Keywords: 1-Wire, OneWire, iButton, API, application program interface, software, examples, TMEX, java, OWAPI, public domain, PD, windows COM, OWCOM, OW.NET, .NET, dotnet
Related Parts

APPLICATION NOTE 155

1-Wire Software Resource Guide Device Description

Abstract: There are over 30 different 1-Wire® devices, including iButton® devices, that Maxim currently produces. Navigating the available APIs, software examples, and other resources to communicate with this array of devices or finding the correct resource for a single device type can be a daunting task. This document provides an overview of the available resources and a selection guide. The current 1-Wire devices are also presented in a convenient table, providing device descriptions and family code lookup.

Available APIs include TMEX (an API for Microsoft Windows®), 1-Wire Public Domain Kit (a cross-platform API), the 1-Wire API for Java™ (OWAPI) and its variant 1-Wire API for .NET (OW.NET), and 1-Wire API for .NET Compact (OW.NET.Compact). All of the APIs described in this document are free to use without restriction and, in most cases, include the complete source code.

Introduction

There are over 30 different 1-Wire devices, including iButton devices, that Maxim currently produces. Navigating the available application program interfaces (APIs), software examples, and other resources to communicate with this array of devices, or finding the correct resource for a single device type, can be a daunting task. This guide provides an overview of available resources and a selection guide. The APIs described in this document are free to use without restriction and, in most cases, include the complete source code.

1-Wire Overview

The 1-Wire bus is a simple signaling scheme that performs two-way communications between a single master and peripheral devices over a single connection. A powerful feature that all 1-Wire bus devices share is that each and every device, in a chip or an iButton, has a factory-programmed registration number that will never be repeated in any other device. In effect, every device is unique. This allows any single

device to be individually selected from among the many that can be connected to the same bus wire. Because one, two, or even dozens of 1-Wire devices can share a single wire for communications, a binary searching algorithm is used to find each device in turn. Once each device registration number is known, any device can be uniquely selected for communication using that registration number to address it.

The first part of any communication involves the bus master issuing a reset, which synchronizes the entire bus. A slave device is then selected for subsequent communications. This can be done by selecting all slaves, selecting a specific slave (using the registration number of the device), or by discovering the next slave on the bus using a binary search algorithm. These commands are referred to collectively as network function or read-only-memory (ROM) commands. Once a specific device has been selected, all other devices drop out and ignore subsequent communications until the next reset is issued.

Once a device is isolated for bus communication, the master can issue device-specific commands to it, send data to it, or read data from it. Because each device type performs different functions and serves a different purpose, each type has a unique protocol once it has been selected. Even though each device type may have different protocols and features, they all have the same selection process and follow the command flow seen in **Figure 1**.

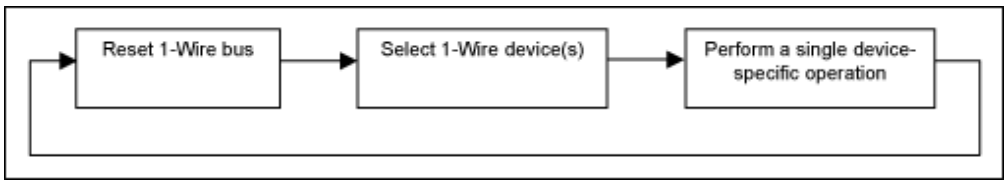

Figure 1. Typical 1-Wire communication flow.

An integral part of the unique registration number in each slave is an 8-bit family code. This code is specific to the device model. Because each device model performs different functions, this code is used to select the protocol that will be used to control or interrogate it. See **Table 1** for a mapping of family codes for Maxim 1-Wire parts.

Table 1. Family Code Reference

Family Code	Part () - iButton Package	Description (Memory size in bits unless specified)
01 (hex)	(DS1990A), (DS1990R), DS2401, DS2411	1-Wire net address (registration number) only
02	(DS1991)[1]	Multikey iButton, 1152-bit secure memory
04	(DS1994), DS2404	4Kb NV RAM memory and clock, timer, alarms
05	DS2405[1]	Single addressable switch
06	(DS1993)	4Kb NV RAM memory
08	(DS1992)	1Kb NV RAM memory
09	(DS1982), DS2502	1Kb EPROM memory
0A	(DS1995)	16Kb NV RAM memory
0B	(DS1985), DS2505	16Kb EPROM memory

0C	(DS1996)	64Kb NV RAM memory
0F	(DS1986), DS2506	64Kb EPROM memory
10	(DS1920)	Temperature with alarm trips
12	DS2406, DS2407[1]	1Kb EPROM memory, 2-channel addressable switch
14	(DS1971), DS2430A[1]	256-bit EEPROM memory and 64-bit OTP register
1A	(DS1963L)[1]	4Kb NV RAM memory with write cycle counters
1C	DS28E04-100	4096-bit EEPROM memory, 2-channel addressable switch
1D	DS2423[1]	4Kb NV RAM memory with external counters
1F	DS2409[1]	2-channel addressable coupler for sub-netting
20	DS2450	4-channel A/D converter (ADC)
21	(DS1921G), (DS1921H), (DS1921Z)	Thermochron® temperature logger
23	(DS1973), DS2433	4Kb EEPROM memory
24	(DS1904), DS2415	Real-time clock (RTC)
27	DS2417	RTC with interrupt
29	DS2408	8-channel addressable switch
2C	DS2890[1]	1-channel digital potentiometer
2D	(DS1972), DS2431	1024-bit, 1-Wire EEPROM
37	(DS1977)	Password-protected 32KB (bytes) EEPROM
3A	(DS2413)	2-channel addressable switch
41	(DS1922L), (DS1922T), (DS1923), DS2422	High-capacity Thermochron (temperature) and Hygrochron™ (humidity) loggers
42	DS28EA00	Programmable resolution digital thermometer with sequenced detection and PIO
43	DS28EC20	20Kb 1-Wire EEPROM

*This list may not include all Maxim 1-Wire device types (families), just the ones directly supported by the Automatic Information Business Unit's (BU's) software libraries.

[1]These devices are no longer recommended for new designs.

API Fundamentals

The different APIs for communicating with 1-Wire devices have common features that reflect the fundamental communication issues arising from the protocol. **Figure 2** outlines the common groupings of the functions for the different APIs. Since most 1-Wire devices have memory, the memory I/O functions are treated as a common API group although the functions do not apply to all devices. All other nonmemory specialty functions are lumped into the device-specific device grouping.

SESSION

Negotiate exclusive use of the 1-Wire bus. This is particularly important on operating systems or environments where several processes or threads could simultaneously attempt access to the same 1-Wire bus. Exclusive use of the network is required when doing multiple operations that must not be interrupted on a single device.

LINK

Primitive 1-Wire bus communication functions. All 1-Wire communication can be condensed down to the 'reset' that resets all devices and reading and writing bits. This group can also contain functions to set the electrical characteristics of the bus, such as when providing special EPROM programming pulses or power delivery.

NETWORK

Network functions for device discovery and selection. The unique registration number programmed into each 1-Wire device is used as its network address. These functions can be constructed from the LINK level functions. Data sheets for 1-Wire devices refer to these as ROM commands, as the registration number is Read-Only Memory. Some 1-Wire masters contain built-in network functions, because they are more efficient than link functions.

TRANSPORT

Block communication and primitive read/write memory functions. This can also include packet read/write memory functions. These functions are constructed from the NETWORK and LINK group functions.

FILE

File memory level functions using the 1-Wire File Structure (refer to application note 114, "1-Wire File Structure"). These functions are constructed from the NETWORK and TRANSPORT level functions, and are only useful for devices with more then one page of memory.

DEVICE

Device-specific 'high-level' functions. These functions are often constructed from the NETWORK, TRANSPORT, and LINK group functions and perform operations such as reading temperature values or setting a switch state.

Figure 2. API function groupings.

