DALLAS SEMICONDUCTOR

应用笔记 155

1-Wire 软件资源指南

www.maxim-ic.com

引言

包括 iButtons(信息钮扣)在内，达拉斯半导体公司现在生产的 1-Wire 器件已有 30 多种。如何选择已有的应用编程接口 (API)、软件和其他资源与这一类器件进行通信或为某一类型器件选择正确的资源是一件非常令人头疼的事。本篇指南提供了该类资源的概述和选择指南。本文中所描述的所有 API 都是免费的，而且大多数情况下还包括完整的源代码。

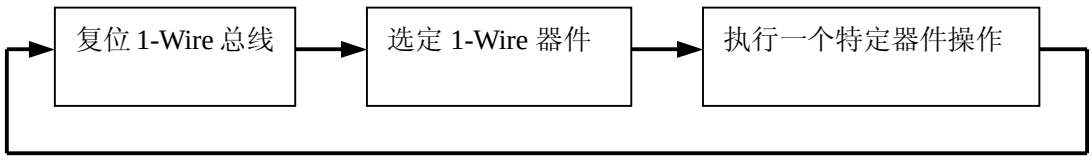
1-Wire（单线）概述

达拉斯半导体公司的 1-Wire 总线是一种简单的信号交换方式，它是在主机与外围器件之间通过一条线路进行双向通信。所有的 1-Wire 总线器件都具有一个共同的特征：无论是芯片内还是 iButton 内，在出厂时每个器件都有一个与其它任何器件互不重复的固定的序列号。也就是说，每一个器件都是唯一的。这就使得在众多连到同一总线的器件中可以选择出任意一个器件。因为一个、二个甚至几十个 1-Wire 器件能共用一条线路进行通信，所以可采用对半检索法依次查找每一个器件。一旦器件的序列号已知，通过访问序列号，任何器件都可被唯一地选出以用于通信。

所有通信的第一步都需要总线控制器发出一个“复位”信号以使总线同步，然后选择一个受控器件进行随后的通信，这可以通过选择所有的受控器件或者选择一个特定的受控器件（利用该器件的序列号进行选择）或者通过对半检索法找到总线上的下一个受控器件来实现。上文所提到的这些指令都是“网络”指令或者 ROM（只读存储器）指令。一旦一个特定的器件被选中，那么在下一次复位信号发出之前，所有其它器件都被挂起而忽略随后的通信。

一旦一个器件被用于总线通信，主机就能向它发出特定器件指令，对它进行数据读写。这是因为每类器件具有不同的功能和不同的用途，而且一旦器件被选定，就有了唯一的协议。虽然每类器件具有不同的协议和特征，但其工作过程却是相同的并且遵循如图 1 所示的工作流程。

典型的 1－Wire 通信流程　图 1



每个受控器件的序列号的整数部分是一个 8 位的家族代码。这个代码对器件模型来说是特定的。因为每种器件模型执行不同的功能，所以可以用代码来选择用于控制或者查询器件的协议。表 1 是达拉斯半导体公司的器件型号的家族代码。

家族代码对照 表 1

家族代码	器件型号（）iButton 封装	说明（除非指明，否则存储器单位为位）
01（十六进制）	(DS1990A)*,DS2401	只做 1-Wire 网络地址（序列号）
02	(DS1991), DS1425	多键 iButton,1152 位安全存储器
04	(DS1994), DS2404	4K NVRAM 存储器和时钟，定时器，报警
05	DS2405	单一的可寻址开关
06	(DS1993)	4K NVRAM 存储器
08	(DS1992)	1K NVRAM 存储器
09	(DS1982), DS2502	1K EPROM 存储器
0A	(DS1995)	16K NVRAM 存储器
0B	(DS1985), DS2505	16K EPROM 存储器
0C	(DS1996), (DS1996x2), (DS1996x4)	64K 到 256K NVRAM 存储器
0F	(DS1986), DS2506	64K EPROM 存储器
10	(DS1920), DS1820, DS18S20	有报警输出的温度传感器
12	DS2406, DS2407	1K EPROM 存储器，2 路地址开关
14	(DS1971), DS2430A	256 位 EPROM 存储器和 64 位 OTP 寄存器
18	(DS1963S)	4K NVRAM 存储器和 SHA-I 引擎
1A	(DS1963L)	具有写周期计数器的 4K NVRAM 存储器
1D	DS2423	具有外部计数器的 4K NVRAM 存储器
1F	DS2409	用于子网的 2 路地址开关
20	DS2450	4 路 ADC
21	(DS1921), (DS1921H), (DS1921Z)	Thermochron 温度记录器
22	DS1822	经济型温度传感器
23	(DS1973), DS2433	4K EEPROM 存储器
24	(DS1904), DS2415	实时时钟
26	DS2438	温度传感器，A/D
27	DS2417	可中断的实时时钟
28	DS18B20	可调节分辨率的温度传感器
2C	DS2890	单路数字电位器
30	DS2760	温度传感器，电流，A/D
33	(DS1961S),DS2432	带 SHA-1 引擎的 1K EEPROM 存储器
91	(DS1981)	512 位 EPROM 存储器(Uniqueware only)
96	(DS1955),(DS1957B)	Java-有动力装置的加密的 iButton(64K 字节 ROM，6 到 134K 字节的 NVRAM)

API 基础

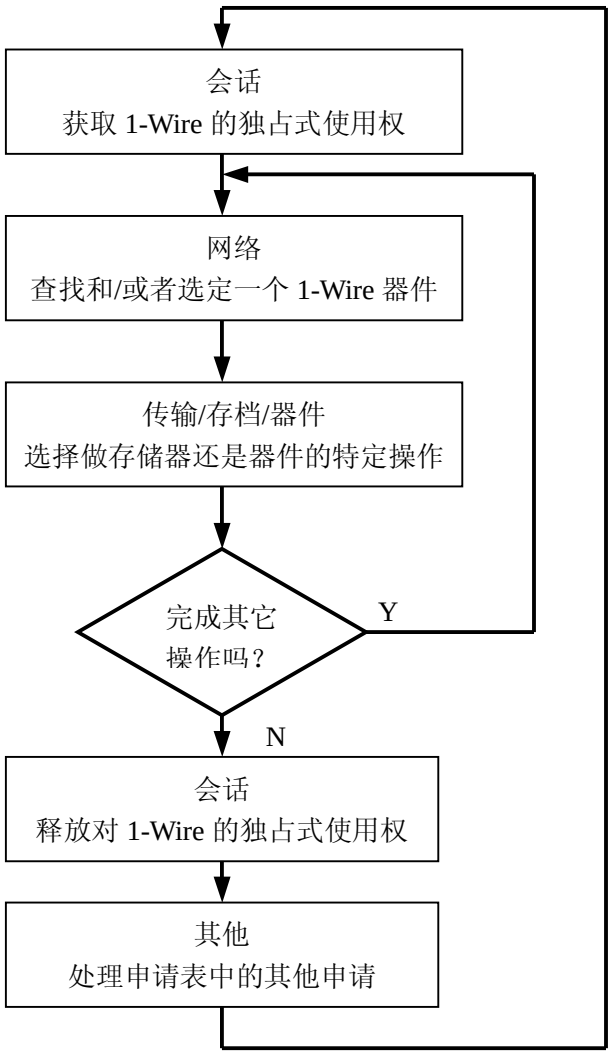
1-Wire 通信器件所使用的不同的 API 有着共同的特性，这反映出源于协议的信息交换的基本原理。图 2 是根据不同 API 功能进行的分类。因为大多数 1-Wire 器件具有存储器，尽管存储器输入输出功能并不适用于所有的器件，我们还是把它们视为一个通用 API 集。其他所有不具备存储器专用功能的划分为一类——专用器件集。

