интеллектуальная» подача питания — это способность определять, когда потребление мощности снизилось, и автоматически прекращать подачу питания. *canFlex* — Проверяет, поддерживается ли адаптером гибкое регулирование временных параметров при обмене данными по длинной линии. *canHyperdrive* — Проверяет, поддерживает ли адаптер сверхскоростной режим обмена данными. *canOverdrive* — Проверяет, поддерживает ли адаптер ускоренный режим обмена данными. *canProgram* — Проверяет, поддерживает ли адаптер напряжение программирования СППЗУ 12 В. *dataBlock* — Посылает в сеть 1-Wire и принимает из сети блок данных. *getBit* — Считывает один бит из сети 1-Wire. *getBlock* — Считывает блок из сети 1-Wire (посылая все «единицы» (0xFF)). *getByte* — Считывает байт из сети 1-Wire, посылая все «единицы» (0xFF). *getSpeed* — Считывает текущую скорость обмена данными в сети 1-Wire. *putBit* — Записывает бит в сеть 1-Wire. *putByte* — Записывает байт в сеть 1-Wire и проверяет корректность ответного сигнала. *reset* — Сбрасывает все приборы сети 1-Wire. *setPowerDuration* — Устанавливает продолжительность подачи питания. *setPowerNormal* — Отключает подачу питания. *setProgramPulseDuration* — Устанавливает длительность программирующего импульса. *setSpeed* — Устанавливает скорость обмена данными в сети 1-Wire. *startBreak* — Начинает интервал разрыва (НИЗКИЙ уровень) в сети 1-Wire. *startPowerDelivery* — Начинает подачу питания. *startProgramPu*lse — Запускает программирующий импульс.

СЕТЕВОЙ УРОВЕНЬ

com.dalsemi.onewire.adapter.DSPortAdapter

excludeFamily — Исключает из поиска определенную группу (семейство).
findFirstDevice — Находит первый прибор в сети 1-Wire без автоматического создания контейнера.
findNextDevice — Находит следующий прибор в сети 1-Wire без автоматического создания контейнера.
getAllDeviceContainers — Ведет поиск и находит все приборы в сети 1-Wire с контейнерами.
getDeviceContainer — Получает контейнер прибора для «текущего» найденного прибора.
getFirstDeviceContainer — Находит первый прибор и создает для него контейнер.
getNextDeviceContainer — Находит следующий прибор и создает для него контейнер.
setNoResetSearch — Запрещает передачу импульса сброса во время поиска в сети 1-Wire.
setSearchAllDevices — Устанавливает поиск всех приборов в сети 1-Wire (за исключением приборов со сработавшими сигнальными таймерами).
SetSearchOnlyAlarmingDevices — Устанавливает поиск в сети 1-Wire только тех приборов, у которых произошло срабатывание сигнального таймера.
targetAllFamilies — Устанавливает поиск всех приборов в сети 1-Wire (снимает исключения).
targetFamily — Указывает конкретную группу (семейство) при поиске в сети 1-Wire.
(также в **com.dalsemi.onewire.container.***)
isAlarming — Проверяет, не произошло ли срабатывание сигнального таймера прибора.
isPresent — Проверяет, присутствует ли прибор в сети 1-Wire.
select — Выбирает прибор в сети 1-Wire для его подготовки к команде, характерной для данного прибора.

ТРАНСПОРТНЫЙ УРОВЕНЬ

com.dalsemi.onewire.container.MemoryBank

getBankDescription — Возвращает текстовое описание банка памяти.
getSize — Получает размер банка памяти в байтах.
getStartPhysicalAddress — Получает начальный физический адрес банка памяти.
isGeneralPurposeMemory — Проверяет, является ли банк памяти банком общего назначения (не с распределением памяти).
isNonVolatile — Проверяет, является ли банк памяти энергонезависимым.
isReadOnly — Проверяет, является ли банк памяти доступным только для чтения.
isReadWrite — Проверяет, имеет ли банк памяти доступ для чтения и записи.
isWriteOnce — Проверяет, является ли банк памятью с однократной записью, например, как СППЗУ.
needsPowerDelivery — Проверяет, требуется ли для записи в этот банк памяти подача питания.
needsProgramPulse — Проверяет, требуется ли для записи в этот банк памяти программирующий импульс.
read — Считывает банк памяти без обработки (пакетная структура отсутствует).
setWriteVerification — Устанавливает режим, при котором данный API выполняет дополнительную проверку после операций записи.
write — Записывает в банк памяти данные в исходном виде (пакетная структура отсутствует).

com.dalsemi.onewire.container.PagedMemoryBank

getExtraInfoDescription — Получает описание дополнительной информации, связанной с этим банком.
getExtraInfoLength — Получает длину (в байтах) дополнительной информации, содержащейся в каждой странице.
getMaxPacketDataLength — Получает максимальную длину данных, которые могут содержаться в «пакетной» структуре, способной разместиться в каждой странице этого банка памяти.
getNumberPages — Получает число страниц в этом банке памяти.
getPageLength — Получает длину (в байтах) исходной (необработанной) страницы в этом банке памяти.
hasExtraInfo — Проверяет, имеет ли этот банк памяти дополнительную информацию, связанную с каждой страницей.