The typical sequence to use these functions is outlined in **Figure 3**. The SESSION functions wrap around the communication calls to the device, which typically involve using a NETWORK function followed by a memory or DEVICE-specific operation.

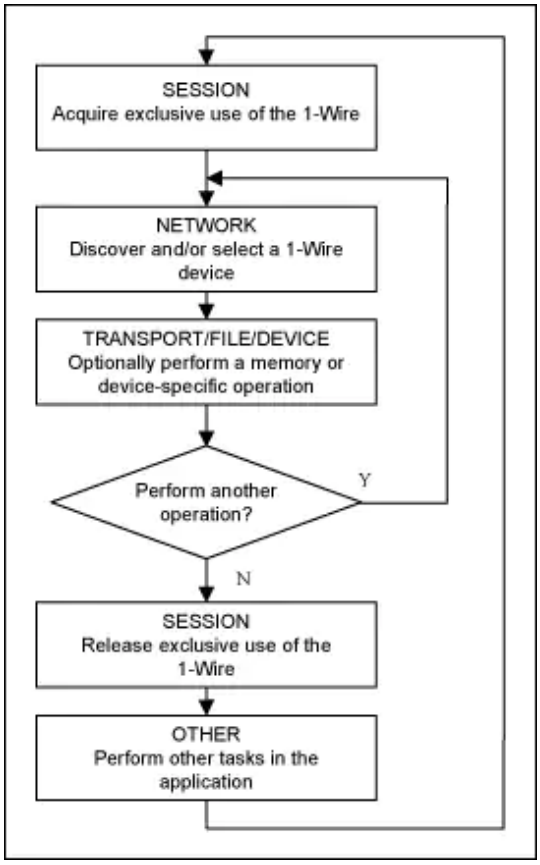

Figure 3. API usage flow.

The nature of iButton communication is inherently 'touch.' This means that contact with the device is not always reliable. The iButton might be inserted into the reader and have intermittent contact during the read. Consequently a consistent methodology of error recovery must be rigorously followed. This usually entails doing retries when a spurious error is detected and utilizing CRC checks in data communication. The file I/O functions in the APIs use a standard file structure detailed in the *1-Wire File Structure* section of application note 114, "1-Wire File Structure." This structure uses a CRC16 on every page of data to quickly verify the validity of the data being read. Most of the 1-Wire API functions have little or no automatic retries. The retries are under application control. See the application note 159, "Software Methods to Achieve Robust 1-Wire® Communication in iButton® Applications," for methodology of error recovery and risk assessment in doing 1-Wire communication.

API Selection

There are principally five different APIs that are considered in this document. The APIs operate on different platforms, use different languages, and have different capabilities. **Table 2** displays these five APIs with a brief description, and **Table 3** maps the operating system with the available APIs divided by language.

Table 2. API Descriptions

API	Abbreviation	Description

1-Wire Public Domain	PD	A completely open-source public domain API written in C and designed to be portable across multiple PC operating systems, handheld operating systems, and microcontroller platforms. For PC platforms, it supports all 1-Wire adapters (masters) through native driver libraries on Microsoft Windows and specific 1-Wire adapters (DS9097U serial and DS9490 USB adapters) on other PC operating systems using cross-platform libraries.
1-Wire API for Java	OWAPI	Completely open-source, high-level Java API that supports almost ALL 1-Wire devices. In addition to native 1-Wire master support, it also supports the DS9097U serial adapter and DS9490 USB adapter through cross-platform libraries.
1-Wire API for .NET	OW.NET	OWAPI code base compiled with J# for the Microsoft .NET Framework.
1-Wire API for .NET Compact	OW.NET.Compact	Compact .NET Framework for Windows CE machines or platforms that do not have the Microsoft Visual J#® Redistributable Package. It currently consists of just the low-level 1-Wire link and Network layer ported to C#.
TMEX API	TMEX	Supports all 1-Wire master adapters on Windows platforms (32 and 64 bit). Provides link and file I/O functions, but no device functions. Drivers are closed source. This API is called by other APIs to obtain access to all of the 1-Wire adapter types.

Table 3. API Operating System and Language Coverage

Language OS	TMEX /OW.NET/OW.NET.Compact (Microsoft Windows language independent)	C	Java
Windows Vista®	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows Vista x64	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows XP	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows XP x64	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows 2008	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows 2008 x64	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows 2000[1]	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows ME[1]	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows 98[1]	TMEX /OW.NET/**OW.NET.Compact**	PD	OWAPI
Windows 95[1]	TMEX	PD	OWAPI
Win3.1[1]	TMEX	PD	
DOS[1]	TMEX	PD	
Pocket PC/CE	OW.NET.Compact	PD	
Linux® and other UNIX®-based OSs		PD	OWAPI
MxTNI*		PD - without MxTNI OS	OWAPI

*MxTNI™ is an embedded platform with a Java-based OS made by Maxim.

[1]No longer supported. Legacy driver downloads still available from the Maxim web site.

The support of the individual device families also varies from API to API. **Table 4** lists all of the currently available 1-Wire devices with flags indicating the available support in each API. The key for the Table 4 flags is located at the bottom. Note that the device cells without shading are considered fully supported by the API. A light-shaded cell indicates partial support, and dark-shaded cell indicates minimal support.

Table 4. API Support by Device

Device	FC	Description	TMEX	PD	OWAPI	OW.NET	OW.NET Compact
DS1982	09	1Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS1985	0B	16Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS1986	0F	64Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS1904	24	RTC	AB	AB	ABI	ABI	AB
DS1920	10	Temperature and alarm trips	AB	ABI	ABCI	ABCI	AB
DS1921G DS1921H DS1921Z	21	Thermochron temperature logger	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS1922L DS1922T DS1923	41	High-capacity Thermochron (temperature) and/or Hygrochron (humidity) loggers	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS1963L[1]	1A	4Kb NV RAM memory with write-cycle counters	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1971	14	256-bit EEPROM memory and 64-bit OTP register	ABD	ABCDI	ABCDI	ABCDI	AB
DS1972	2D	1024-bit EEPROM memory	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	
DS1973	23	4Kb EEPROM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1977	37	Password-protected 32KB (bytes) EEPROM	AB	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1990A DS1990R	01	1-Wire address only	AB	AB	AB	AB	AB
DS1991[1]	02	Multikey iButton, 1152-bit secure memory	AB	ABC	ABC	ABC	AB
DS1992	08	1Kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1993	06	4Kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1994[1]	04	4Kb NV RAM memory and clock, timer, alarms	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS1995	0A	16Kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS1996	0C	64Kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	AB
DS2401	01	1-Wire address only	AB	AB	AB	AB	AB

DS2405[1]	05	Single switch	AB	ABI	ABI	ABI	AB
DS2404[1]	04	4Kb NV RAM memory and clock, timer, alarms	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS2406 DS2407[1]	12	1Kb EPROM memory, 2-channel addressable switch	ABCE	ABCDEI	ABCDEI	ABCDEI	AB
DS2408	29	8-channel addressable switch	AB	ABI	ABI	ABI	AB
DS2409[1]	1F	Dual switch, coupler	AB	ABI	ABI	ABI	AB
DS2411	01	1-Wire address only	AB	AB	AB	AB	AB
DS2413	3A	Dual-channel addressable switch	AB	ABI	ABI	ABI	AB
DS2415	24	RTC	AB	AB	ABI	ABI	AB
DS2417	27	RTC with interrupt	AB	AB	ABI	ABI	AB
DS2422	41	High-capacity Thermochron (temperature)/Hygrochron (humidity) logger	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS2423[1]	1D	4Kb NV RAM memory with external counters	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS2430A[1]	14	256-bit EEPROM memory and 64-bit OTP register	ABD	ABCDI	ABCDI	ABCDI	AB
DS2431	2D	1024-bit EEPROM memory	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS2450	20	Quad ADC	AB	ABI	ABI	ABI	AB
DS2502	09	1Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS2505	0B	16Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS2506	0F	64Kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	AB
DS2890[1]	2C	Single-channel digital potentiometer	AB	AB	ABI	ABI	AB
DS28E04-100	1C	4096-bit EEPROM memory, two-channel addressable switch	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	AB
DS28EA00	42	Programmable-resolution digital thermometer with sequence detect and PIO	AB	AB	AB	AB	AB
DS28EC20	43	20Kb EEPROM	AB	AB	AB	AB	AB

Support Shading Guide	Support Flags
Full Support	A. 1-Wire link primitive support B. 1-Wire network support C. Transport memory byte read/write support D. Transport memory packet read/write support
Partial Support	E. 1-Wire file structure type AA support (see application note 114, "1-Wire File Structure," for file structure types) F. 1-Wire file structure type AB support
Minimal Support	G. 1-Wire file structure type BA support H. 1-Wire file structure type BB support I. Other device-specific support

[1]These devices are no longer recommended for new designs.