API 功能集（图 2）

会话
分时使用总线。这对于操作系统或几个进程或线程要求同时使用总线的情况下是非常重要的。当多项操作在同一器件上运行而又不能被打断的时候，需要独占总线的使用权。
链路
基本的 1－Wire 总线通信功能。所有的 1－Wire 总线的通信功能可以归结为：复位所有的器件和读写位。这也包括设置总线电特性的功能，如提供专用的 EPROM 编程脉冲或进行供电。
网络
查找和选择器件的网络功能。每个 1－Wire 总线器件都有一个固定的序列号，作为它的唯一的网络地址。这些功能可以通过链路类功能进行构造。在 1－Wire 器件数据表中将其称为 ROM 指令，这是因为序列号是只读的。还有一些 1－Wire 控制器包括一些内置功能，它们比通过链路功能进行构造有效得多。
传输
块通信和基本的存储器读/写功能。它也包括存储器包读/写功能。这些功能是通过网络和链路类功能构造的。
文件操作
使用 1-Wire 文件结构（见应用笔记 114）的文档存储层功能。这些功能是通过网络和传输类功能构造的。只对使用多页存储器的器件有效。
器件
专用器件的高层功能。这些功能通常是通过网络、传输和链路类功能构造的，并执行诸如读取温度值或设置开关状态等功能。

图 3 概括了使用这些功能的典型顺序。“会话”功能围绕着调用器件进行通信，具有代表性的是先使用一个“网络”功能，然后运行存储器或“器件”的特定操作。

API 用法流程 图 3



iButton 通信实质上是通过与其触头相接触来实现的。这意味着与器件的联系有时是不可靠的。iButton 也可能被安装到阅读器里，在阅读的时候弹出，从而必须有一个相容的纠错方法紧跟其后。当检测到虚假错误时必须重发数据并在数据通信中进行 CRC 校验。API 中的文件输入输出功能所利用的标准文件结构在应用笔记 114“1-Wire 文件结构”中进行了详细的阐述。这种结构在每页数据上都使用 CRC16，以快速地校验所读数据的正确性。大多数 1-Wire API 功能很少或者不能自动重发。重发受应用软件控制。应用笔记 159“极其可靠的 1-Wire 通信”讲述了纠错方法论和 1-Wire 通信中的风险评估。

API 的选择

本文主要列举了四种不同的应用程序接口(API)。这些 API 运行在不同的工作平台上，使用不同的语言，具有不同的性能。表 2 简要地描述了这四种 API。

API 概述 表 2

API	缩写	说明
1-Wire 公众域	PD	完全公开的 C 语言代码，在许多平台上支持串行 DS9097U 适配器
基于 Java 的 1-Wire API	OWAPI	完全公开资料的 Java 程序，它支持几乎所有的 1-Wire 器件。它所支持的唯一的异地主机是 DS9097U 串行适配器。
1-Wire COM	OWCOM	Windows 组件对象模型（COM），OWAPI 的外壳，可以从标准语言和诸如 Java 和 VB 的脚本语言中获得。
TMEX API	TMEX	在 32 位 Windows 平台上支持所有的 1-Wire 主适配器。提供链接和文件输入输出功能但没有器件功能。驱动器是不公开的资源。这种 API 可以被其他的 API 调用来存取所有类型的 1-Wire 适配器。

表 3 说明了按语言划分的可利用 API 及其操作系统。*注：“TINI”是达拉斯半导体公司 http://www.ibutton.com/TINI 研制的基于 Java 操作系统的内置平台。

API 操作系统和语言适用范围 表 3

语言 / 操作系统	独立的 Windows 语言	C	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)
Win 3.1		PD	
DOS		PD	
PALM		PD	
VISOR		PD	
PocketPC		PD	
Solaris		(PD)	OWAPI
Linux		PD	OWAPI
TINI*		(PD)－w/o TINI OS	OWAPI
( )－计划支持但尚未实现			

API 不同，器件族的支持亦不同。表 4 列出了当前所有可利用的 1-Wire 器件，并用标志说明它们各受哪种 API 支持。标志的解释在表的底部。注：表中无阴影的器件单元全部受 API 支持，淡阴影的单元只是部分地受 API 支持而浓阴影的单元受 API 支持的是最少的。

器件支持的 API　表 4

器件	族号	说明	TMEX	PD	OWAPI	OWCOM
DS1425	02	多键 iButton,1152B 加密存储器	AB	AB**I**	ABC	ABC
DS1427	04	4K NVRAM 存储器和时钟、定时器、报警	ABDE	ABCDE**I**	ABCDEFGHI	ABCDE**FGH**I
DS1820	10	有报警输出的温度传感器	AB	ABI	ABCI	ABCI
DS18B20	28	可调节分辨率的温度传感器	AB	AB	ABI	ABI
DS18S20	10	有报警输出的温度传感器	AB	ABI	ABCI	ABCI
DS1981	91	512B EPROM存储器 (Uniqueware only)	ABCE	AB	AB	AB
DS1982	09	1K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E**
DS1985	0B	16K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E**
DS1986	0F	64K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E** **FGH**
DS1904	24	实时时钟	AB	AB	ABI	ABI
DS1920	10	有报警输出的温度传感器	AB	ABI	ABCI	ABCI
DS1921 DS1921H DS1921Z	21	Thermochron 温度记录器	ABDE	ABCDEI	ABCDEFGHI	ABCD**E** **FGH**I
DS1955 DS1957	96	Java 驱动的加密 iButton(64K 字节 ROM，6 到 134K 字节的 NVRAM)	AB	ABI	ABI	ABI
DS1961S	33	带 SHA-1 引擎的 1K EEPROM 存储器	ABDE	ABCDEI	ABCDEFGHI	ABCD**E** **FGH**I
DS1963L	1A	带写周期计数器的 4K NVRAM 存储器	ABDE	ABCDE	ABCDEFGH	ABCD**E** **FGH**
DS1963S	18	4K NVRAM 存储器和 SHA-I 发动机	ABDE	ABCDEI	ABCDEFGHI	ABCD**E** **FGH**I
DS1971	14	256 位 EEPROM 存储器和 64 位 OTP 寄存器	ABD	ABCDI	ABCDI	ABCDI
DS1973	23	4K EEPROM 存储器	ABDE	ABCDE	ABCDEFGH	ABCD**E** **FGH**
DS1990A	01	只有 1-Wire 寻址	AB	AB	AB	AB
DS1991	02	多键 iButton,1152B 安全存储器	AB	ABC	ABC	ABC
DS1992	08	1K NVRAM 存储器	ABDE	ABCDE	ABCDEFGH	**ABCD**EFGH
DS1993	06	4K NVRAM 存储器	ABDE	ABCDE	ABCDEFGHI	ABCD**E** **FGH**
DS1994	04	4K NVRAM 存储器和时钟，定时器，警报	ABDE	ABCDE**I**	ABCDEFGHI	ABCD**E** **FGH**I
DS1995	0A	16K NVRAM 存储器	ABDE	ABCDE	ABCDEFGH	ABCD**E** **FGH**

DS1996	0C	64K NVRAM 存储器	ABDE	ABCDE	ABCDE FGH	ABCD**E** **FGH**
DS1996x2 DS1996x4	0C	64K*2(128K)NVRAM 存储器 64K*4(256K)NVRAM 存储器	ABDE	ABCDE	ABCDE FGH	ABCD**E** **FGH**
DS2401	01	只有 1-Wire 寻址	AB	AB	AB	AB
DS2405	05	单路开关	AB	ABI	ABI	ABI
DS2404	04	4K NVRAM 存储器和时钟，定时器，警报	ABDE	ABCDE **I**	ABCDE FGHI	ABCD**E** **FGH**I
DS2406 DS2407	12	1K EPROM 存储器，2 路可寻址开关	ABCE	ABCDE I	ABCDE I	ABCD**E** I
DS2409	1F	双路开关，耦合器	AB	ABI	ABI	ABI
DS2415	24	实时时钟	AB	AB	ABI	ABI
DS2417	27	可中断的实时时钟	AB	AB	ABI	ABI
DS2423	1D	具有外部计数器的 4K NVRAM 存储器	ABDE	ABCDE I	ABCDE FGHI	ABCD**E** **FGH**I
DS2430A	14	256 位 EEPROM 存储器和 64 位 OTP 寄存器	ABD	ABCDI	ABCDI	ABCDI
DS2432	33	带 SHA-1 引擎的 1K EEPROM 存储器	ABDE	ABCDE I	ABCDE FGHI	ABCD**E** **FGH**I
DS2450	20	四 A/D	AB	ABI	ABI	ABI
DS2438	26	温度传感器，A/D	AB	ABI	ABCI	**ABCI**
DS2502	09	1K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E**
DS2505	0B	16K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E**
DS2506	0F	64K EPROM 存储器	ABCE	ABCDE	ABCDE	ABCD**E** **FGH**
DS2760	30	温度传感器，电流，A/D	AB	AB	ABI	ABI
DS2890	2C	单路数字电位器	AB	AB	ABI	ABI