hasPageAutoCRC — Проверяет, имеют ли страницы в этом банке памяти проверку CRC, которая обеспечивается прибором.
readPage — Считывает страницу из банка памяти.
readPageCRC — Считывает страницу из банка памяти, используя сгенерированную прибором контрольную сумму CRC.
readPagePacket — Считывает структуру пакета из страницы в банке памяти.
writePagePacket — Записывает структуру пакета на страницу в банке памяти.
com.dalsemi.onewire.container.OTPMemoryBank
canLockPage — Проверяет, могут ли страницы в банке памяти быть заблокированы от последующих операций записи.
canLockRedirectPage — Проверяет, может ли быть заблокирована возможность переадресации в банке памяти, чтобы не допустить последующую переадресацию.
canRedirectPage — Проверяет, допускает ли банк памяти переадресацию страниц в качестве способа обновления страниц с однократной записью.
getRedirectedPage — Получает номер страницы, на которую происходит переадресация страницы.
isPageLocked — Проверяет, заблокирована ли страница от последующих операций записи.
isRedirectPageLocked — Проверяет, заблокирована ли страница от последующей переадресации.
lockPage — Блокирует страницу.
lockRedirectPage — Блокирует страницу от переадресации.
redirectPage — Переадресует страницу на новую страницу. Используется для обновления приборов с однократной записью.

ФАЙЛОВЫЙ УРОВЕНЬ

com.dalsemi.onewire.utils.OWFile
Те же методы, что и в java.io.File (для версии JDK 1.2) плюс следующие дополнительные методы:
close — Закрывает файл и освобождает все связанные с ним ресурсы.
format — Форматирует файловую систему 1-Wire, связанную с этим прибором (приборами), с помощью предназначенного для этого OWFile.
getFD — Получает OWFileDescriptor для этого файла, чтобы файл мог быть «синхронизирован» с прибором.
getFreeMemory — Получает имеющуюся свободную память в файловой системе 1-Wire.
getLocalPage — Получает ссылку на местную страницу банка памяти из страницы файловой системы 1-Wire.
getMemoryBankForPage — Получает экземпляр банка памяти, который может быть использован для чтения/записи указанной страницы файловой системы 1-Wire.
getOneWireContainer — Получает контейнер (контейнеры), который составляет файловую систему.
getPageList — Получает список страниц файловой системы 1-Wire, которые содержат данный файл.
com.dalsemi.onewire.utils.OWFileDescriptor
Те же методы, что и в java.io.FileDescriptor (для версии JDK 1.2).
com.dalsemi.onewire.utils.OWFileOutputStream
Те же методы, что и в java.io.FileOutputStream (для версии JDK 1.2).
com.dalsemi.onewire.utils.OWFileInputStream
Те же методы, что и в java.io.FileInputStream (для версии JDK 1.2).

УРОВЕНЬ ПРИБОРА

com.dalsemi.onewire.container.*
Более 30-ти различных, характерных для прибора, реализаций контейнеров, включая шесть интерфейсов к различным типам «датчиков»:
ADContainer — АЦП.
ClockContainer — Часы.
SwitchContainer — Ключ.
TemperatureContainer — Датчик температуры.
PotentiometerContainer — Цифровой потенциометр.
HumidityContainer — Датчик влажности.
com.dalsemi.onewire.application.*
Классы утилит SHA и 1-Wire Tagging (тегирование 1-Wire).
com.dalsemi.onewire.jib.*
Обеспечивает обслуживание OpenCard для iButton, поддерживающих Java (информация, касающаяся Opencard, находится на сайте: http:\\www.opencard.org).

В Примере 2, приведенном ниже, показан фрагмент кода OWAPI, который соответствует блок-схеме использования API, изображенной на Рис. 3. Для простоты предполагается, что обнаружение каждого прибора в сети 1-Wire происходит при каждом прохождении через рабочий цикл. Более сложное приложение, например, могло бы осуществлять поиск приборов только одного типа или, возможно, выбирать прибор, найденный в предыдущем поиске.

Пример 2. Пример кода OWAPI

```
boolean doing_work = true;

// получить от провайдера (поставщика услуг) адаптер по умолчанию
DSPortAdapter adapter = OneWireAccessProvider.getDefaultAdapter();

// рабочий цикл
while (doing_work)
{
// получить исключительное использование адаптера (СЕАНСОВЫЙ УРОВЕНЬ)
adapter.beginExclusive(true);

// сбросить любые предыдущие установки ограничения поиска (СЕТЕВОЙ УРОВЕНЬ)
adapter.setSearchAllDevices();
adapter.targetAllFamilies();
adapter.setSpeed(adapter.SPEED_REGULAR);

// составить перечень всех найденных приборов 1-Wire (СЕТЕВОЙ УРОВЕНЬ)
for (Enumeration owd_enum = adapter.getAllDeviceContainers();
owd_enum.hasMoreElements(); )
{
// получить «контейнер» для каждого прибора
OneWireContainer owd = ( OneWireContainer ) owd_enum.nextElement();

// выполнить КАКУЮ-ТО ОПЕРАЦИЮ с найденным прибором (УРОВНИ
// ТРАНСПОРТНЫЙ/ФАЙЛОВЫЙ/ПРИБОРА)
// . . .
}

// завершить исключительное использование адаптера (СЕАНСОВЫЙ УРОВЕНЬ)
adapter.endExclusive();

// выполнить другую задачу приложения
// . . .
}
```

