アプリケーションノート 155
1-Wire ソフトウェアリソースガイドデバイス解説

はじめに

ダラスセミコンダクタが現在製造中の 1-Wire®デバイスは、iButton®などを含めて 30 点を超えます。利用可能なアプリケーションプログラムインタフェース（API）、ソフトウェア事例、およびこの一連の素子と通信する他のリソース、あるいは単一のデバイスタイプ用の適切なリソースを突き止めるのは煩わしい場合があります。そこで、この解説書では、利用可能なリソースの概要と選択ガイドを提供します。この解説書に記載されたすべてのAPIは無制限に自由に利用することが可能で、ほとんどの場合ソースコード一式を備えています。

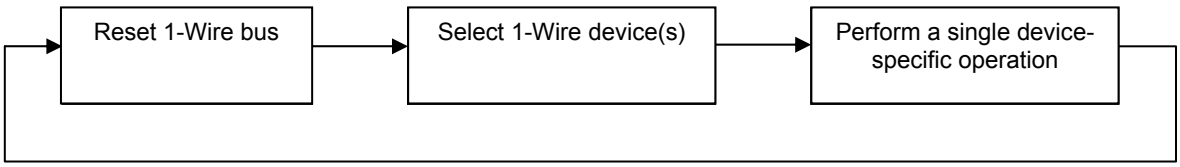
1-Wire の概要

ダラスセミコンダクタの 1-Wireバスは、単一接続で 1 つのマスタと複数の周辺機器との双方向通信を実行するシンプルな信号方式です。全 1-Wireバスデバイスが共有している高機能とは、チップやiButtonのどのデバイスも他のデバイスと重複しない、工場でレーザ処理したシリアル番号を備えていることです。つまり、どのデバイスも他と重複しません。このため、同一バスワイヤに接続できる多数のデバイスの中から、どのデバイスでも個別に選び出すことができます。また、通信用に 1 本のワイヤを 1 台、2 台、または数十台の 1-Wireデバイスでも共有することができるため、バイナリ検索アルゴリズムを使って各デバイスを順次検出することができます。一度各デバイスのシリアル番号が判明すると、そのデバイスにアドレスするためにシリアル番号を使い、どのデバイスも通信用に一意に選択することができます。

どの通信でも、最初はバス全体を同期させる「リセット」を送出するバスマスタが必要です。次にスレーブデバイスが以降の通信用に選択されます。これは、全スレーブを選択するか、特定のスレーブを選択するか(デバイスのシリアル番号によって)、またはバイナリ検索アルゴリズムを使用してバス上の次のスレーブを検出することによって実行することができます。こうしたコマンドは、「ネットワーク」または読取り専用メモリ（ROM)コマンドと総称されています。特定のデバイスが選択されると、その他のすべてのデバイスは脱落し、次のリセットが送出されるまで以降の通信を無視します。

また、あるデバイスがバス通信のために分離されると、マスタはそのデバイスに固有のコマンドを出したり、データを送信したり、またはそのデバイスからデータを読み込んだりすることができます。それぞれのデバイスタイプは、各種の機能を実行し各種の目的を果たすため、各デバイスがいったん選択されるとそれぞれ固有のプロトコルを確保します。たとえ個々のデバイスタイプが異なったプロトコルと機能を備えていても、すべて同じ選択プロセスになり、**図 1** のようなコマンドフローにしたがいます。

図 1. 標準的 1-Wire の通信フロー



各スレーブにおける固有のシリアル番号の絶対必要な部分は、8 ビットのファミリコードです。このコードは、各デバイスモデルに固有のコードです。各デバイスモデルは各種機能を実行しますので、デバイスの制御と信号を送るのに使用するプロトコルを選択するためにこのコードを使用します。ダラスセミコンダクタ製品に対するファミリコードの割当てについては、**表 1** を参照してください。