1-Wire Public Domain (PD) Overview

The functions provided in 1-Wire PD API are completely written in 'C' and are intended to be used on platforms not supported by the TMEX API. The '1-Wire net' (or MicroLAN™) is a single wire and ground network with one master and one or more slave devices. This API creates a 1-Wire master that can be used to identify and communicate with slave devices. It provides all of the 1-Wire, transport, and file level services to communicate with all of the 1-Wire devices, including iButtons. This API kit and sample platform builds are available on the iButton web site.

The 'C' source code to this API is designed to be portable. There are provided 'TODO' templates to be completed for a specific platform. Several platform example implementations are provided including: Windows 64-bit, Windows 32-bit, and Linux. There are also several example applications that use these platform implementations.

There are three sets of portable source files defined as 'general,' 'userial,' and 'other.' The first set is general purpose and is intended for platforms that already have the primitive link 1-Wire communication functions (general). This is the lowest level that is hardware dependent. The second set of portable source files assumes that the user has a serial port (RS-232) and wishes to utilize the 'Universal Serial 1-Wire Line Driver Master: DS2480B' (userial). This chip receives commands over the serial port, performs 1-Wire operations, and then sends the results back to the serial port. The source code converts the intended 1-Wire operations into serial communications packets to the DS2480B. The only module that need be provided for a platform are the serial port read/write primitives. The DS2480B is the interface chip used in all of the DS9097U series serial adapters. Finally, the third set of portable source files deal with specific 1-Wire adapter functionality and/or does not quite fall into the previous categories (other). An example of this deals with the USB port, specifically utilizing the DS2490 USB to 1-Wire bridge chip. In many ways, this is similar to the general build, but has been modified for DS2490 specificity. No matter which file set is used (general, userial, or other), in the end, the same API is actually published to the software developer.

In particular, this application note also presents the open-source, cross-platform libusb build of the 1-Wire Public Domain Kit (classified as an 'other' build). Libusb is an open-source USB library that has been ported across many different OSs. This particular build(s) of the 1-Wire PD API compiles in the necessary libusb functions that are used to then construct the 1-Wire PD API.

See table below for the various file sets available, which have already been built for each platform presented in the table and are available for download.

Table 5. 1-Wire File Sets and Prebuilt Binaries

Portable Source File Set (Builds)	Platform	Port	Description
userial	Win64, Win32, Linux, (other UNIX), DS550	COM	Supports the DS9097U 1-Wire serial port adapters and other DS2480B-based solutions (included embedded).

general	Win64, Win32	LPT	The LPT build supports the DS1410E[1] parallel port adapter on Windows.
	DS550	Microprocessor (µP) port pin	The DS550 general build uses a µP port pin.
other 'libusb'	Win64, Win32, Linux, Macintosh, (other UNIX)	USB	Supports DS9490 USB 1-Wire adapters and any other DS2490-based USB solution specifically with the appropriate libusb drivers installed.
other 'WinUsb'	Win64, Win32	USB	Supports DS9490 USB 1-Wire adapters and any other DS2490-based USB adapter under Windows specifically with the WinUSB device driver.
other 'wrapper' (TMEX)	Win32	USB, COM, LPT	Wraps the TMEX API, which gives the ability for multiport support under Windows (DS9490, DS9097U, DS1410E[1], respectively).
other 'multiport'	Win64, Win32	USB, COM, LPT	Gives mutliport support by communicating directly with low-level native Windows drivers.

Note: Refer to the 1-Wire Public Domain Kit web site for latest platform builds.
[1]The DS1410E parallel port adapter is not recommended for new designs.

These sets of portable source code files implement the same 1-Wire API functions and are interchangeable. **Figure 4** shows the available API for version 3.xx of the 1-Wire PD code base. Note that the nonmemory-device-specific functions are not listed in detail due to their large number. Figure 4 maps the source files that provide the functions and the required modules for new platforms.

In addition to the portable 'C' modules in the PD kit download, there are also a limited number of microprocessor (µP) assembly examples to perform 1-Wire communication.

The file functions in this API implement the 1-Wire File Structure type 'AA' defined by application note 114, "1-Wire File Structure."

As the name of this API would imply, the source code provided has a license that is as close to being public domain as possible. Developers are free to use and integrate this code into their applications without restriction.

SESSION

owAcquire - Acquires the 1-Wire net.
owRelease - Releases the previously acquired 1-Wire net.

LINK

owHasOverDrive - Indicates whether the adapter has overdrive capability.
owHasPowerDelivery - Indicates whether the adapter can deliver power.
owHasProgramPulse - Indicates whether or not EPROM programming voltage is available.
owLevel - Sets the 1-Wire net line level to Normal (5V weak pullup), Power Delivery (5V strong pullup), or Program Level (12V EPROM programming level).
owProgramPulse - Sends timed programming pulse for EPROM 1-Wire device writing.
owReadBitPower - Reads 1 bit and then optionally supplies power.
owReadByte - Receives 8 bits from the 1-Wire net by sending all 1's (0xFF).
owSpeed - Sets the speed of the 1-Wire net to Normal (16kb) or Overdrive (142kb).
owTouchBit - Sends and receives 1 bit from the 1-Wire net.
owTouchByte - Sends and receives 8 bits from the 1-Wire net.
owTouchReset - Resets all devices on the 1-Wire net and returns result.
owWriteByte - Sends 8 bits to the 1-Wire net and verifies the echo received matches.
owWriteBytePower - Sends 8 bits of communication to the 1-Wire net and then supplies power.

NETWORK

owAccess - Selects the current device and readies it for a device-specific command.
owFamilySearchSetup - Sets up the following search (*owNext*) to find a specific family type.
owFirst - Searches to find the first 1-Wire device on the 1-Wire net.
owNext - Searches to find the next 1-Wire device on the 1-Wire net.
owOverdriveAccess - Selects the current device and sets the speed to Overdrive.
owSerialNum - Retrieves or sets the currently selected device registration number (ROM number).
owSkipFamily - Skips all of the 1-Wire devices with the family type that was found in the last search.
owVerify - Selects and verifies that the current device is present (alarming option).