ТЕГИРОВАНИЕ 1-Wire

С увеличением числа датчиков 1-Wire и ростом их разнообразия сложность управления сетью 1-Wire значительно возрастает. Например, датчик типа АЦП может измерять множество различных значений, поэтому необходимо иметь возможность «тегировать» (помечать) датчик для описания его функции. В API 1-Wire для Java была создана и реализована схема тегирования 1-Wire (1-Wire Tagging), использующая XML. Эти теги позволяют приложению динамически загружать и конфигурировать датчик, чтобы придать ему контекст. Подробнее см. в документе *Application Note 158* «1-Wire Tagging with XML» («Тегирование 1-Wire с помощью XML»).

УСТАНОВКА

Все требуемые модули, которые исполняют вызовы функций API, представленные на Рис. 8, содержатся в одном jar-файле OneWireAPI.jar. Правильное размещение одного такого модуля или указание пути (classpath) обеспечивает доступ ко всему API. Однако существуют два исключения: для конкретной платформы может потребоваться установка собственных API или API для обмена данными или же платформа может иметь ограничения по размеру, поэтому нежелательно наличие всего API полностью. Оба эти исключения подробно рассматриваются в комплекте OWAPI.

ОБЗОР COM ДЛЯ 1-Wire (OWCOM)

Основой COM (модели компонентных объектов) Windows сети 1-Wire является просто упаковщик для OWAPI (API 1-Wire для Java). Практически любой язык программирования, который работает в 32-битных операционных системах Microsoft Windows, может использовать объекты COM. Поскольку в данном случае используется OWAPI, разработчикам предоставляется богатое собрание классов поддержки приборов 1-Wire производства Dallas Semiconductor для различных сред программирования, включая C++, VisualBasic, JavaScript, JScript, Perl и VBScript.

Поскольку Java и COM не полностью совместимы, чтобы получить существующий базовый код на Java для работы в COM-интерфейсе, необходимо применить несколько искусственных приемов. Эти приемы описываются в следующем разделе.

Комплект разработчика программного обеспечения (Software Developer's Kit), содержащий OWCOM (и TMEX), можно найти на сайте: http://www.ibutton.com/software/tmex/index.html.

Исходный код на Java для создания OWCOM предоставляется согласно свободной лицензии типа PD. Исходными кодами не обеспечиваются только драйверы API TMEX, вызываемые для поддержки различных адаптеров 1-Wire (см. раздел API TMEX). Эти драйверы, впрочем, могут распространяться без каких-либо ограничений.

РАЗЛИЧИЯ МЕЖДУ OWCOM И OWAPI

Поскольку структура, лежащая в основе этих интерфейсов, фактически реализована с помощью OWAPI, оба они имеют одну и ту же документацию. Существует всего несколько правил использования документации Java для COM-интерфейса.

1) В Java используется 64-битный тип long (целое двойной точности), а интерфейсы, которые применяются в COM, не поддерживают числа больше 32-х бит. Для любой функции в Java, оперирующей в качестве параметра или возвращаемого значения с переменной типа long, используется тип string (строка). В языках программирования с неявным определением типов (например, VisualBasic или JavaScript) преобразование между типами long и string будет автоматическим и, вероятно, не окажет какого-либо влияния на код. Для всех остальных языков следует уделить особое внимание тому, чтобы параметры преобразовывались в string и возвращали значения, преобразованные в 64-битный тип long.

2) Некоторые из классов в OWAPI используют массив байтов для кэширования информации состояния, касающейся объекта. К сожалению, языки, которые не предусматривают передачу значений по ссылке (подобно JavaScript), в действительности клонировали данные в массиве, предназначенном для передачи их в качестве параметра. Поскольку при этом нарушается правильная передача значений массива в качестве параметров, был создан объект для содержания массивов байтов. Этот объект — ByteArray — является единственным объектом, эквивалента которого не существует в имеющемся на сегодня API 1-Wire для Java. Необходимая для него документация (описание) приводится ниже:

```
ByteArray
{
// присваивает элементу массива с индексом nIndex значение nValue
void setAt(nIndex,nValue);

// возвращает числовое значение элемента массива с индексом nIndex
nValue getAt(nIndex);

// увеличивает или сокращает размер массива, копирует все данные, которые
// ему соответствуют
void setSize(nSize);

// возвращает размер массива
nSize getSize();
}
```