表 1. ファミリコード対照表

ファミリコード	製品 () - iButtonパッケージ	説明 (特に指定のない限りメモリのサイズはビットで表記)
01 (16 進)	(DS1990A), (DS1990R), DS2401, DS2411	1-Wire ネットアドレス(シリアル番号)のみ
02	(DS1991), DS1425	Multikey iButton、1152 ビットの安全なメモリ
04	(DS1994), DS2404	4kb NV RAM メモリとクロック、タイマ、アラーム
05	DS2405	単一のアドレス可能なスイッチ
06	(DS1993)	4kb NV RAM メモリ
08	(DS1992)	1kb NV RAM メモリ
09	(DS1982), DS2502	1kb EPROM メモリ
0A	(DS1995)	16kb NV RAM メモリ
0B	(DS1985), DS2505	16kb EPROM メモリ
0C	(DS1996)	最大 64kb NV RAM メモリ
0F	(DS1986), DS2506	64kb EPROM メモリ
10	(DS1920), DS1820, DS18S20	警報発生付き温度
12	DS2406, DS2407	1kb EPROM メモリ、2 チャネルアドレス可能スイッチ
14	(DS1971), DS2430A	256 ビット EEPROM メモリと 64 ビット OTP レジスタ
18	(DS1963S)	4kb NV RAM メモリと SHA-1 エンジン
1A	(DS1963L)	書込みサイクルカウンタ付き 4kb NV RAM メモリ
1C	DS28E04-100	4096ビットEEPROMメモリ、2チャネルアドレス可能スイッチ
1D	DS2423	外部カウンタ付き 4kb NV RAM メモリ
1F	DS2409	サブネット用 2 チャネルアドレス可能カプラ
20	DS2450	4 チャネル A/D コンバータ(ADC)
21	(DS1921), (DS1921H), (DS1921Z)	Thermochron®温度ロガー
22	DS1822	Econo ディジタル温度計
23	(DS1973), DS2433	4kb EEPROM メモリ
24	(DS1904), DS2415	リアルタイムクロック(RTC)
26	DS2438	温度、ADC
27	DS2417	割込み付き RTC
28	DS18B20	分解能を調整可能な温度
29	DS2408	8 チャネルアドレス可能スイッチ
2C	DS2890	単一チャネルのディジタルポテンショメータ
2D	DS2431	1024 ビット、1-Wire EEPROM
30	DS2760	温度、電流、ADC
33	(DS1961S), DS2432	SHA-1 エンジン付き 1k EEPROM メモリ
37	(DS1977)	パスワードで保護された 32kB (バイト) EEPROM
3A	(DS2413)	デュアルチャネルのアドレス可能スイッチ
41	(DS1922L), (DS1922T), (DS1923), DS2422	高容量のThermochron (温度)とHygrochron™ (湿度)のロガー

*このリストには、ダラスの 1-Wireデバイスの全タイプ(ファミリ)が含まれているわけではなく、Automatic Information Business Unit (BU)のソフトウェアライブラリがサポートしている一つのディレクトリに過ぎません。

API の基礎

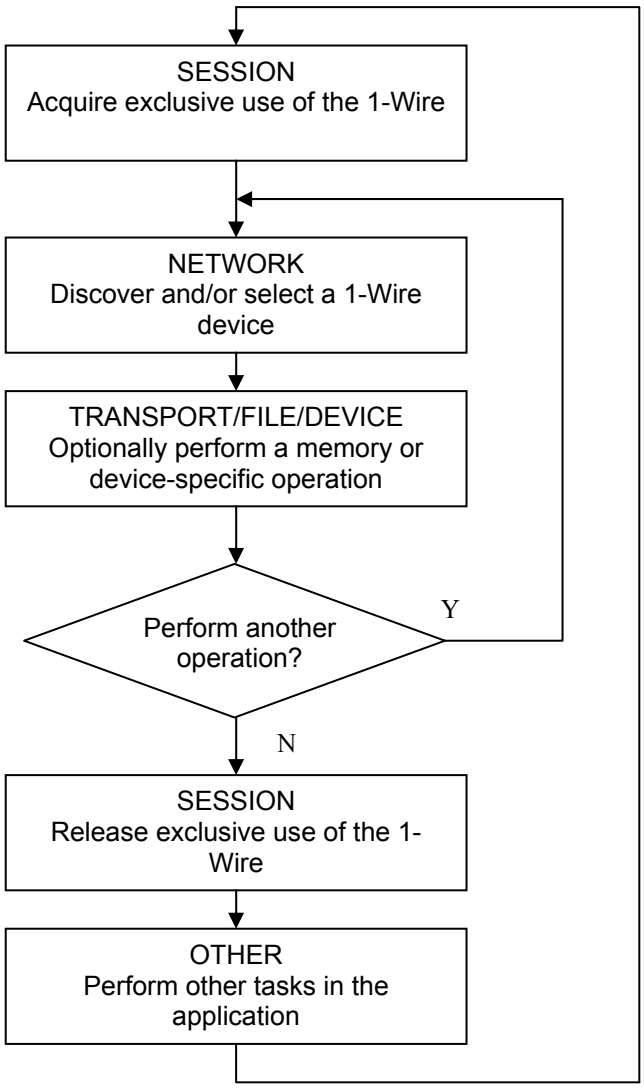
1-Wire デバイスとの通信用の各種アプリケーションプログラムインタフェース（API）は、プロトコルに起因する基礎的な通信課題に対応する共通の機能を備えています。**図 2** は、各種 API の機能を共通グループ化したものを概説しています。ほとんどの 1-Wire デバイスはメモリを装備しているため、メモリ I/O 機能は全デバイスに該当しなくても、共通 API グループとして取り扱われます。すべてのメモリ以外のその他の特殊機能は、デバイス固有のデバイスグループに一括されます。