TRANSPORT

owBlock - Sends and receives a block of data to the 1-Wire net with optional reset.
owCanLockPage - Checks to see if the given memory bank has pages that can be locked.
owCanLockRedirectPage - Checks to see if the given memory bank has pages that can be locked from being redirected.
owGetAlternateName - Gets the alternate part numbers or names.
owGetBankDescription - Gets a string description of the memory bank.
owGetDescription - Gets a short description of the 1-Wire device type.
owGetExtraInfoDesc - Gets a description of what is contained in the extra information.
owGetExtraInfoLength - Gets the length in bytes of extra information in this memory bank.
owGetMaxPacketDataLength - Gets maximum data length in bytes for a packet.
owGetName - Gets the part number of the 1-Wire device as a string.
owGetNumberBanks - Gets the number of memory banks for a certain 1-Wire family group.
owGetNumberPages - Gets the number of pages in a given memory bank.
owGetPageLength - Gets the raw page length in bytes for a given memory bank.
owGetSize - Gets the size of a given memory bank in bytes.
owGetStartingAddress - Gets the physical starting address of the given memory bank.
owHasExtraInfo - Checks to see if this memory bank's pages deliver extra information when read.
owHasPageAutoCRC - Checks to see if the memory bank has device-generated CRC verification when reading a page.
owIsGeneralPurposeMemory - Checks to see if the memory bank is general-purpose user memory.
owIsNonvolatile - Checks to see if current memory bank is nonvolatile.
owIsReadOnly - Checks to see if the memory bank is read-only.
owIsReadWrite - Checks to see if the memory bank is read/write.
owIsWriteOnce - Checks to see if the memory bank is write once, such as with EPROM.
owNeedsPowerDelivery - Checks to see if this memory bank requires 'PowerDelivery' to write.
owNeedsProgramPulse - Checks to see if this memory bank requires a 'ProgramPulse' to write.
owRead - Reads a portion of a memory bank in raw mode (no packets, CRC).
owReadPage - Reads an entire page of a memory bank in raw mode (no packets, CRC).
owReadPageCRC - Reads an entire page of a memory bank with device-generated CRC verification.
owReadPageExtra - Reads an entire raw page of a memory bank including any 'extra' information. (no packets, CRC)
owReadPageExtraCRC - Reads an entire raw page of a memory bank including any 'extra' information and with device-generated CRC verification.
owReadPagePacket - Reads a Universal Data Packet from a page in a memory bank. (Refer to application note 114, "1-Wire File Structure," for Universal Data Packet structure description.)
owReadPagePacketExtra - Reads a Universal Data Packet from a page in a memory bank with 'extra' information.
owRedirectPage - Checks to see if the memory bank has pages that can be redirected.
owWrite - Writes a portion of a memory bank in raw mode.
owWritePagePacket - Writes a Universal Data Packet to a page in a memory bank.

FILE

owAttribute - Changes the attributes of a file.
owChangeDirectory - Changes the current directory.
owCloseFile - Closes a file.
owCreateDir - Creates a directory.
owCreateFile - Creates a file for writing.
owCreateProgramJob - Creates a write buffer for logging EPROM programming pending jobs.
owDeleteFile - Deletes a file.
owDoProgramJob - Write the pending EPROM programming jobs.
owFirstFile - Finds the first file in the current directory.
owFormat - Formats the 1-Wire File Structure file system.
owGetCurrentDir - Gets the current directory.
owNextFile - Finds the next file in the current directory.
owOpenFile - Opens a file for reading.
owReadFile - Reads an opened file.
owReadFile - Reads data from a file.
owRemoveDir - Removes a directory.
owReNameFile - Changes the name of a file.
owWriteFile - Writes to a file that has been created.

DEVICE

DoAtoDConversion - Does an A/D conversion on DS2450.
ReadSwitch12 - Reads the state of the DS2406 switch.
readCounter - Reads the counter value associated with a specific memory page on a DS2423 1-Wire chip.
...(too numerous to list all device-specific functions)

Figure 4. PD API functions.

Example 1 shows a PD code fragment that follows the API usage flow outlined in Figure 3. For simplicity, each device on the 1-Wire network is discovered during each pass through the work loop. A more sophisticated application could potentially find just one device type or perhaps select a device found in a previous search.

```
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
  // . . .
}
```

Example 1. PD CODE example.

Figures 5a and **5b** list the C-language modules that make up each of the two sets of 1-Wire PD libraries. Also displayed are the 'TODO' functions that must be provided to port the library to a new platform. Several example platform link files that implement the 'TODO' functions are provided in the kit.

SESSION

owsesu.c

LINK

owllu.c ds2480ut.c ds2480.h

NETWORK

ownetu.c crcutil.c (required to compile)

TRANSPORT

mbappreg.c mbappreg.h mbee.c mbee.h mbee77.c mbee77.h

mbeewp.c	mbeewp.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	mbscrx77.c	mbscrx77.h
mbsha.c	mbsha.h	mbshaee.c	mbshaee.h	owtrnu.c	pw77.c
pw77.h	rawmem.c	rawmem.h			

FILE

owcache.c	owfile.c	owfile.h	owpgrw.c	owprgm.c	

DEVICE

ad26.c	ad26.h	atod20.c	atod26.c	atod26.h	cnt1d.c
humutil.c	humutil.h	jib96.c	jib96.h	jib96o.c	ps02.c
ps02.h	sha18.c	sha33.c	shadbtvm.c	shadebit.c	shaib.c
shaib.h	swt05.c	swt12.c	swt12.h	swt1c.c	swt1c.h
swt1f.c	swt29.c	swt29.h	swt3a.c	swt3a.h	temp10.c
thermo21.c	hermo21.h	time04.c	time04.h	weather.c	weather.h

MISC UTILITY

ioutil.c	owerr.c	findtype.c	ownet.h	screenio.c	sprintf.c
crcutil.c					

TODO

Provided a SERIAL interface module that implements the following functions:
*BreakCOM** - Sends a 'BREAK' on the serial port for at least 2ms.
CloseCOM - Closes the previously opened serial port. (optional for some platforms)
*FlushCOM** - Allows any pending write operations to complete and clear input buffer.
*msDelay** - Delays at least the specified number of milliseconds.
msGettick - Returns an increment millisecond counter. (optional for some examples)
OpenCOM - Opens the specified serial port for communication. (optional for some platforms)
*ReadCOM** - Reads a specified number of bytes from the serial port.
SetCOMBaud - Changes the serial BAUD rate to the rate specified. (optional if needs overdrive)
*WriteCOM** - Writes a specified number of bytes to the serial port.
* Minimum functions required for basic operation.

Figure 5a. PD 'USERIAL' implementation.

SESSION

(see TODO)

LINK

(see TODO)

NETWORK

ownet.c crcutil.c (required to compile)

TRANSPORT

Same as USERIAL implementation except 'owtrnu.c' is replaced by 'owtran.c'.

FILE

Same as USERIAL implementation.

DEVICE

Same as USERIAL implementation.

MISC UTILITY

Same as USERIAL implementation.

TODO

Provided a LINK and SESSION interface module that implements the following functions:
owAcquire - Acquires the 1-Wire net.
owRelease - Releases the previously acquired 1-Wire net.
owHasOverDrive - Indicates whether the adapter has overdrive capability.
owHasPowerDelivery - Indicates whether the adapter can deliver power.
owHasProgramPulse - Indicates whether or not EPROM programming voltage is available.
owLevel - Sets the 1-Wire net line level to Normal (5V weak pullup), Power Delivery (5V strong pullup), or Program Level (12V EPROM programming level).
owProgramPulse - Sends timed programming pulse for EPROM 1-Wire device writing.
owReadBitPower - Reads 1 bit and then optionally supplies power.
owReadByte - Receives 8 bits from the 1-Wire net by sending all 1's (0xFF).
owSpeed - Sets the speed of the 1-Wire net to Normal (16kb) or Overdrive (142kb).
*owTouchBit** - Sends and receives 1 bit from the 1-Wire net.
owTouchByte - Sends and receives 8 bits from the 1-Wire net.
*owTouchReset** - Resets all devices on the 1-Wire net and return result.
owWriteByte - Sends 8 bits to the 1-Wire net and verifies the echo received matches.
owWriteBytePower - Sends 8 bits of communication to the 1-Wire net and then supplies power.
* Minimum functions required for basic operation.

Figure 5b. PD 'GENERAL' implementation.

Installation

The 1-Wire PD API is a set of C modules, so there is no formal installation. As provided in the example builds, the required modules are compiled directly into the applications. This does not preclude developers from combining the modules into a loadable library such as a Windows DLL.