3) Для того чтобы различать методы как по типам параметров, так и по числу параметров, в Java предусматривается перезагрузка метода. В COM два метода с одним и тем же именем можно отличить только по числу параметров. Все методы, которые не отвечали этому требованию, соответствовали одному и тому же шаблону. Они ожидали или тип string, или тип long, или массив байтов. Метод, ожидающий тип string, был оставлен без изменений, в то время как метод, ожидающий тип long, имеет теперь новое имя, в конце которого добавлено «FromLong». Метод, ожидающий массив байтов, имеет в конце своего имени добавление «FromBytes». Например:

Java		**COM**
isPresent(String)	-->	isPresent(String)
isPresent(long)	-->	isPresentFromLong(String) //Согласно правилам 1 и 3
isPresent(byte[])	-->	isPresentFromBytes(ByteArray) //Согласно правилам 2 и 3

4) Работа некоторых контейнеров в OWAPI зависит от полиморфизма. В сущности, полиморфизм — это возможность обращаться с классом, как будто в действительности это экземпляр родительского (порождающего) класса. В COM полиморфизм проявляет себя не совсем так, поскольку COM не поддерживает наследование. Каждый компонент COM в нашей архитектуре представляет реальный экземпляр объекта из OWAPI, но обычно компонент высокого уровня (подобный MemoryBank) фактически содержит экземпляр объекта низкого уровня (например, PagedMemoryBank или OTPMemoryBank). В Java объект просто приводится к установленному для него типу. К сожалению, в COM это не так просто, поэтому к трем контейнерам верхнего уровня (OneWireContainer, MemoryBank и DSPortAdapter) был добавлен вспомогательный метод. Интерфейс для вспомогательного метода:

```
Component getMostSpecificComponent();
```

Возвращаемое значение зависит от того, какой класс вызывается. Например, экземпляр OneWireContainer, который фактически содержит объект Thermochron (OneWireContainer21), при вызове getMostSpecificComponent возвратит экземпляр COM-интерфейса OneWireContainer21.

5) Еще одной возможностью, утерянной из-за полиморфизма, является механизм, с помощью которого обнаруживается, поддерживает или нет конкретный прибор 1-Wire конкретный интерфейс. К тому же, полиморфизм позволяет обращаться с классом таким образом, будто он в действительности является экземпляром интерфейса, реализуемого этим классом. Например, OneWireContainer21 реализует интерфейс TemperatureContainer, то есть OneWireContainer21 реализует все методы, которые являются общими для интерфейсов TemperatureContainer (такие, как «readTemperature»). В Java общим является сравнение экземпляра объекта 1-Wire с конкретным интерфейсом при помощи ключевого слова «instanceof». Если выражение «myOneWireDevice instanceof TemperatureContainer» истинно, то прибор реализует все методы, необходимые для считывания значений температуры. Решение для COM состоит в использовании этих методов, определенных во всех OneWireContainer:

```
boolean isTemperatureContainer();
boolean isADContainer();
boolean isSwitchContainer();
boolean isClockContainer();
boolean isPotentiometerContainer();
boolean isHumidityContainer(); [ПОКА ЕЩЕ НЕ РЕАЛИЗОВАН]
```

6) Функции в Java, которые используются для возвращения объекта java.util.Calendar, возвращают теперь текущее время в миллисекундах, начиная с 1 января 1970 года среднего времени по Гринвичу (GMT). Это стандарт, принятый в Unix для представления времени.

7) В настоящее время все интерфейсы постоянно находятся на одном и том же уровне, а не в соответствующих пакетах Java. Например, OneWireContainer21 больше не находится в пакете com.dalsemi.onewire.container.OneWireContainer21, он доступен из пакета COM просто как OneWireContainer21.

В Примере 3, приведенном ниже, показан фрагмент кода OWCOM, который соответствует блок-схеме использования API, изображенной на Рис. 3. Для простоты предполагается, что обнаружение каждого прибора в сети 1-Wire происходит при каждом прохождении через рабочий цикл. Более сложное приложение, например, могло бы осуществлять поиск приборов одного типа или, возможно, выбирать прибор, найденный в предыдущем поиске.

Пример 3. Пример кода Java Script в OWCOM

```
boolean doing_work=true;

// получить провайдера доступа к 1-Wire
var access = WScript.CreateObject("owapi.OneWireAccessProvider");

// получить адаптер по умолчанию
var adapter = access.getDefaultAdapter();

// рабочий цикл
while (doing_work)
{
// получить исключительное использование адаптера (СЕАНСОВЫЙ УРОВЕНЬ)
adapter.beginExclusive(true);

// сбросить любые предыдущие установки ограничения поиска (СЕТЕВОЙ УРОВЕНЬ)
adapter.setSearchAllDevices();
adapter.targetAllFamilies();
adapter.setSpeed(adapter.SPEED_REGULAR);

// составить перечень всех найденных приборов 1-Wire (СЕТЕВОЙ УРОВЕНЬ)
for (Enumeration owd_enum = adapter.getAllDeviceContainers();
owd_enum.hasMoreElements(); )
{
// получить «контейнер» для каждого прибора
owd = owd_enum.nextElement();

// выполнить КАКУЮ-ТО ОПЕРАЦИЮ с найденным прибором (УРОВНИ
// ТРАНСПОРТНЫЙ/ФАЙЛОВЫЙ/ПРИБОРА)
// . . .
}

// завершить исключительное использование адаптера (СЕАНСОВЫЙ УРОВЕНЬ)
adapter.endExclusive();

// выполнить другую задачу приложения
// . . .
}
```