阴影标志

全部支持	部分支持	很少支持

支持标志说明

A. 支持 1-Wire 简单连接
B. 支持 1-Wire 网络
C. 支持存储器的字节读写
D. 支持存储器的包读写
E. 支持 AA 型 1-Wire 文件结构（见应用笔记 114：文件结构类型）
F. 支持 AB 型 1-Wire 文件结构类型
G. 支持 BA 型 1-Wire 文件结构类型
H. 支持 BB 型 1-Wire 文件结构类型
I. 支持其它专用器件

黑体字标志

表明现在预发行的 API 尚不支持这一特性，最终发行的版本将支持。

公用(PD)1-Wire 概述

公用 1-Wire API 提供的功能都是用 C 语言写的，用于 TMEX API 所不支持的平台。“1-Wire net”（或者 Micro LAN）是一种在主机和一个或多个外设之间只有一条数据线和地线的网络。这种 API 使得 1-Wire 处于主动地位，从而可以用来识别外设并与之进行通信。它提供所有的 1-Wire，传输和文件操作类服务，可以与包括 iButton 在内的所有达拉斯半导体公司的 1-Wire 器件进行通信。

该类 API 软件包和平台示例设在：

http://www.ibutton.com/software/1wire/wirekit.html

用 C 语言设计的 API 是可移植的。那有在专用平台上完成的“TODO”模板，也提供了几种平台范例的实现，包括 32 位 Windows，Linux，Visor，Palm 和 PockerPC。还有用这些平台实现的几个应用。

有两个可移植的源文件。第一个是通用源文件，适用于已经与 1-Wire 通信功能建立了基本连接的平台（通用），是与硬件有关的最底层源文件。另一个是假设用户已经有了一个串口（如 RS232），并请求使用“通用串行 1-Wire 总线驱动器：DS2480B”（专用）。这种芯片从串口读入命令后，执行 1-Wire 操作，然后再把结果送回串口。源代码把预期的 1-Wire 操作转换为串行通信包传送给 DS2480B。唯一需要提供给平台的参数就是串口的读写初始值。所有 DS9097U 系列的串行适配器的接口芯片用的都是 DS2480B。

这两个可移植的源代码文件实现的是同样的 1-Wire API 功能，并且它们之间可以相互转换。图 3 列出了适用于 1-Wire 公用代码库（版本 3.00）的可利用 API 功能。注：由于不具有存储功能的器件的专用功能太多，因而在表中没有详细地列出。表 4 列出了提供功能和创建平台所需模块的源文件。

在该网站除了可以下载可移植的 C 语言模块外，还可以下载一些进行 1-Wire 通信的微处理器汇编实例。

应用笔记 114 中所定义的“AA”型 1-Wire 文件结构的文档功能可以在：http://pdfserv.maxim-ic.com/arpdf/AppNotes/app114.pdf 找到。

正如 API 的名字所暗示的那样，所提供的源代码有一个尽可能接近公用的许可证。开发者可以自由使用，并可与他们的应用软件不受任何限制地结合为一体。

公用 API 功能 图 3

会话
owAcquire－获得 1-Wire 网的独占式使用权
owRelease－释放以前得到的 1-Wire 网独占式使用权
链接
owHasOverDrive－说明适配器是否具有快速适配性能
owHasPowerDelivery－说明适配器是否能供电
owHasProgramPulse－说明 EPROM 编程电压是否可用
owLevel－设置传输线上某点电平为常态（5V 电源加小阻值上拉电阻），供电（5V 电源加强上拉），或者编程电平（12V EPROM 编程电平）
owProgramPulse－发同步编程脉冲写 EPROM 1-Wire 器件
owReadBitPower－读取 1 位数据，并有选择地提供供电
owReadByte－通过发送全 1（即 0xFF）接收来自 1-Wire 网的 8 位数据
owSpeed－设置 1-Wire 网的速度为常态（16K bits）或快速（142K bits）
owTouchBit－发送并接收 1 位 1-Wire 网的数据
owTouchByte－发送并接收 8 位 1-Wire 网的数据
owTouchReset－复位 1-Wire 网上的所有器件并返回结果
owWriteByte－向 1-Wire 网发送 8 位数据并验证收到的应答是否匹配
owWriteBytePower－向 1-Wire 网发送 8 位通信数据然后供电
网络
owAccess－选定当前器件并准备运行特定器件的指令
owFamilySearchSetup－确定随后的搜索顺序（owNext）以找到特定的家族类型
owFirst－搜索并找到 1-Wire 网上的第一个 1-Wire 器件
owNext－搜索并找到 1-Wire 网上的下一个 1-Wire 器件
owOverdriveAccess－选择当前的器件，并设置它的速度为快速
owSerialNum－保持或设定当前选定的器件的序列号（ROM 号）
owSkipFamily－略过最后一次查询到的家族类型的所有器件
owVerify－选择并验证当前的器件正在运行（报警选项）
传输
owBlock－发送和接收 1-Wire 网上的一个数据块，并有选择地进行复位
owCanLockPage－检查给定的内存条是否有可被锁定的页
owCanLockRedirectPage－检查给定的内存条是否有可被锁定的以防被重定向的页
owGetAlternateName－获得改变的零件号或零件名
owGetBankDescription－获得内存条的字符性描述
owGetDescription－获得对 1-Wire 器件类型的简单描述
owGetExtraInfoDesc－获得对附加信息的内容的描述
owGetExtraInfoLength－获得内存条中附加信息的长度，单位为字节
owGetMaxPacketDataLength－获得一个数据包中最大的数据长度，单位为字节
owGetName－以字符串的形式获得 1-Wire 器件的零件号
owGetNumberBanks－获得某一 1-Wire 家族类的内存条号
owGetNumberPages－获得给定内存条的页数
owGetPageLength－获得给定内存条的原始页长度，单位是字节
owGetSize－获得给定内存条的大小，单位是字节

o*wGetStartingAddress*－获得给定内存条的物理首地址
owHasExtraInfo－检测读数据时内存条的页是否传送附加信息
owHasPageAutoCRC－检测读数据时内存条是否有器件产生的 CRC 校验码
owIsGeneralPurposeMemory－检测内存条是否是通用用户存储器
owIsNonvolatile－检测当前的内存条是否是永久性的
owIsReadOnly－检测内存条是否是只读的
owIsReadWrite－检测内存条是否是可读写的
owIsWriteOnce－检测内存条是否是只可写入一次，如 EPROM
owNeedsPowerDelivery－检测写内存条时是否需要“供电”
owNeedsProgramPulse－检测写内存条时是否需要“编程脉冲”
owRead－读初始状态下内存条的一部分（没有包，CRC 校验码）
owReadPage－读初始状态下内存条的完整的一页（没有包，CRC 校验码）
owReadPageCRC－在器件产生校验码的状态下，读内存条的完整的一页
owReadPageExtra－读初始状态下的包括所有附加信息在内的内存条的完整的一页（没有包，CRC 校验码）
owReadPageExtroCRC－读取包括所有附加信息和器件生成的 CRC 校验码在内的内存条的完整的一页
owReadPagePacket－从内存条的页中读取通用数据包（见应用笔记 114：通用数据包结构概述）
owReadPagePacketExtra－从含有附加信息的内存条的页中读取通用数据包
owRedirectPage－检测内存条中是否有能被重定向的页
owWrite－写初始模式下的一部分内存条
owWritePagePacket－把一个通用数据包写入内存条的一页中

文件操作

owAttribute－改变文件的属性
owChangeDirectory－改变当前目录
owCloseFile－关闭一个文件
owCreateDir－创建一个目录
owCreateFile－创建一个新文件
owCreateProgramJob－创建一个写缓冲器，以记录 EPROM 编程未完成的任务
owDeleteFile－删除一个文件
owDoProgramJob－完成未完成的 EPROM 编程任务
owFirstFile－寻找当前目录下的第一个文件
owFormat－格式化 1-Wire 文件结构文件系统
owGetCurrentDir－获取当前目录
owNextFile－寻找当前目录的下一个文件
owOpenFile－打开一个可读文件
owReadFile－读一个已打开的文件
owReadFile－从文件读取数据
owRemoveDir－删除一个目录
owReNameFile－更改文件名
owWriteFile－写一个已创建的文件