Keep in mind that some builds require native 1-Wire adapter drivers or equivalent. As most OS platforms have built-in serial port drivers, the userial builds of the 1-Wire PD API do not need any other driver. However, for USB and parallel port 1-Wire adapter support, native or cross-platform drivers will need to be installed. References to the appropriate driver downloads are available in the documentation of the actual 1-Wire PD Kit builds available online.

1-Wire API for Java (OWAPI) Overview

The 1-Wire API for Java was designed from the ground up to be a very robust, highly object-oriented foundation for building 1-Wire applications in Java. It extends the ability of programmers to develop portable, cross-platform software and shortens the time to market for their 1-Wire integrated products.

The API consists of many Java classes and interfaces. One special group of Java classes in the 1-Wire API is the container (class OneWireContainer). Support for particular 1-Wire devices, including iButtons, is provided through containers. The API has over 30 different container types, representing most 1-Wire devices. Each container encapsulates and implements the functionality of an individual device.

A container interacts with a 1-Wire device through a 1-Wire adapter class that represents a physical 1-Wire adapter (class DSPortAdapter). The instance of the adapter is produced from the provider class (class OneWireAccessProvider). The actual implementations of the 1-Wire adapters vary from platform to platform, but they all have the same interface. Some platforms use native drivers, but most at least support the DS9097U-XXX serial adapters using RXTX (a cross-platform serial COM port API). This API is available from RXTX's web site.

The 1-Wire API for Java Software Development Kit is available on the iButton web site. Like the 1-Wire PD Kit, the complete Java source to OWAPI is provided under a public-domain-style license.

Figure 6 shows the typical object creation sequence for this API. The 'provider' creates an instance (or enumeration) of an 'adapter,' which in turn can create instances of device 'containers.' Communication to the device is then performed almost exclusively through the container.

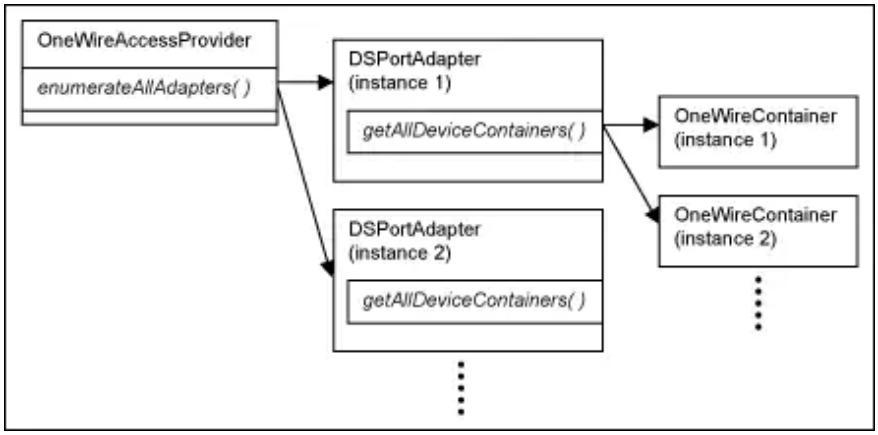

Figure 6. OWAPI object creation.

Figure 7 shows the common features of a container. A device that contains memory will create a memory bank instance for each memory bank. The memory is divided up into banks depending on the feature set of the bank. For example, one bank could be volatile while another is nonvolatile. Or, a bank could be general-purpose memory or it could be memory-mapped to change the functionality of the device.

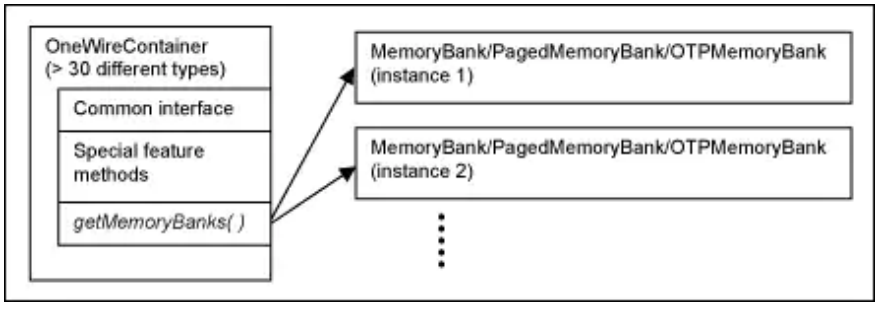

Figure 7. OWAPI ONEWIRECONTAINER features.

Figure 8 shows the methods provided by the base OWAPI classes. The class or package is displayed in

bold. Note that, because each container has high level methods to manipulate each device type, the LINK level methods in the adapter are not usually called directly.

SESSION

com.dalsemi.onewire.adapter.DSPortAdapter
beginExclusive - Acquires exclusive use of the 1-Wire net.
endExclusive - Releases the exclusive lock on the 1-Wire net.

LINK

com.dalsemi.onewire.adapter.DSPortAdapter
canBreak - Checks if a 1-Wire 'break' (long low) operation is supported by the adapter.
canDeliverPower - Checks if 'strong-pullup' power delivery is supported by the adapter.
canDeliverSmartPower - Checks if 'smart' power delivery is supported by the adapter. 'Smart' power delivery is the ability to sense when power consumption has reduced and automatically stop the power delivery.
canFlex - Checks if flexible long-line communication timing is supported by the adapter.
canHyperdrive - Checks if hyperdrive communication speed is supported by the adapter.
canOverdrive - Checks if overdrive communication speed is supported by the adapter.
canProgram - Checks if 12V EPROM programming voltage is supported by the adapter.
dataBlock - Sends and receives a block of data to the 1-Wire net.
getBit - Reads a single bit from the 1-Wire net.
getBlock - Reads a block from the 1-Wire net by sending all (0xFF)s.
getByte - Reads a byte from the 1-Wire net by sending all 1's (0xFF).
getSpeed - Reads the current 1-Wire communication speed.
putBit - Writes a bit to the 1-Wire net.
putByte - Writes a byte to the 1-Wire net and verifies the echo is correct.
reset - Resets all of the 1-Wire net devices.
setPowerDuration - Sets the power delivery duration.
setPowerNormal - Turns off the power delivery.
setProgramPulseDuration - Sets the program pulse duration.
setSpeed - Sets the 1-Wire communication speed.
startBreak - Starts a break (low) on the 1-Wire net.
startPowerDelivery - Starts the power delivery.
startProgramPulse - Starts the program pulse.

NETWORK

com.dalsemi.onewire.adapter.DSPortAdapter
excludeFamily - Excludes a family group from the search.
findFirstDevice - Finds the first device on the 1-Wire net without auto container creation.
findNextDevice - Finds the next device on the 1-Wire net without auto container creation.
getAllDeviceContainers - Searches and finds all devices on the 1-Wire net with containers.
getDeviceContainer - Gets a device container for the 'current' device found.
getFirstDeviceContainer - Finds the first device and create a container for it.
getNextDeviceContainer - Finds the next device and create a container for it.
setNoResetSearch - Sets the 1-Wire net search to not issue a 1-Wire reset.
setSearchAllDevices - Sets the 1-Wire net search to include all devices (remove alarm-only).
setSearchOnlyAlarmingDevices - Sets the 1-Wire net search to only include alarming devices.
targetAllFamilies - Sets the 1-Wire net search to include all devices (remove exclusions).
targetFamily - Targets a particular family group in the 1-Wire net search.
(also in **com.dalsemi.onewire.container**.*)
isAlarming - Checks to see if the device is in an alarm state.
isPresent - Checks to see if the device is present on the 1-Wire net.
select - Selects the 1-Wire net device to ready it for a device-specific operation command.