УСТАНОВКА

Существует командный файл, который автоматически устанавливает драйвер OWCOM. Он регистрирует объект COM с помощью программы regsvr32, входящей в Windows, а затем копирует файлы класса OWAPI в «доверяемый» каталог.

ОБЗОР API TMEX (TMEX)

API TMEX является набором независимых от языка 32-битных библиотек (DLL) Windows, который обеспечивает выполнение основных функций для всех приборов 1-Wire, включая ограниченную поддержку файловой структуры 1-Wire для приборов памяти. Этот API предназначен для работы в многозадачных многопоточных приложениях, которые поочередно используют одни и те же или различные порты 1-Wire. API может поддерживать до 16-ти различных типов адаптеров 1-Wire, каждый с 16-ю отдельными портами. Он поддерживает все адаптеры 1-Wire, созданные фирмой Dallas Semiconductor. Поэтому этот API используется в качестве «собственных» драйверов для API 1-Wire для Java на платформах Win32. Поскольку API COM для 1-Wire базируется на API 1-Wire для Java, API TMEX используется и в API COM. На Рис. 9 графически показано, как другие API пользуются преимуществом поддержки Win32, которую обеспечивает API TMEX. На этом рисунке приведены имена файлов реальных драйверов и показана иерархия этих драйверов.

Комплект разработчика программного обеспечения (Software Developer's Kit — SDK), содержащий API TMEX (и OWCOM), находится на сайте: http://www.ibutton.com/software/tmex/index.html.

Все примеры, представленные в SDK TMEX, даются с исходными кодами, однако исходные коды для драйверов в настоящее время не обеспечиваются. Впрочем, драйверы могут распространяться без каких-либо ограничений.

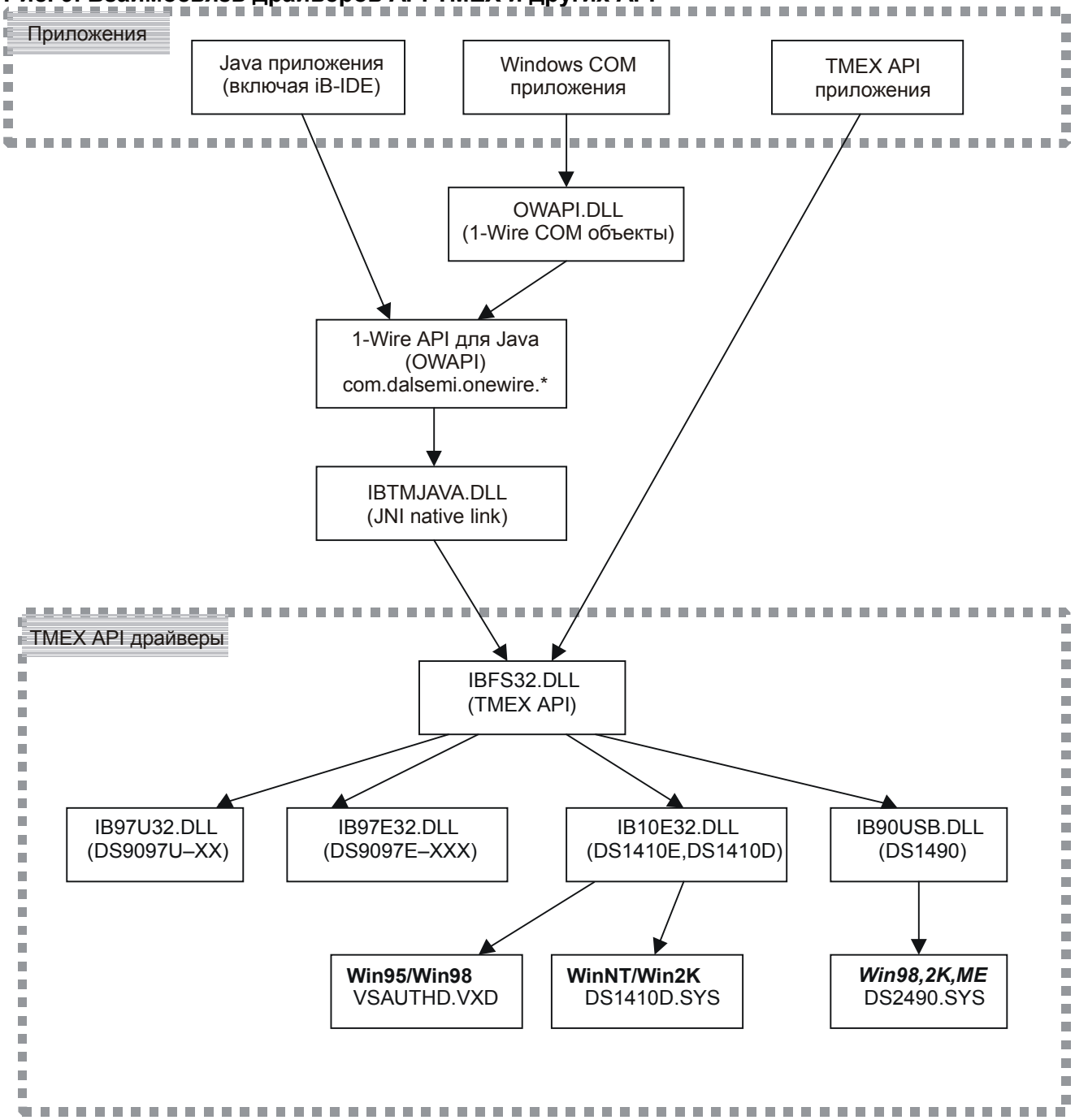
В Табл. 5, приведенной ниже, перечислены поддерживаемые в настоящее время адаптеры 1-Wire, а также особенности каждого адаптера и поддерживаемые им платформы Windows. Заметим, что поддержка Windows XP пока находится в стадии разработки.