器件
DoAtoDConversion－在 DS2450 上进行 A/D 转换 *ReadSwitchI2*－读取 DS2406 开关的状态 *GetFirmwareVersionString*－获取 Java 驱动的 iButton 固件版本描述符 …… 所有的专用器件的功能太多而不能一一列举

下面的例 1 就是一段公用代码，它遵循着图 3 所概括的“API 用法流程”。简便起见，1-Wire 网络上的器件在程序的循环中一次找到一个。更复杂的应用一次可能找到一类器件或者可能选定一个在上次查找过程中找到的器件。

公用代码示例　例 1

```
Int    rslt , portnum = 0 , doing work =1 ;
char portString [50] ;    // set to platform appropriate port string

// work loop
while    (doing_ work)
{
    // acquire the 1-Wire Net    (SESSION)
    if    (owAcquire (portnum , portString) )
    {
        // find all devices (NETWORK)
        rslt = owFirst (portnum, TRUE, FALSE) ;
        while (rslt)
        {
            // do SOMETHING with device found (TRANSPORT/FILE/DEVICE)
            // .   .   .

            // find the next device    (NETWORK)
            rslt = owNext ( portnum, TRUE, FALSE);
         }
         //    release the 1-Wire Net (SESSION)
         owRelease ( portnum) ;
      }
      else
      {
          // Could not acquire 1-Wire network
          // .   .   .
      }
      // do other application work
      // .   .   .
  }
```

图 4（a 和 b）列出了 C 语言模块，它们分别构成了 1-Wire 的两个公用程序库，同时也列出了必需的“TODO”功能，以把程序库传送到新平台。在这个软件包里也有执行“TODO”功能的几个平台链接文件的例子。

公用“用户”执行程序 图 4a

会话					
owsesu.c					
链接					
owllu.c	ds2480ut.c	ds2480.h			
网络					
ownetu.c	crcutil.c				
传输					
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	owtmu.c	rawmem.c
rawmem.h					
文件操作					
owcache.c	owfile.c	owfile.h	owpgrw.c	owprgm.c	
器件					
ad26.c	ad26.h	atod20.c	cntld.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	swtlf.c	temp10.c
thermo21.c	thermo21.h	weather.c	weather.h		
杂项功能					
ioutil.c	owerr.c	findtype.c	ownet.h	screenio.c	sprintf.c
TODO					
提供一个可以运行以下功能的串口模块： *BreakCOM**－在串口上发送一个至少持续 2 毫秒“BREAK”指令 *CloseCOM*－关闭以前打开的串口（只对一些平台适用） *FlushCOM**－允许任意未完成的写操作执行完毕，并清空输入缓冲器 *msDelay**－至少延迟指定的毫秒数 *msGettick*－返回一个增量毫秒计数器（只对一些平台适用） *OpenCOM*－打开指定的串口以进行通信 *ReadCOM**－从串口读取指定数目的字节 *SetCOMBand*－更改串口的波特率为指定值（也可设置为快速） *WriteCOM**－写指定数目的字节到串口 *基本操作所必需的功能					

公用“通用”执行程序 图 4b

会话
(见 TODO)
链接
（见 TODO）
网络
ownet.c crcutil.c
传输
除 owtrnu.c 换为 owtran.c 外，其余的均同专用执行程序
文件操作
同专用执行程序
器件
同专用执行程序
杂项功能
同专用执行程序
TODO
提供可以运行以下功能的链接和会话接口： *owAcquire*－获得 1-Wire 网 *owRelease*－释放以前得到的 1-Wire 网 *owHasOverDrive*－说明适配器是否具有快速适配性能 *owHasPowerDelivery*－说明适配器是否能供电 *owHasProgramPulse*－说明是否 EPROM 编程电平是否可用 *owLevel*－设置传输线上某点电平为常态（5V 加小阻值上拉电阻），供电（5V 电源加强上拉），或者为编程电平（12V EPROM 编程电平） *owProgramPulse*－发送同步编程脉冲写 EPROM 1-Wire 器件 *owReadBitPower*－读取 1 位数据，然后有选择地提供供电 *owReadByte*－通过发送全 1 信号（0xFF），接收来自 1-Wire 网的 8 位数据 *owSpeed*－设置 1-Wire 网的速度为常态（16K bits）或者快速（142K bits） *owTouchBit**－发送并接收一位 1-Wire 网的数据 *owTouchByte*－发送并接收 8 位 1-Wire 网的数据 *owTouchReset**－复位 1-Wire 网上的所有器件并返回结果 *owWriteByte*－向 1-Wire 网发送 8 位数据，并验证响应是否匹配 *owWriteBytePower*－向 1-Wire 网发送 8 位通信数据，并提供供电 *执行基本操作所必须的功能

安装

由于 1-Wire 公用 API 是一套 C 模块，所以没有正规的安装程序。正如例子“builds”那样，所需的模块被直接编译到应用程序里。当然这并不排除开发者把模块组合到可加载的程序库里，比如 windows dll。

JAVA 的 1-Wire API（OWAPI）概述

为创建 1-Wire 在 Java 中的应用，适用于 Java 的 1-Wire API 从一开始就被设计成一种高度面向对象的基础牢靠的 API。它增强了程序员设计可移植的交互平台软件的能力，并缩短了基于 1-Wire 器件设计开发的产品进入市场的时间。

很多 Java 类和接口构成 API。1-Wire API 的 Java 类中有一个特殊的库：软件包（1-Wire 软件包）。软件包支持包括 iButtons 在内的特殊 1-Wire 器件。API 有 30 多种软件包类型，代表了大多数 1-Wire 器件。每个软件包都可以压缩和执行单一器件的功能。

“软件包”通过 1-Wire 物理适配器（DS 端口适配器类）和 1-Wire 器件进行通信。适配器由提供者（1-Wire 访问提供者类）生产。1-Wire 适配器会由于平台的不同而不同，但是它们都有同样的接口。有一些平台使用本地适配器，但大多数支持使用 Java 通信 API（或者等价物）的 DS9097U-XXX 串行适配器。这种 API 可以从 Sun 网站上找到：http://java.sun.com/products/javacomm.

Java 软件包的 1-Wire API 在 http://www.ibutton.com/software/lwire/lwire_api.html 可以找到。

就象 1-Wire 公用软件包一样，通过“公用”许可证也可以得到 OWAPI 的完整的 Java 源程序。

API 对象生成的典型顺序见图 5。“提供者”提供一个适配器实例（或列举），反过来它又生成器件“软件包”实例。然后与器件的通信几乎直接通过“软件包”来进行。

OWAPI 对象生成　图 5

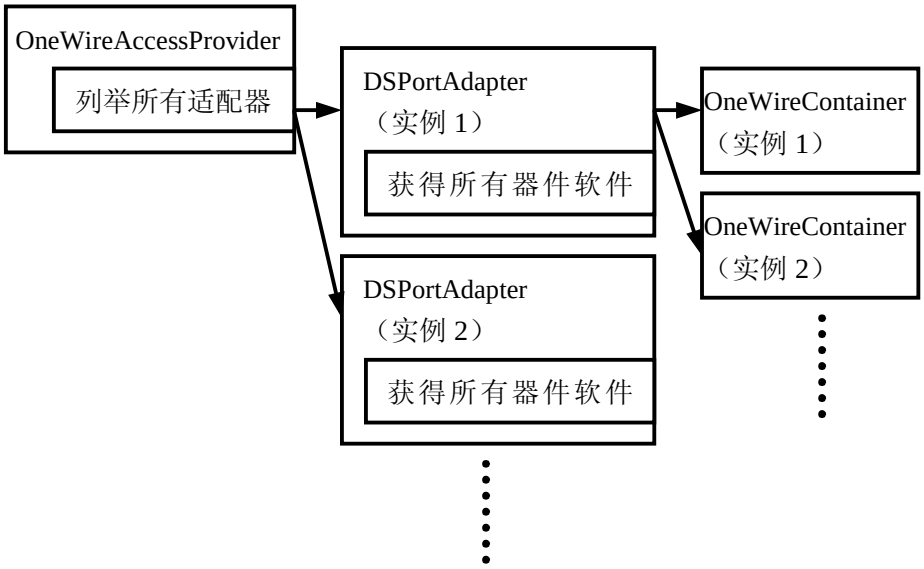


图 6 说明了“软件包”的共同特征。具有存储器的器件将为每个内存条生成一个“内存条”实例。根据特征可以把存储器划分成几组。比如：一组存储器是非永久性的，而另一组却是永久性的。又如一组是通用存储器，而另一组是能被“存储映象”以改变器件功能的存储器。