TRANSPORT

com.dalsemi.onewire.container.MemoryBank
getBankDescription - Returns a text description of the memory bank.
getSize - Gets the size of the memory bank in bytes.
getStartPhysicalAddress - Gets the starting physical address of the memory bank.
isGeneralPurposeMemory - Checks if the memory bank is general purpose (not memory mapped).
isNonVolatile - Checks if the memory bank is nonvolatile.
isReadOnly - Checks if the memory bank is read-only.
isReadWrite - Checks if the memory bank is read and write capable.
isWriteOnce - Checks if the memory bank is write once, such as EPROM.
needsPowerDelivery - Checks if this memory bank requires power delivery to write.
needsProgramPulse - Checks if this memory bank requires program pulse to write.
read - Reads the memory bank without interpretation (no packet structure).
setWriteVerification - Sets the API to do an extra verification after writes.
write - Writes the memory bank raw (no packet structure).
com.dalsemi.onewire.container.PagedMemoryBank
getExtraInfoDescription - Gets a description of extra information associated with this bank.
getExtraInfoLength - Gets the length in bytes of the extra information in each page.
getMaxPacketDataLength - Gets the maximum length of data that can be contained in the 'packet' structure that will fit in each page of this memory bank.
getNumberPages - Gets the number of pages in this memory bank.
getPageLength - Gets the length in byte of the raw page in this memory bank.
hasExtraInfo - Checks if this memory bank has extra information associated with each page.
hasPageAutoCRC - Checks if the pages in this memory bank have CRC verification that is supplied by the device.
readPage - Reads a page from the memory bank.
readPageCRC - Reads a page from the memory bank utilizing the device-generated CRC.
readPagePacket - Reads a packet structure from a page in the memory bank.
writePagePacket - Writes a packet structure to a page in the memory bank.
com.dalsemi.onewire.container.OTPMemoryBank
canLockPage - Checks if the pages in the memory bank can be locked from further writes.
canLockRedirectPage - Checks to see if the redirection facilities in the memory bank can be locked to prevent further redirection.
canRedirectPage - Checks to see if the memory bank can have pages redirected as a way to update write-once pages.
getRedirectedPage - Gets the page number that a page is redirected to.
isPageLocked - Checks if the page is locked from further writes.
isRedirectPageLocked - Checks if the page is locked from further redirection.
lockPage - Locks a page.
lockRedirectPage - Locks the page from being redirected.
redirectPage - Redirects a page to a new page. This is used to update a write-once device.

FILE

com.dalsemi.onewire.utils.OWFile
Same methods in java.io.File (for version 1.2 of the JDK) plus the following extra methods:
close - Closes the file and releases any resources associated with it.
format - Formats the 1-Wire File System associated with the device(s) provided to this OWFile.
getFD - Gets a OWFileDescriptor for this file so the file can be synchronized with the device.
getFreeMemory - Gets the available free memory in the 1-Wire File System.
getLocalPage - Gets a memory bank local page reference from the 1-Wire File System page.
getMemoryBankForPage - Gets the memory bank instance that can be used to read/write the provided 1-Wire File System page.
getOneWireContainer - Gets the container(s) that make up the file system.
getPageList - Gets a list of 1-Wire File System pages that comprise the file.
com.dalsemi.onewire.utils.OWFileDescriptor
Same methods as in java.io.FileDescriptor (for version 1.2 of the JDK).
com.dalsemi.onewire.utils.OWFileOutputStream
Same methods as in java.io.FileOutputStream (for version 1.2 of the JDK).
com.dalsemi.onewire.utils.OWFileInputStream
Same methods as in java.io.FileInputStream (for version 1.2 of the JDK).

DEVICE

com.dalsemi.onewire.container.*
Over 30 different device-specific container implementations including six different sensor-type interfaces:
ADContainer - Analog-to-digital converter
ClockContainer - Clock
SwitchContainer - Switch
TemperatureContainer - Temperature sensor
PotentiometerContainer - Digital potentiometer
HumidityContainer - Humidity sensor
MissionContainer - For missioning temperature and humidity loggers
OneWireSensor - 1-Wire sensors
PasswordContainer - Password-protected memory
com.dalsemi.onewire.application.*
SHA and 1-Wire tagging utility classes.
*Minimum functions required for basic operation.

Figure 8. OWAPI functions.

Example 2 shows a OWAPI code fragment that follows the API usage flow outlined in Figure 3. Like the PD code example in Example 1, each device on the 1-Wire network is discovered during each pass through the work loop, while a more sophisticated application could find just one device type or a previously found device.

```
boolean doing_work=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
  // . . .
}
```

Example 2. OWAPI code example.

1-Wire Tagging

As 1-Wire sensors get more numerous and diverse, it becomes increasingly difficult to manage a 1-Wire network. For example, a sensor like an ADC could measure various different values, so it becomes important to be able to tag the sensor to describe its function. A 1-Wire tagging scheme using XML has been created and implemented in the 1-Wire API for Java. These tags allow an application to dynamically load and configure a sensor to give it a context. Please refer to application note 158, "1-Wire Tagging with XML," for details.

Installation

All of the required modules that make up the API calls described in Figure 8 are contained in a single jar file: OneWireAPI.jar. Placing this one module in the correct location or classpath provides the entire API. There are two noted exceptions to this: there could be native or communication APIs required to be installed for a particular platform, or the platform could have size constraints so that it is undesirable to have the entire API available. These two exceptions are examined in detail in the OWAPI kit.

1-Wire .NET (OW.NET) Overview

For all intents and purposes, the .NET support we offer is the 1-Wire API for Java, only compiled with the Microsoft J# language. All .NET 1-Wire applications only need to reference the OneWire.NET.dll. This includes support for the latest .NET languages, such as C#, J#, and VB.NET. For 1-Wire .NET examples, please refer to the appropriate software development kit (SDK) download on the iButton web site.

Keep in mind that the OneWire.NET.dll also needs the following redistributables installed on a PC before the 1-Wire applications will work properly:

1. Native 1-Wire port adapter device drivers. These are known as 1-Wire Drivers and can be downloaded from the iButton web site.
2. Microsoft .NET 2.0 Framework
3. Visual J# .NET 2.0 Redistributable

For those desiring Compact .NET Framework 1-Wire support for devices such as Pocket PCs, personal digital assistants (PDAs), mobile phones, and set-top boxes, a new link-layer (only) version of the OneWire.NET.dll is available. It was written entirely in C#. Currently, no OneWireContainers have been written for it, but the DSPortAdapter classes are available. See the 1-Wire SDK for Windows for further details.

The OW.NET API is provided in a 1-Wire drivers package that also includes the TMEX API. For details on how to create a custom install of the 1-Wire Drivers, please see the following application note 1740, "White Paper 6: 1-Wire Drivers Installation Guide for Windows."

1-Wire .NET Compact (OW.NET Compact) Overview

The interface for this object is identical to the 1-Wire .NET (OW.NET) however only the Session, Link, and Network layers are implemented. The object does not have a formal distribution install, but is instead a single DLL (OneWireLinkLayer.dll). This object communicates directly with the TMEX API on both 32- and 64-bit Windows platforms. The DLL file can be included with a 2.0 .NET application for low-level 1-Wire support. This object was written in C#, so it does not require the J# 2.0 .NET redistributable package.

TMEX API (TMEX) Overview

The TMEX API is a set of language-independent Windows DLLs that provides basic functionality to all 1-Wire devices including limited 1-Wire File Structure support to memory devices. The API is designed to work in multiprocess, multithreaded applications all vying for the same or different 1-Wire ports. The API can

support up to 16 different types of 1-Wire adapters each with 16 distinct ports. It supports all 1-Wire adapters created by Maxim.

The lowest level of this API (the lowest level device-driver layer) is used as the native drivers for the 1-Wire API for Java on Windows platforms (1-Wire Drivers). Since the 1-Wire .NET API is also based on the 1-Wire API for Java, it is also used there. **Figure 9** shows graphically how the other APIs take advantage of the native support that the TMEX API provides. Included in this figure are the actual driver file names and how they are layered.

Both available on the Maxim web site are the 1-Wire Drivers installation package (containing the TMEX API and OW.NET libraries) and the Software Developer's Kit containing many example programs linked to TMEX (and accompanying APIs).