Таблица 5. Адаптеры, поддерживаемые TMEX

Адаптер	Порт	Особенности	Win95	Win98	WinME	WinNT	Win2K	WinXP*
DS1490F Держатель 2-в-1	USB	Подача питания Ускоренный режим Светодиодный индикатор Держатель одного iButton		X	X		X	X
DS1490x (в стадии разработки)	USB	Подача питания Ускоренный режим RJ-11 или держатель iButton Как опция, запись СППЗУ		X	X		X	X
DS1410E	Параллельный	Подача питания Ускоренный режим Держатель на два iButton Идентификатор DS2401	X	X	X	X	X	X
DS1410D	Параллельный	Держатель на два iButton Идентификатор DS2401	X	X	X	X	X	X
DS9097U -009	Последовательный	Подача питания Ускоренный режим Разъем RJ-11 Идентификатор DS2502	X	X	X	X	X	X
DS9097U -S09	Последовательный	Подача питания Ускоренный режим Разъем RJ-11	X	X	X	X	X	X
DS9097U -E25	Последовательный	Подача питания Ускоренный режим Разъем RJ-11 Запись СППЗУ	X	X	X	X	X	X
DS1411	Последовательный	Подача питания Ускоренный режим Держатель одного iButton	X	X	X	X	X	X
DS9097E	Последовательный	Разъем RJ-11 Запись СППЗУ	X	X	X	X	X	X
DS9097	Последовательный	Разъем RJ-11	X	X	X	X	X	X
DS1413	Последовательный	Держатель одного iButton	X	X	X	X	X	X

* Поддержка Windows XP пока находится в стадии разработки.

Рис. 9. Взаимосвязь драйверов API TMEX и других API



На Рис. 10 перечислены функции, предоставляемые API TMEX. Заметим, что этот API не обеспечивает каких-либо характерных для приборов функций, не связанных с памятью.