OWAPI ONEWIRECONTAINER 特征　图 6

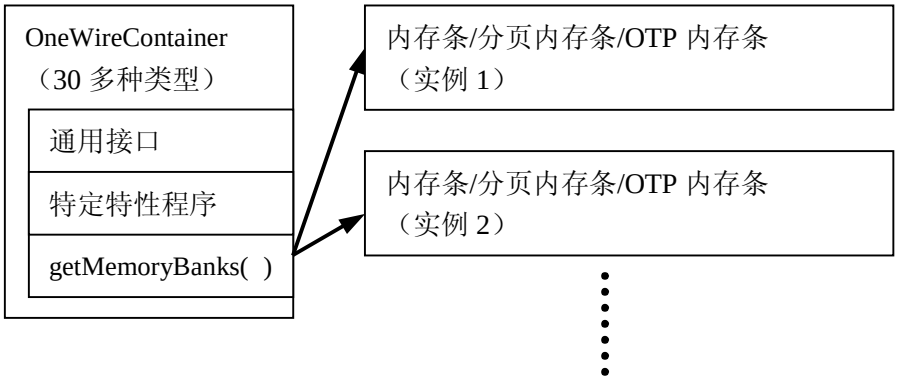


图 6 说明了 OWAPI 类库提供的方法。类或包用**黑体字**表示。注：因为每个软件包都有操作器件类型的高级方法，所以一般不直接访问适配器中链路层的方法。

OWAPI 功能　图 7

会话
com.dalsemi.onewire.adapter.DSPortAdapter *beginExclusive*－获取 1-Wire 网的独占式使用权 *endExclusive*－释放对 1-Wire 网的独占式使用权
链路
com.dalsemi.onewire.adapter.DSPortAdapter *canBreak*－检测适配器是否支持 1-Wire 的“break”（长时间的低电平）操作 *canDeliverPower*－检测适配器是否支持“强上拉”的供电 *canDeliverSmartPower*－检测适配器是否支持“智能”供电。所谓“智能”供电指的是能够检测出何时能量消耗开始减少并能自动停止供电 *canFlex*－检测适配器是否支持远距离同步通信 *canHyperdrive*－检测适配器是否支持超光速通信 *canOverdrive*－检测适配器是否支持快速通信 *canProgram*－检测适配器是否支持 12V 的 EPROM 编程电压 *dataBlock*－发送和接收 1-Wire 网的一个数据包 *getBit*－从 1-Wire 网读取一位数据 *getBlock*－从 1-Wire 网读取数据块（通过发送全 1（即 0xFF））信号 *getByte*－通过发送全 1（即 0xFF）信号从 1-Wire 网读取一个字节 *getSpeed*－获取当前 1-Wire 网的通信速度 *putBit*－向 1-Wire 网写一位数据 *putByte*－向 1-Wire 网写一个字节数据，并验证响应是否正确 *reset*－复位所有 1-Wire 网上的器件 *setPowerDuration*－设定供电持续时间 *setPowerNormal*－停止供电 *setProgramPulseDuration*－设定编程电平持续时间 *setSpeed*－设定 1-Wire 网通信速度 *startBreak*－在 1-Wire 网上开始一次中断（低电平） *startPowerDelivery*－开始供电 *startProgramPulse*－启动编程脉冲

网络

com.dalsemi.onewire.adapter.DSPortAdapter

excludeFamily－在搜索过程中把一个家族都排除在外

findFirstDevice－不经过软件包的自动生成而找到 1-Wire 网上的第一个器件

findNextDevice－不经过软件包的自动生成而找到 1-Wire 网上的下一个器件

getAllDeviceContainers－找出 1-Wire 网上所有具有软件包的器件

getDeviceContainer－获取当前找到的器件的软件包

getFirstDeviceContainer－找到第一个器件，并为之建立一个软件包

getNextDeviceContainer－找到下一个器件，并为之建立一个软件包

setNoResetSearch－设置 1-Wire 网搜索未被复位的器件

setSearchAllDevices－设置 1-Wire 网搜索所有器件（除了警报）

setSearchOnlyAlarmingDevices－设置 1-Wire 网使其只搜索报警器件

targetAllFamilies－设置 1-Wire 网搜索所有器件（除被排除在外的器件）

targetFamily－在 1-Wire 网的搜索过程中，只搜索某一特定的家族器件

（也在 **com.dalsemi.onewire.container.***里）

isAlarming－检测器件是否处于警报状态

isPresent－检测器件是否在 1-Wire 网上

select－选择 1-Wire 网器件，准备执行器件的特定操作指令

传输

com.dalsemi.onewire.container.MemoryBank

getBankDescription－返回内存条的文本描述

getSize－获取内存条的大小，单位为字节

getStartPhysicalAddress－获得内存条的物理首地址

isGeneralPurposeMemory－检测内存条是否是通用的（不是内存映象的）

isNonVolatile－检测内存条是否是非永久性的

isReadOnly－检测内存条是否是只读的

isReadWrite－检测内存条是否是可读可写的

isWriteOnce－检测内存条是否像 EPROM 一样只能写入一次

needsPowerDelivery－检测内存条写时是否需要供电

needsProgramPulse－检测内存条写时是否需要编程脉冲

read－不译码直接读取内存条（没有包结构）

setWriteVerification－设置 API 写数据后进行校验

write－写内存条（没有包结构）

com.dalsemi.onewire.container.PagedMemoryBank

getExtraInfoDescription－获取和内存条有关的对附加信息的说明

getExtraInfoLength－获取每一页上附加信息的长度，单位为字节

getMaxPacketDataLength－获取包结构所能容纳的最大数据长度，以适合内存条的每一页。

getNumberPages－获取内存条的页数

getPageLength－获取内存条空白页的长度，单位为字节

hasExtraInfo－检测内存条是否有与每一页有关的附加信息

hasPageAutoCRC－检测内存条中的页是否有器件产生的 CRC 校验码

readPage－从内存条中读取一页

readPageCRC－利用器件产生的 CRC 校验码从内存条读取一页数据

readPagePacket－从内存条的一页中读取包结构

writePagePacket－写包结构到内存条的页中

com.dalsemi.onewire.container.OTPMemoryBank

canLockPage－检测内存条的页是否可以写保护

canLockRedirectPage－检测内存条的重定向器件是否可以被锁，以防再次重定向

canRedirectPage－检测内存条是否有被重定向的页，以此更改一次性写入的页

getRedirectedPage－获取被重定向的页的数目

isPageLocked－检测页是否加了写保护

isRedirectPageLocked－检测页是否被锁，以防被重定向

lockPage－锁定一页

lockRedirectPage－锁定页以防被重定向

redirectPage－重定向一页到新页，更新一次性写器件时使用

文件操作

com.dalsemi.onewire.utils.OWFile

除了与 java.io.File（JDK 的版本 1.2）相同的方法以外，还有下面的方法：

close－关闭文件和释放所有相关的资源

format－格式化提供给 OWFile 的与器件有关联的 1-Wire 文件系统

getFD－获得文件的 OWFile 描述，以使文件操作与器件同步

getFreeMemory－获得 1-Wire 文件系统中可利用的剩余存储空间

getLocalPage－从 1-Wire 文件系统页获得内存条的当前页

getMemoryBankForPage－获取可以用来读写 1-Wire 文件系统页的内存条实例

getOneWireContainer－获得构成文件系统的软件包

getPageList－获得含有文件的 1-Wire 文件系统列表

com.dalsemi.onewire.utils.OWFileDescriptor

同样的方法见 java.io.FileDescription（JDK 的版本 1.2）

com.dalsemi.onewire.utils.OWFileOutputStream

同样的方法见 java.io.FileOutputStream（JDK 的版本 1.2）

com.dalsemi.onewire.utils.OWFileInputStream

同样的方法见 java.io.FileInputStream（JDK 的版本 1.2）

器件

com.dalsemi.onewire.container.*

包括六种不同的传感器类接口，共有 30 多种器件专用软件包执行程序：

ADContainer－模数转换器

ClockContainer－时钟

SwitchContainer－开关

TemperatureContainer—温度传感器

PotentionmeterContainer－数字电位器

HumidityContainer－湿度传感器

com.dalsemi.onewire.application.*

安全散列算法（SHA）和 1-Wire 标志实用程序类

com.dalsemi.onewire.jib.*

提供 OpenCard 以支持 Java 环境下的 iButton（有关 Opencard 的信息可在 http://www.opencard.org 查到）。

下面的例 2 是 OWAPI 的代码段，它也遵循图 3 所概括的“API 用法流程”。为简便起见，每次循环只找出一个 1-Wire 网上的器件。复杂点的应用程序一次循环中可以找出一类器件或者可以选定在上次循环中找到的器件。