All of the examples provided in the TMEX SDK are provided with source code. However, the source code for the low-level drivers is currently not provided, though the drivers can be redistributed without restriction.

Table 6 lists the currently supported 1-Wire adapters along with the features of each adapter.

Table 6. TMEX Adapters Supported

Adapter	Port	Features
DS9490R, DS9490B	USB	Power delivery overdrive RJ-11 or iButton holder DS2401 ID
DS1410E[1]	Parallel	Power delivery overdrive dual iButton holder DS2401 ID
DS1410D[1]	Parallel legacy	Dual iButton holder DS2401 ID
DS9097U-009	Serial	Power delivery overdrive RJ-11 connector DS2502 ID
DS9097U-S09	Serial	Power delivery overdrive RJ-11 connector
DS9097U-E25	Serial	Power delivery overdrive RJ-11 connector EPROM write
DS1411	Serial	Power delivery overdrive single iButton holder
DS9097E[1]	Serial legacy	RJ-11 connector EPROM write
DS9097[1]	Serial legacy	RJ-11 connector
DS1413[1]	Serial legacy	Single iButton holder

[1]These adapters are no longer recommended for new designs.

Windows Platforms Supported by TMEX

The following Microsoft Windows platforms are supported by TMEX: Windows 2008, Windows 2003, Windows Vista, and Windows XP SP2. This includes both x86 (32-bit) and x64 (64-bit) operating system versions. Please note that if drivers for earlier Windows operating system versions are required, legacy versions of TMEX are available, though not actively supported, from Maxim's web site. Please download version 4.00 or below of the 1-Wire Drivers (which install the TMEX API libraries). 1-Wire Drivers installation packages can be found here: iButton: 1-Wire Drivers for Windows.

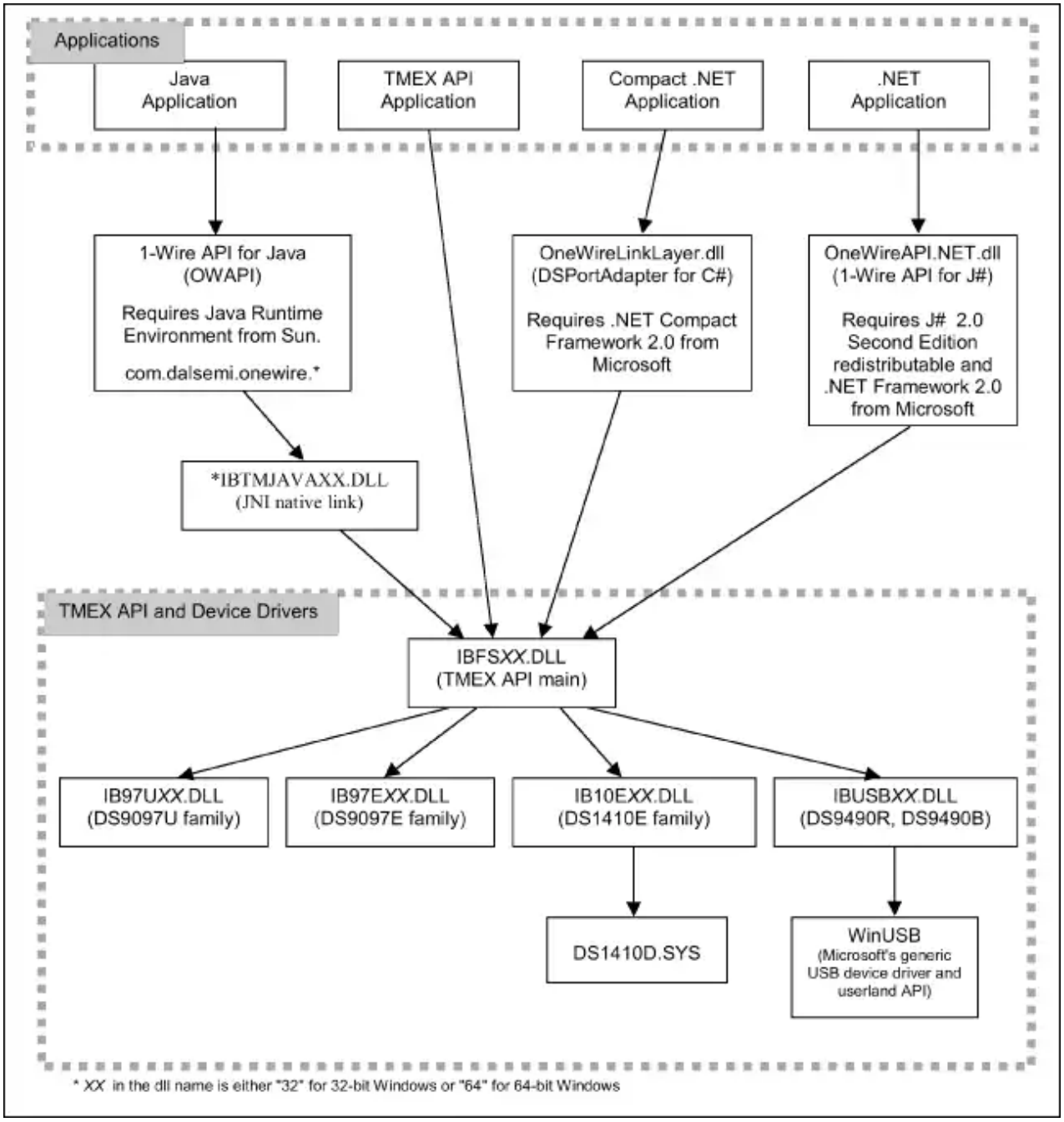

Figure 9. TMEX API drivers and other API connectivity.

Figure 10 lists functions provided by the TMEX API. Note that this API does not provide any nonmemory-device-specific functions.

SESSION

TMEndSession - Relinquishes the 1-Wire net.
TMExtendedStartSession - Requests exclusive use of the 1-Wire net.
TMValidSession - Checks to see if the current 1-Wire net session is valid.

LINK

TMClose - Releases resources for the opened port. (not always applicable)
TMOneWireCom - Sets the speed of the 1-Wire net to Normal (16kb) or Overdrive (142kb).
TMOneWireLevel - Sets the 1-Wire net line level to Normal (5V weak pullup), Power Delivery (5V strong pullup), or Program Level (12V EPROM programming level).
TMProgramPulse - Sends a timed programming pulse to 1-Wire net for EPROM programming.
TMSetup - Checks and verifies the port and adapter is functioning.
TMTouchBit - Sends and receives 1 bit from the 1-Wire net.
TMTouchByte - Sends and receives 1 byte from the 1-Wire net.
TMTouchReset - Resets all devices on the 1-Wire net and returns the result.

NETWORK

TMAccess - Selects the current device and readies it for a device-specific command.
TMAutoOverDrive - Sets the driver to automatically get the device in and out of Overdrive speed.
TMFamilySearchSetup - Sets the current search state to find a specified family type on the next search. (TMNext or TMNextAlarm)
TMFirst - Searches to find the first 1-Wire device on the 1-Wire net.
TMFirstAlarm - Searches to find the first alarming 1-Wire device on the 1-Wire net.
TMNext - Searches to find the next 1-Wire device on the 1-Wire net.
TMNextAlarm - Searches to find the next alarming 1-Wire device on the 1-Wire net.
TMOverAccess - Selects the current device and sets the speed to Overdrive.
TMRom - Retrieves or sets the currently selected device registration number (ROM number).
TMSkipFamily - Skips all of the family type that was found in the last search.
TMStrongAccess - Selects and verifies that the current device is present.
TMStrongAlarmAccess - Selects and verifies that the current device is present AND alarming.