Рис. 10. Функции API TMEX

СЕАНСОВЫЙ УРОВЕНЬ
TMEndSession — Освобождает сеть 1-Wire. *TMExtendedStartSession* — Запрашивает исключительное использование сети 1-Wire. *TMValidSession* — Проверяет, является ли текущий сеанс в сети 1-Wire действительным.
КАНАЛЬНЫЙ УРОВЕНЬ
TMClose — Освобождает ресурсы для открытого порта (не всегда применимо). *TMOneWireCom* — Устанавливает скорость сети 1-Wire, соответствующую обычному (16 Кбит в секунду) или ускоренному режиму (142 Кбит в секунду). *TMOneWireLevel* — Устанавливает уровень (режим) линии 1-Wire: Обычный (5 В, слабая подтяжка), Подача питания (5 В, мощная подтяжка) или Программирование (12 В, уровень программирования СППЗУ). *TMProgramPulse* — Посылает синхронизированный программирующий импульс в сеть 1-Wire для программирования СППЗУ. *TMSetup* — Проверяет функционируют ли порт и адаптер. *TMTouchBit* — Посылает и принимает 1 бит из сети 1-Wire. *TMTouchByte* — Посылает и принимает 1 байт из сети 1-Wire. *TMTouchReset* — Сбрасывает все приборы в сети 1-Wire и возвращает результат операции.
СЕТЕВОЙ УРОВЕНЬ
TMAccess — Выбирает текущий прибор и готовит его к характерной для прибора команде. *TMAutoOverDrive* — Устанавливает драйвер для автоматического выбора режима обмена данными с прибором: ускоренного или обычного. *TMFamilySearchSetup* — Устанавливает текущее состояние поиска для нахождения определенной группы (семейства) при следующем поиске (TMNext или TMNextAlarm). *TMFirst* — Ведет поиск «первого» прибора 1-Wire в сети 1-Wire. *TMFirstAlarm* — Ведет поиск «первого» прибора 1-Wire, у которого произошло срабатывание сигнального таймера, в сети 1-Wire. *TMNext* — Ведет поиск «следующего» прибора 1-Wire в сети 1-Wire. *TMNextAlarm* — Ведет поиск «следующего» прибора 1-Wire, у которого произошло срабатывание сигнального таймера, в сети 1-Wire. *TMOverAccess* — Выбирает текущий прибор и переводит его в ускоренный режим. *TMRom* — Извлекает или задает серийный номер выбранного прибора (номер ПЗУ). *TMSkipFamily* — Пропускает все приборы одной группы (семейства), которые были найдены в процессе последнего поиска. *TMStrongAccess* — Выбирает и проверяет, присутствует ли текущий прибор. *TMStrongAlarmAccess* — Выбирает и проверяет, присутствует ли текущий прибор И произошло ли у него срабатывание сигнального таймера.
ТРАНСПОРТНЫЙ УРОВЕНЬ
TMBlockIO — Посылает в сеть 1-Wire и принимает из сети блок данных, которому предшествовал сброс 1-Wire. *TMBlockStream* — Посылает в сеть 1-Wire и принимает из сети блок данных БЕЗ сброса 1-Wire. *TMExtendedReadPage* — Считывает всю страницу банка памяти с проверкой сгенерированной прибором контрольной суммы CRC (применимо не для всех типов приборов). *TMProgramByte* — Программирует байт в СППЗУ прибора 1-Wire. *TMReadPacket* — Считывает универсальный пакет данных из страницы (описание структуры универсального пакета данных см. в документе *Application Note 114*). *TMWritePacket* — Записывает универсальный пакет данных на страницу.
ФАЙЛОВЫЙ УРОВЕНЬ
TMAttribute — Изменяет файл или атрибуты директории. *TMChangeDirectory* — Считывает или изменяет текущую рабочую директорию. *TMCloseFile* — Закрывает открытый или созданный файл, чтобы высвободить дескриптор файла. *TMCreateFile* — Создает файл для записи. *TMCreateProgramJob* — Создает буфер записи для регистрации незавершенных заданий программирования СППЗУ. *TMDeleteFile* — Удаляет файл.

TMDirectoryMR — Создает или удаляет поддиректорию. *TMDoProgramJob* — Записывает незавершенные задания программирования СППЗУ. *TMFirstFile* — Находит первый файл в текущей директории. *TMFormat* — Форматирует файловую систему файловой структуры 1-Wire. *TMNextFile* — Находит следующий файл в текущей директории. *TMOpenFile* — Открывает файл для чтения. *TMReadFile* — Читает открытый файл. *TMReNameFile* — Переименовывает файл или директорию. *TMTerminateAddFile* — Завершает «AddFile». «AddFile» — это специальный тип файла в приборе с СППЗУ, который может быть добавлен без перезаписи. *TMWriteAddFile* — Добавляет или изменяет «AddFile» в приборе СППЗУ. *TMWriteFile* — Записывает в файл, который был создан.
УРОВЕНЬ ПРИБОРА
Функций нет
ПРОЧИЕ
TMGetTypeVersion — Получает информацию о версии для драйвера адаптера. *Get_Version* — Получает полную версию драйвера.

В Примере 4, приведенном ниже, показан фрагмент кода TMEX, который соответствует блок-схеме использования API, изображенной на Рис. 3. Для простоты предполагается, что обнаружение каждого прибора в сети 1-Wire происходит при каждом прохождении через рабочий цикл. Более сложное приложение, например, могло бы осуществлять поиск приборов только одного типа или, возможно, выбирать прибор, найденный в предыдущем поиске.

Пример 4. Пример кода «Си» TMEX

```
   int PortNum, PortType; // установить номер порта и тип для имеющегося
адаптера
   long session_handle; // дескриптор сеанса
   unsigned char state_buffer[5120];
   int doing_work=1, did_setup=0;
   short rslt;
   // рабочий цикл
   while (doing_work)
   {
   // запросить исключительное использование сети 1-Wire (СЕАНСОВЫЙ УРОВЕНЬ)
   session_handle = TMExtendedStartSession(PortNum,PortType,NULL);
   if (session_handle > 0)
   {
   // проверить, был ли TMSetup уже однажды выполнен
   if (!did_setup)
   {
   if (TMSetup(session_handle) == 1)
   did_setup = 1;
   else
   {
   // ошибка установки порта, возможно отсутствует адаптер
   // . . .
   }
   }
   else
   {
   // найти все приборы (СЕТЕВОЙ УРОВЕНЬ)
   rslt = TMFirst(session_handle, state_buf);
   while (rslt > 0)
   {
   // выполнить КАКУЮ-ТО ОПЕРАЦИЮ с найденным прибором (УРОВНИ
   // ТРАНСПОРТНЫЙ/ФАЙЛОВЫЙ/ПРИБОРА)
   // . . .

   // найти следующий прибор (СЕТЕВОЙ УРОВЕНЬ)
   rslt = TMNext(session_handle, state_buf);
   }
   }

   // освободить сеть 1-Wire от исключительного использования (СЕАНСОВЫЙ
   // УРОВЕНЬ)
   TMEndSession(session_handle);
   }
   else
   {
   // не смог получить исключительное использование сети 1-Wire
   // . . .
   }
   // выполнить другое задание приложения
   // . . .
   }
```