OWAPI 代码实例　例 2

```
boolean doing_working = true;

// get the default adapter from the service provider
DSPortAdapter adapter = OneWireAccessProvider.getDefaultAdapter ( ) ;
// work loop
while (doing_work)
{
   // get exclusive use of adapter (SESSION)
   adapter.beginExclusive ( true ) ;
   // clear any previous search restrictions (NETWORK)
   adapter.setSearchAllDevices ( );
   adapter.targetAllFamilies ( ) ;
   adapter.setSpeed (adapter.SPEED_REGULAR ) ;
   // enumerate through    all the 1-Wire devices found (NETWORK)
   for (Enumeration owd_enum = adapter.getAllDeviceContainers ( ) ;
         owd_enum.hasMoreElements ( ) ;}
   {
      // get a ‘container’ for each device
      OneWireContainer owd = (OneWireContainer ) owd_enum.nextElement ( ) ;
      // do SOMETHING with device found (TRANSPORT/FILE/DEVICE)
      // . . .
    }
    // end exclusive use of adapter (SESSION)
    adapter.endExclusive ( ) ;
    // do other application work
    // . . .
}
```

1-Wire 标志

随着 1-Wire 传感器数量和种类的增多，1-Wire 网的管理也越来越困难。比如，象 A/D 这样的传感器可以测量各种不同的值，所以给传感器加上标签来说明它的功能也变得重要起来。一种使用 XML 的 1-Wire 标志方案已经确定并应用于 Java 的 1-Wire API。这些标志允许应用程序动态的载入和设置传感器并赋给它一定的环境。详细内容参见应用笔记 158“使用 XML 的 1-Wire 标志”。

安装

图 7 所提到的调用 API 所需的所有模块都包含在一个 jar 文件里：OneWireAPI.jar。将模块导入正确的位置或类途径提供了全部的 API。但有两个特例：一个是本地或通信 API 需要安装在特定的平台上，另一个是平台有容量的限制以致不能满足所有 API 的需要。这两个特例将在 OWAPI 软件包里进行详细的说明。

1-WIRE COM(OWCOM)概述

1-Wire Windows COM（组件对象模型）结构是 Java 的 1-Wire API（OWAPI）包。几乎所有工作在 32 位 Windows 操作系统下的编程语言都能使用 COM 组件。自从使用了 OWAPI，支持达拉斯半导体公司的 1-Wire 器件的最大类集已适用于包括 C＋＋，VisualBasic，JavaScript，Jscript 和 VBScript 在内的多种编程环境。

因为 Java 和 COM 并不完全兼容，所以必须有几个工作区能使现有的基于 Java 的代码能工作在 COM 接口上。这几个工作区将在下一节进行介绍。

包含 OWCOM（和 TMEX）在内的软件开发包可以在下面的网站找到：http://www.ibutton.com/software/tmex/index.html。

产生 OWCOM 的 Java 源代码已被慷慨地设为公用。唯一未提供的是支持各种不同 1-Wire 适配器（见 TMEX API 部分）的 TMEX API 驱动程序。但是这些驱动程序也可以自由地重新发布。

OWCOM 和 OWAPI 的区别

所有接口的基本结构实际上都是用 OWAPI 来实现的，文档也是一样的。把 Java 文件用于 COM 接口有一些经验：

1. Java 使用 64 位长整型数据，而用于 COM 的接口不支持大于 32 位的数。对于 Java 中的所有功能都可把长整型视为一个参数或者返回一个长整型字符串。在类型要求不严格的编程语言中（如 VisualBasic 或者 JavaScript），长整型和字符串之间的转换是自动完成的而且决不会影响到代码。而在其它语言中，则要特别注意把参数转换为字符串并把返回值转换为 64 位长整型。
2. OWAPI 中的几个类使用字节矩阵来隐藏对象的状态信息。可惜的是，语言不允许传送参考值（象 JavaScript），实际上它是把复制的数据作为参数来传送的。虽然这减弱了把计算结果矩阵作为参数进行传递的功能，但还是能生成一个包含这一字节矩阵的对象。这个对象就是 ByteArray，也是当前在 Java OneWire API 中唯—— 一个不存在等价物的对象，因此，必需的文件可用下列程序产生：

```
ByteArray
 {
   // sets the item at the given index with the given value
   void setAt ( nIndex , nValue ) ;

   // returns the numerical value at the given index
   nValue getAt ( nIndex ) ;

   // grows or shrinks the size of the array , copies all data that fits
   void setSize ( nSize ) ;

   // returns the size of the array
   nSize getSize ( ) ;
}
```

3. 在 Java 中，程序过载允许程序参数的种类和数量不同。在 COM 中，只有参数的个数才能区分同名的两个程序。所有的程序都将与适合同样模式的需求相冲突。它们期望得到一个字符串，一个长整型数据，或者是一字节矩阵。当程序得到一个字符串的时候，可以不加任何处理，而当它得到一个长整型数据时就在它的末尾加上“FromLong”。当它得到一个字节矩阵的时候，就在矩阵的末尾加上“FromBytes”。例如：

Java		**COM**
isPresent（字符串）	—>	isPresent（字符串）
isPresent（长整型）	—>	isPresentFromLong（字符串）//按照经验 1 和经验 3
isPresent（字节矩阵）	—>	isPresentFromBytes（字节矩阵）//按照经验 2 和经验 3

4. OWAPI 中的一些软件包依赖于它们功能的多态性。从本质上讲，多态性是指一个类被看成是它的父类的一个真正的实例的能力。在 COM 中，多态性不能以同样的方式证明它自己，因为 COM 不支持继承。在我们的体系机构中，每个 COM 组件代表着一个 OWAPI 对象的实例，但通常是高层组件（如内存条）包含低层对象（如多页存储器或者 OTP 内存条）的实例。在 Java 中，可简单地把对象归结到它所属的特定类型中。遗憾的是，在 COM 中这并不容易，因此，在三个高层软件包：OneWireContainer，MemoryBank 和 DSPortAdapter 中加入了帮助程序。帮助程序的接口很简单：

   Component getMostSpecificComponent ( ) ;

   返回值依赖于所调用的是哪个类。比如：OneWireContainer 的一个实例拥有 Thermochron 对象（OneWireContainer21），调用 getMostSpecificComponent，将返回 OneWireContainer21 COM 接口的一个实例。
5. 发现一个特殊的 1-Wire 器件是否支持特殊的接口的机制是多态性所不具备的另一个特点。多行性也允许把类看作是类实现的接口的实例。例如，OneWireContainer21 实现 TemperatureContainer 接口。也就是说，OneWireContainer21 实现 TemperatureContainers 中的所有通用程序（如“readTemperature”）。在 Java 中，把 1-Wire 对象与使用关键字“instanceof”的特殊接口相比较是通用的。如果“myOneWireDevice instanceof TemperatureContainer”的返回值为“真”，则器件执行所有读取温度值所必需的程序。在 COM 中，这可以通过 OneWireContainers 所定义的下列程序来实现：

   boolean isTemperatureContainer ( ) ;
   boolean isADContainer ( ) ;
   boolean isSwitchContainer ( ) ;
   boolean isClockContainer ( ) ;
   boolean isPotentiometerContainer ( ) ;
   Boolean isHumidityContainer ( ) ;[还未实现]

6. Java 功能以前返回的是 java.util.Calendar 对象，而现在返回的是从格林尼治标准时间 1970 年 1 月 1 日以来到目前的毫秒数。这是 unix 系统表示时间的方法。
7. 接口不是存在于 Java 类包中，而是存在于同一级的包中。比如：OneWireContainer21 不存在于 com.dalsemi.onewire.container.OneWireContainer21 包中，而是简单地通过象 OneWireContainer21 一样的 COM 包进行访问。