TRANSPORT

TMBlockIO - Sends and receives a block of data to the 1-Wire net preceded with a 1-Wire reset.
TMBlockStream - Sends and receives a block of data to the 1-Wire net with NO 1-Wire reset.
TMExtendedReadPage - Reads an entire page of a memory bank with device-generated CRC verification (not applicable to all device types).
TMProgramByte - Programs a byte into an EPROM-based 1-Wire device.
TMReadPacket - Reads a Universal Data Packet from a page. (See application note 114, "1-Wire File Structure," for Universal Data Packet structure description).
TMWritePacket - Writes a Universal Data Packet to a page.

FILE

TMAttribute - Changes file or directory attributes.
TMChangeDirectory - Reads or changes the current working directory.
TMCloseFile - Closes an opened or created file to release the file handle.
TMCreateFile - Creates a file for writing.
TMCreateProgramJob - Creates a write buffer for logging EPROM programming pending jobs.
TMDeleteFile - Deletes a file.
TMDirectoryMR - Makes or removes a subdirectory.
TMDoProgramJob - Writes the pending EPROM programming jobs.
TMFirstFile - Finds the first file in the current directory.
TMFormat - Formats the 1-Wire File Structure file system.
TMNextFile - Finds the next file in the current directory.
TMOpenFile - Opens a file for reading.
TMReadFile - Reads an opened file.
TMReNameFile - Renames a file or directory.
TMTerminateAddFile - Terminates an 'AddFile.' An 'AddFile' is a special file type on an EPROM device that can be appended to without rewriting.
TMWriteAddFile - Appends or alters an 'AddFile' on an EPROM device.
TMWriteFile - Writes to a file that has been created.

DEVICE

none

OTHER

TMGetTypeVersion - Gets the version information for the adapter driver.
Get_Version - Gets the overall driver version.

Figure 10. TMEX API functions.

Example 4 shows a TMEX code fragment that follows the API usage flow outlined in Figure 3. Like the last three examples, a simple process allows each device on the 1-Wire network to be discovered during each pass through the work loop while other, more sophisticated applications can find just one device type or select from those devices previously found.

```
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 present
      // . . .
     }
   }
   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
  // . . .
}
```

Example 4. TMEX 'C' code example.

Installation

The TMEX API gets installed with the previously mentioned 1-Wire Drivers installation package (which also installs OW.NET API libraries). It is a Microsoft installer package that loads all of the Windows drivers and registry keys for every supported 1-Wire adapter. The driver and API files (no install) are also freely available online for custom installations. The OneWireViewer is a demonstration program that exercises most 1-Wire and iButton devices and is installed with the 1-Wire drivers.

More information on the 1-Wire Drivers and OneWireViewer demo can be found on the iButton web site under "Software Resources."

Other Tools

While the APIs discussed in this document represent a large part of the available resources for communication with 1-Wire devices, it is by no means the only resource. This section outlines some other available resources.

Software Authorization API

Software authorization is simply the locking of a software application to require the presence of a hardware token. The token in this case is an iButton connected to the user's workstation. The application polls for the presence and validity of the iButton before running. It can also continuously query while the application executes. In practice, any of the APIs outlined in this document can be used for this type of application; however, there are specific concerns that must be considered. External drivers present a weakness to the security by allowing a user to replace the driver and thereby defeat the lock. The software authorization API, built upon the 1-Wire PD Kit, has been developed specifically for this type of application and includes linkable modules instead of external drivers.

The design of the Software Authorization API is centered on the simplest use cases possible, to make integration into the software programmer's existing code easy. There are three main services provided by the API:

- Initializing a 1-Wire device
- Authenticating the device
- Clearing a device (so it is no longer a valid part of the system)

Information on the 1-Wire Software Authorization Kit can be found on the iButton web site.

Software Example Search Engine

Maxim continually develops 1-Wire example software applications demonstrating the APIs described in this document. Examples that do not appear in the SDKs can be found by utilizing our online Software Example Search Engine. Many of these examples (with source code) are added to the search engine first (as a separate download) before being added to its respective SDK. The search engine can search for software examples by:

- 1-Wire device (Thermochron, SHA iButton, etc.)
- Platform (Win32, Linux, MxTNI, etc.)
- API (TMEX, 1-Wire Public Domain, 1-Wire .NET, etc.)
- Programming language (C, Java, C#, Visual Basic, Delphi, etc.)

Included in the search results are a description of the program and its appropriate download link.

Third Party

A large volume of third-party software is available for 1-Wire devices. Some are applications produced by Maxim authorized solutions developers (ASDs) for purchase with their solution. A complete list of ASDs and

a solution locator are provided on the iButton site, as well as open-source projects on public forums like SourceForge.net.

Conclusion

This document provides an overview of the characteristics of any 1-Wire API. It also details several different APIs, including supported platforms and programming languages. A quick-reference table providing each API's coverage of each device type helps facilitate selection of which API to use. With the correct API in hand, an application utilizing 1-Wire can readily be designed.

Related Parts	
DS1425	Multi iButton
DS1904	iButton RTC
DS1920	Temperature iButton
DS1921G	Thermochron iButton Device
DS1921H	High-Resolution Thermochron iButton Devices
DS1921Z	High-Resolution Thermochron iButton Devices
DS1922L	iButton Temperature Loggers with 8KB Data-Log Memory
DS1922T	iButton Temperature Loggers with 8KB Data-Log Memory
DS1923	iButton Hygrochron Temperature/Humidity Logger with 8KB Data-Log Memory
DS1963L	4kbit Monetary iButton

DS1963S	iButton Monetary Device with SHA-1 Function	
DS1971	iButton 256-Bit EEPROM	
DS1973	iButton 4Kb EEPROM	Free Samples
DS1977	iButton 32KB EEPROM	
DS1982	iButton 1Kb Add-Only	Free Samples
DS1985	iButton 16Kb Add-Only	Free Samples
DS1986	iButton 64Kb Add-Only	
DS1990A	iButton Serial Number	Free Samples
DS1990R	Serial Number iButton	
DS1991	iButton MultiKey	
DS1992	iButton 1Kb/4Kb Memory	Free Samples
DS1993	iButton 1Kb/4Kb Memory	Free Samples
DS1994	iButton 4Kb Memory Plus Time	
DS1996	iButton 64Kb Memory	Free Samples
DS2401	Silicon Serial Number	Free Samples
DS2404	EconoRAM Time Chip	
DS2405	Addressable Switch	
DS2406	Dual Addressable Switch Plus 1Kb Memory	Free Samples
DS2407	Dual Addressable Switch Plus 1kbit Memory	
DS2408	1-Wire 8-Channel Addressable Switch	Free Samples
DS2409	MicroLAN Coupler	
DS2411	Silicon Serial Number with V_{CC} Input	Free Samples
DS2413	1-Wire Dual Channel Addressable Switch	Free Samples
DS2415	1-Wire Time Chip	
DS2417	1-Wire Time Chip With Interrupt	Free Samples
DS2422	1-Wire Temperature/Data Logger with 8KB Datalog Memory	
DS2423	4kbit 1-Wire RAM with Counter	
DS2430A	256-Bit 1-Wire EEPROM	
DS2431	1024-Bit 1-Wire EEPROM	Free Samples
DS2432	1Kb Protected 1-Wire EEPROM with SHA-1 Engine	Free Samples
DS2433	4Kb 1-Wire EEPROM	

DS2450	1-Wire Quad A/D Converter	
DS2502	1Kb Add-Only Memory	Free Samples
DS2505	16Kb Add-Only Memory	Free Samples
DS2506	64Kb Add-Only Memory	
DS2890	1-Wire Digital Potentiometer	
DS28E04-100	4096-Bit Addressable 1-Wire EEPROM with PIO	
DS28E05	1-Wire EEPROM	Free Samples

Next Steps	
EE-Mail	Subscribe to EE-Mail and receive automatic notice of new documents in your areas of interest.
Download	Download, PDF Format (228.9kB)

APP 155: Jul 08, 2008
APPLICATION NOTE 155, AN155, AN 155, APP155, Appnote155, Appnote 155