УСТАНОВКА

API TMEX поставляется вместе с автоматизированной программой установки Install-Shield, которая загружает все драйверы Windows и разделы системного реестра для каждого поддерживаемого адаптера 1-Wire. Установка доступна как с программой просмотра iButton, так и без нее. Предоставляются также файлы для заказной (определяемой пользователем) установки. Программа просмотра iButton является демонстрационной программой для изучения возможностей большинства приборов 1-Wire, включая iButton.

ДРУГИЕ ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА

Хотя четыре API, рассмотренных в этом документе, представляют большую часть доступных ресурсов для работы с приборами 1-Wire, имеются и другие ресурсы. Некоторые из них описываются в данном разделе.

IB-IDE

iB-IDE обеспечивает готовую среду программирования, предназначенную для ускорения разработки программного обеспечения для iButton, поддерживающих Java. Приложение Java-хоста, находящееся на персональном компьютере или во встроенной системе, осуществляет обмен данными с любыми апплетами, установленными в iButton, и с введением iB-IDE разработка подобного приложения и апплетов становится проще. Некоторые основные характеристики iB-IDE приведены ниже:

- полнофункциональное моделирующее устройство прибора iButton, поддерживающего Java, с отладкой на уровне исходных кодов Java;
- встроенный текстовый редактор с выделением ключевых слов Java, макросами, поиском и заменой, и ориентированным на язык Java форматированием;
- объединение возможностей компилировать и выполнять Java-программы, а также способность обмениваться данными и управлять приборами iButton, подсоединенными к вашей системе.

Начиная с версии 2.0Beta, в качестве ядра для осуществления обмена данными с iButton, поддерживающим Java, в iB-IDE используется API 1-Wire для Java.

Найти информацию по iB-IDE и скачать эту программу можно на сайте: http://www.ibutton.com/iB-IDE/.

АВТОРИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Авторизация программного обеспечения представляет собой просто блокировку программного приложения, если отсутствует аппаратный маркер. Маркером в этом случае является iButton, подсоединенный к рабочей станции пользователя. Приложение запрашивает наличие и проверяет достоверность iButton перед запуском, а в перспективе и при выполнении приложения. На практике для данного типа приложения может быть использован любой из API, представленных в настоящем документе, однако необходимо принимать во внимание следующие моменты. С точки зрения безопасности, внешние драйверы представляют слабое звено, позволяя пользователю заменить драйвер и, таким образом, аннулировать блокировку. API для авторизации программного обеспечения был разработан специально для такого типа приложений, и вместо внешних драйверов он включает сцепляемые модули.

Информацию о комплекте авторизации программного обеспечения можно найти на сайте: http://www.ibutton.com/software/softauth/index.html.

ГРУППА СПЕЦИАЛИСТОВ, ИНТЕРЕСУЮЩИХСЯ РАЗРАБОТКОЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 1-WIRE

Разработчикам 1-Wire рекомендуется присоединиться к форуму «1-Wire Software Developer's Forum». Цель этой группы — исследовать, обсуждать и отвечать на вопросы, касающиеся разработки приложений, инструментальных средств и применения изделий семейства 1-Wire. За этим форумом следят инженеры-разработчики приложений фирмы Dallas Semiconductor с тем, чтобы отвечать на вопросы, поднимаемые данной группой. Кроме того, здесь анонсируются обновления API и комплектов загрузки. Для подписки следует обращаться на сайт: http://lists.dalsemi.com/.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОТ СТОРОННИХ ФИРМ

Для приборов 1-Wire имеется большой объем стороннего программного обеспечения. Некоторая часть этого программного обеспечения представляет собой приложения, которые разработаны ASD (Authorized Solutions Developer) от Dallas Semiconductor и предназначены для приобретения с их «разрешения». Полный список всех ASD и локатор решений представлены на сайте: http://dbserv.maxim-ic.com/ibutton/solutions/search.cfm. Имеются также проекты с открытыми исходными кодами на общедоступных форумах, таких как «Source Forge» (http://sourceforge.net/).

ЗАКЛЮЧЕНИЕ

В настоящем документе был представлен обзор характеристик всех API 1-Wire. Здесь также были подробно представлены четыре различных API, включая поддерживаемые ими платформы и языки программирования. Краткая справочная таблица, предоставляющая информацию о каждом API для каждого типа прибора, помогает упростить процесс выбора API для использования. Имея в своем распоряжении подходящий API, можно быстро и легко создавать приложения 1-Wire.