下面的例 3 说明了 OWCOM 代码段，它同样遵循图 3 所概括的“API 用法流程”。简便起见，每次循环只找出一个 1-Wire 网上的器件。复杂点的应用程序一次循环中可以找出一类器件或者可以选定在上次循环中找到的器件。

OWCOM ‘Java Script’ 代码实例　例 3

```
boolean doing_work = true ;
// get the 1-Wire access provider
var access = Wscript.CreateObject ( “owapi.OneWireAccessProvider”) ;
// work loop
while (doing_work )
{
// get exclusive use of adapter (SESSION)
adapter.beginExclusive (true) ;

// clear any previous search restrictions (NETWORK)
adapter.setSearchAllDevices ( ) ;
adapter.targetAllFamilies ( ) ;
adapter.setSpeed (adapter.SPEED_REGULAR) ;

// enumerate through all the 1-Wire devices found (NETWORK)
for    (Enumeration owd_enum = adapter.getAllDeviceContainers ( ) ;
         owd_enum.hasMoreElements ( ) ; }
{
   // get a ‘container’ for each device
   owd = owd enum.nextElement ( ) ;

   // do SOMETHING with device found ( TRANSPORT/FILE/DEVICE)
   // . . .
}
// end exclusive use of adapter (SESSION)
adapter.endExclusive ( ) ;

// do other application work
// . . .
}
```

安装

有一批文件可以自动安装 OWCOM 驱动程序。它用 regsvr32 Windows 工具登记 COM 对象，然后把 OWAPI 类文件复制到一个“受托”文件夹。

TEMX API(TMEX)概述

TMEX 是一套独立于 Windows 32 位动态链接库的语言，它提供所有 1-Wire 器件的基本功能，包括用以支持存储器件的有限的 1-Wire 文件结构。这种 API 被设计为在多进程多线程应用程序争用同一或不同 1-Wire 端口时工作。它可以支持 16 种不同的 1-Wire 适配器，每种适配器又有 16 个截然不同的端口。它支持达拉斯半导体公司生产的所有 1-Wire 适配器。因此，这种 API 被用作 Win32 平台下 Java 1-Wire API 的“本地”驱动程序。1-Wire COM API 是以 Java 的 1-Wire API 为基础的，在这也同样。图 8 用图形说明了其它的 API 是如何利用 TMEX API 提供的 Win32 本地支持的。图中也包括实际驱动程序的文件名和它们是如何分层的。

带有 TMEX API（和 OWCOM）的软件开发工具包位于：
http://www.ibutton.com/software/tmex/index.html

TMEX SDK 所提供的所以例子都有源代码，但现在还未提供驱动程序的源代码。驱动程序可以自由地进行重新发布。

表 5 列出了当前支持的 1-Wire 适配器，同时还有每种适配器的特征和所支持的 Windows 平台。*注：对 Windows XP 的支持还在开发中。

TMEX 所支持的适配器　　表 5

适配器	**端口**	**特征**	Win95	Win98	WinME	WinNT	Win2K	WinXP
DS1490F 2-in-1 Fob	USB	供电，强驱动 LED 指示 单 iButton 座		X	X		X	X*
DS1490x (开发中)	USB	供电，强驱动 RJ-11 或 iButton 座，可选可写 EPROM		X	X		X	X
DS1410E	并口	供电，强驱动 双 iButton 座， DS2401 识别号	X	X	X	X	X	X
DS1410D	并口	双 iButton 座， DS2401 识别号	X	X	X	X	X	X
DS9097U-009	串口	供电，强驱动 RJ-11 连接器， DS2502 识别号	X	X	X	X	X	X
DS9097U-S09	串口	供电，强驱动 RJ-11 连接器	X	X	X	X	X	X
DS9097U-E25	串口	供电，强驱动 RJ-11 连接器 可写 EPROM	X	X	X	X	X	X
DS1411	串口	供电，强驱动 单 iButton 座	X	X	X	X	X	X
DS9097E	串口	RJ-11 连接器 可写 EPROM	X	X	X	X	X	X
DS9097	串口	RJ-11 连接器	X	X	X	X	X	X
DS1413	串口	单 iButton 座	X	X	X	X	X	X

TMEX API 驱动程序与其它 API 的连通性 图 8

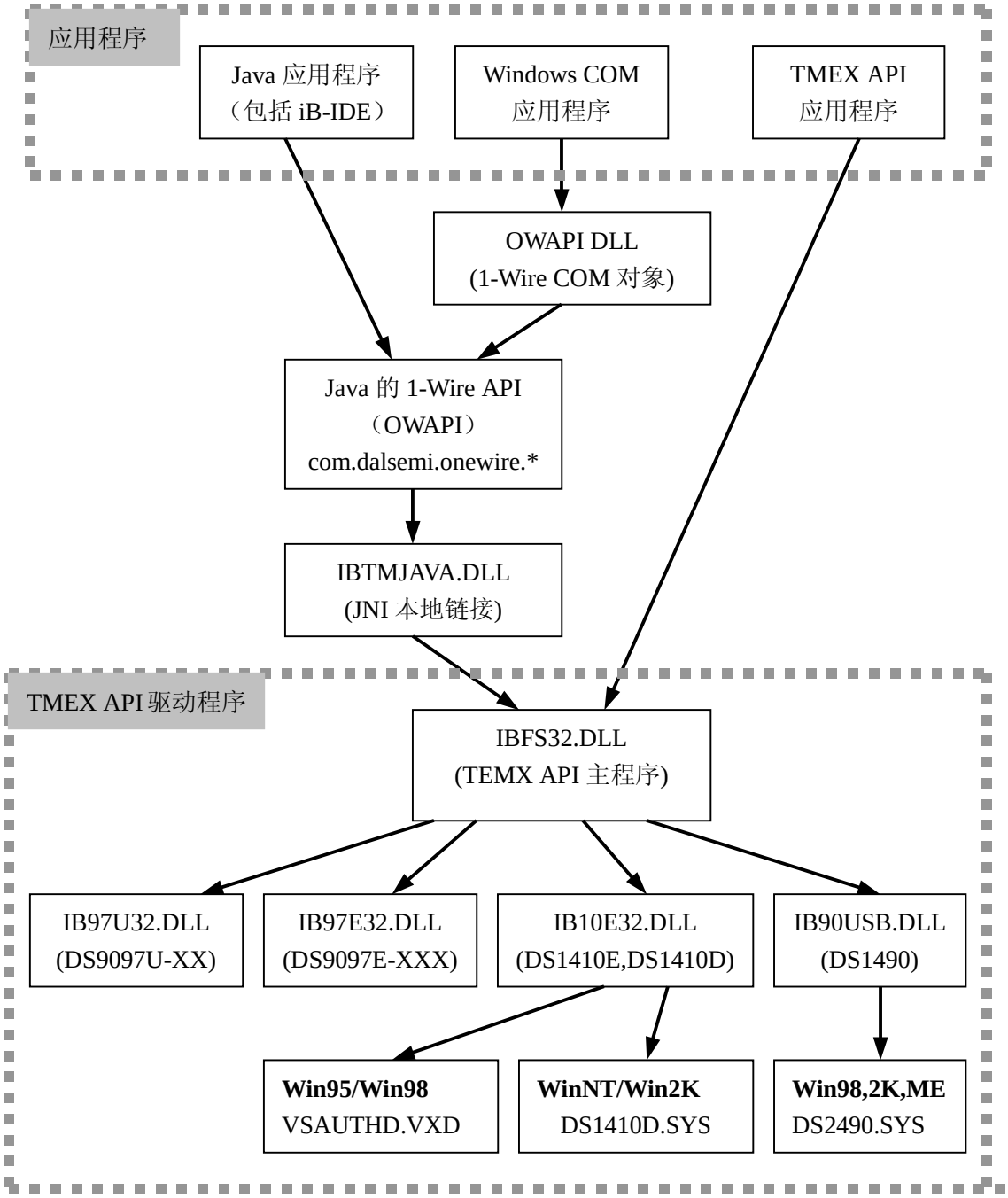


图 9 列出了 TMEX API 提供的功能。注：这种 API 不提供任何无存储器件的专用功能。

TMEX API 功能　图 9

会话
TMEndSession－放弃独占 1-Wire 网的使用权
TMExtendedStartSession－请求独占 1-Wire 网的使用权
TMValidSession－检测当前 1-Wire 网会话层是否有效
链接
TMClose－为开放端口释放资源（不总是可用）
TMOneWireCom－设置 1-Wire 网的速度为常态（16K 位）或快速传输（142K 位）
TMOneWireLevel－设置传输线上某点电平为常态（5V 电源加小阻值上拉电阻），供电传输（5V 电源加强上拉）或者编程电平（12V EPROM 编程电平）
TMProgramPulse－向 1-Wire 网发送同步编程脉冲进行 EPROM 的编程
TMSetup－检测和验证端口和适配器是否正在运行
TMTouchBit－发送并接收 1-Wire 网的 1 位数据
TMTouchByte－发送并接收 1-Wire 网的 1 个字节
TMTouchReset－复位 1-Wire 网上的所有数据并返回结果
网络
TMAccess－选择当前器件并准备执行一条专用器件指令
TMAutoOverDrive－设置驱动器自动地决定器件是否进行快速传输
TMFamilySearchSetup－设置当前的搜索状态为在下一次搜索（TMNext 或 TMNextAlarm）中找出指定的家族类型
TMFirst－找出 1-Wire 网上“第一个”1-Wire 器件
TMFirstAlarm－找出 1-Wire 网上“第一个”警报 1-Wire 器件
TMNext－找出 1-Wire 网上“下一个”1-Wire 器件
TMNextAlarm－找出 1-Wire 网上“下一个”警报 1-Wire 器件
TMOverAccess－选定当前器件并将其速度设为快速传输
TMRom－找回或设置当前器件的串口号（ROM 编号）
TMSkipFamily－跳过最后一次搜索发现的所有家族类型
TMStrongAccess－选择和验证当前器件是否存在
TMStrongAlarmAccess－选择和验证当前器件是否存在并且处于报警状态
传输
TMBlockIO－在 1-Wire 复位之前，发送和接收 1-Wire 网的一个数据块
TMBlockStream－发送和接收带有 NO 1-Wire 复位的 1-Wire 网的一个数据块
TMExtendedReadPage－读取内存条中含有器件产生的 CRC 校验码的完整的一页（并不适用于所有器件类型）
TMProgramByte－将一个字节编入 1-Wire 器件的 EPROM 基址
TMReadPacket－从存储器的页中读取通用数据包（见应用笔记 114：通用数据包结构描述）
TMWritePacket－向存储器的页中写入通用数据包
文件操作
TMAttribute－更改文件或目录属性
TMChangeDirectory－读取或更改当前的工作目录
TMCloseFile－关闭已打开或新建的文件以释放文件句柄
TMCreatFile－新建一个可写文件
TMCreateProgramJob－新建一个写缓冲器以记录 EPROM 未完成的编程工作

TMDeleteFile－删除一个文件 *TMDirectoryMR*－新建或移走一个子目录 *TMDoProgramJob*－写 EPROM 未完成的编程工作 *TMFirstFile*－找出当前目录下第一个文件 *TMFormat*－格式化 1-Wire 文件结构文件系统 *TMNextFile*－找出当前目录下的下一个文件 *TMOpenFile*－打开一个可读文件 *TMReadFile*－读已打开的文件 *TMReNameFile*－改变文件或目录的名称 *TMTerminateAddFile*－结束“AddFile”。“AddFile”是 EPROM 上的一种特殊的文件类型，不用重写就可以打开它 *TMWriteAddFile*－添加或改变 EPROM 器件上的“AddFile” *TMWriteFile*－写已创建的文件
器件
无
其它
TMGetTypeVersion－获得适配器驱动程序的版本信息 *Get_Version*－获得所有驱动程序的版本

下面的例 4 是 TMEX 的代码段，它遵循图 3 所概括的“API 用法流程”。简便起见，每次循环只找出一个 1-Wire 网上的器件。复杂点的应用程序一次循环中可以找出一类器件或者可以选定在上次循环中找到的器件。

TMEX 'C'代码实例 例 4

```
int PortNum , PortType ; // port number and type set for adapter present
long session_handle ; // session handle
unsigned char state_buffer [5120] ;
int doing_work = 1 , did_setup = 0 ;
short   rslt ;
// work loop
while ( doing work )
{
    // aquire the 1-Wire Net   ( SESSION)
  session_handle = TMExtendedStartSession ( PortNum , PortType ,NULL ) ;
  if   (session_handle > 0 )
  {   // check to see if TMSetup has been done once
      if   (!did_setup )
      {      if   (TMSetup (session_handle ) = = 1 )
             did_setup = 1 ;
        else
        {      // error setting up port , adapter may not be presents
               // .   .   .    }
        }
        else
        { // find all devices (NETWORK)
          rslt = TMFirst (session_handle , state_buf ) ;
          while ( rslt >0 )
          {   // do SOMETHING with device found (TRANSPORT/FILE/DEVICE)
              //   .   .   .
              // find the next device   (NETWORK)
              rslt = TMNext (session handle , state buf ) ;
            }
          }
          // release the 1-Wire Net (SESSION)
          TMEndSession (session_handle ) ;
        }
        else
        {     // Could not acquire 1-Wire network
            //   .   .   .
         }
         // do other application work
         //   .   .   .
    }
```

安装

TMEX API 能自动安装 Install-Shield 安装程序，载入所有支持 1-Wire 适配器的 Windows 驱动程序和注册表。它的安装有没有 iButton Viewer 都可以。同时也提供了客户安装所需的文件。iButton Viewer 是练习大多数 1-Wire 器件和 iButtons 的演示程序。

其它工具

本文所讨论的四种 API 代表了与 1-Wire 器件进行通信的相当一大部分可利用资源，但决不是唯一的资源。这一部分将介绍一些其它的可利用资源。

iB-IDE

iB-IDE 提供了完整的编程环境，用来适应 Java 激励的 iButton 软件的迅速发展。Java 主机的应用程序装在个人计算机上或者嵌在与 iButton 上的任意 applets 程序进行通信的系统上，而且随着 iB-IDE 的引入，这两者的设计都变得简单。IB-IDE 的一些主要特点是：

- 除具有 Java 激励的 iButton 模拟器的全部功能外，还有在 Java 源代码级进行调试的功能。
- 内置的文本编辑器，具有 Java 的主要的关键词，宏，查找和替换，以及 Java 特定语言格式化。
- 完整的 Java 编译和运行功能，控制与系统相连的 iButtons 并与之通信。

从版本 2.0Beta 开始，iB-IDE 就把 Java 的 1-Wire API 做为它的核心来与 Java 激励的 iButton 进行通信。

有关 iB-IDE 的信息以及它的下载见网站：http://www.ibutton.com/iB-IDE/。

Software Authorization

Software Authorization 是应用软件对所需硬件令牌的一种锁定。这里的令牌是一种连接到用户工作站的 iButton。在应用软件运行之前，也可能是在它运行时，查询 iButton 存在与否及其有效性。实际上，本文提到的任何 API 都能用于这种应用程序，但也有一些必须注意的问题。由于允许用户替换驱动程序，这些外部的驱动便产生了安全漏洞，从而使锁定失败。Software Authorization API 就是专门为这种应用程序开发的，它用可链接的模块代替了外部驱动。

关于 Software Authorization 软件包的信息可在下面的网站找到：http://www.ibutton.com/software/softauth/index.html。

1-Wire 软件开发利益集团

建议 1-Wire 开发者参加“1-Wire 软件开发者论坛”。该组织的目的就是探索，讨论和回答有关开发应用程序，工具和 1-Wire 产品系列的用法问题。该论坛由达拉斯半导体公司发起，其应用工程师将回答论坛提出的问题。关于 API 的升级报告和可下载的软件包也分发给论坛。有意者请到 http://lists.dalsemi.com/ 注册。

第三方

有大量的适用于 1-Wire 器件的第三方软件。其中有一些应用程序是购买的达拉斯半导体公司授权解决方案开发者（ASD）的“解决方案”。ASD 的全部名单和解决方案可在下面的网站找到：http://dbserv.maxim-ic.com/ibutton/solutions/search.cfm. ，在公共论坛也有公开的源程序，如“Source Forge” http://sourceforge.net/

结束语

本文对各种 1-Wire API 的特性做了概括性论述。详细介绍了四种不同的 API，和支持它们的平台及编程语言。速查参照表概括了每种器件类型的 API，使我们很容易选出所要的 API。手头有了正确的 API，使用 1-Wire 可以很容易设计应用程序。