図 2. API 機能グループ

セッション
1-Wire バスの排他的使用をネゴシエート。これは、複数のプロセスやスレッドが同じ 1-Wire バスに同時にアクセスしようとするオペレーティングシステムや環境では特に重要となります。割込み禁止の単一デバイス上で複数の操作を実行する際には、ネットワークの排他的使用が必要です。
リンク
根源となる 1-Wire バス通信機能。すべての 1-Wire 通信は、全デバイスと読取り/書込みビットをリセットする「リセット」に凝縮することができます。また、このグループは、特別な EPROM プログラミングパルスまたは電力を供給するときのバスの電気的特性を設定する機能を含めることもできます。
ネットワーク
デバイス検出および選択用のネットワーク機能。各 1-Wire デバイスにレーザ処理されている固有のシリアル番号は、各デバイスのネットワークアドレスとして使用されます。この機能は LINK レベル機能で構成されています。シリアル番号は読取り専用メモリ(**R**ead-**O**nly-**M**emory)であるため、1-Wire デバイスのデータシートではこの機能を ROM コマンドと呼んでいます。一部の 1-Wire マスタは、リンク機能よりも効果的なネットワーク機能を内蔵しています。
転送
ブロック通信および基礎的な読取り/書込みメモリ機能。パケットの読取り/書込みメモリ機能を含めることもできます。これらの機能は、「ネットワーク」と「リンク」のグループ機能で構成されています。
ファイル
1-Wireファイル構造を使用したファイルメモリレベル機能(アプリケーションノート 114を参照)。これらの機能は、「ネットワーク」と「転送」のレベル機能で構成されており、1 ページ以上のメモリを備えたデバイスにのみ有効です。
デバイス
デバイス固有の「ハイレベル」機能。多くの場合、これらの機能は、「ネットワーク」、「転送」、および「リンク」のグループ機能で構成され、温度の読取りやスイッチ状態の設定などの操作を実行します。

これらの機能を使用するための標準的な手順は、**図 3** に概説されています。「セッション」機能はデバイスへの通信呼び出しを包括し、これは通常、メモリまたは「デバイス」固有の操作の前に来る「ネットワーク」機能を利用することになります。

図 3. API の使用法フロー



iButton通信の特質は、本質的に「接触」です。これは、デバイスとの接触が常に信頼性があるとは限らないことを意味します。iButtonはリーダに挿入され、読取り時に跳ね返ることがあります。したがって、エラーリカバリに関して一貫した方式を厳格に守る必要があります。これは通常、データ通信でスプリアスエラーが検出されたときに再試行を行うこと、およびCRCチェックを利用することを必要とします。APIのファイルI/O機能は、アプリケーションノート 114の「1-Wire File Structure (1-Wireファイル構造)」の箇所に詳述されている標準ファイル構造を使用します。この構造は、読取り中のデータの妥当性を迅速に検証するために、データの全ページでCRC16 を使用します。たいていの 1-Wire API機能には、自動再試行機能が全くないか少ししかありません。再試行は、アプリケーションが制御します。1-Wire通信実行時のエラー回復とリスク評価の方法については、アプリケーションノート 159:『超高信頼性 1-Wire通信』を参照してください。

API の選択

このアプリケーションノートでは、主として 5 種類の API を取り上げています。これらの API は各種プラットフォームで動作し、各種言語を使用し、また各種機能を備えています。**表 2** は 5 種類の API を簡単な説明付きで紹介しており、また**表 3** は、言語で分類された利用可能な API をオペレーティングシステムに割り当てています。

表 2. API の概要

API	省略形	概要
1-Wire パブリックドメイン	PD	Cで記述された完全なオープンソースのパブリックドメインAPIであり、多数のPCオペレーティングシステム、ハンドヘルドオペレーティングシステム、およびマイクロコントローラのプラットフォーム間で移植することができるように設計されています。PCプラットフォームの場合、32 ビットMicrosoft Windows®上でネイティブドライバのライブラリを通じてすべての 1-Wireアダプタ(マスタ)をサポートしており、またクロスプラットフォームライブラリを使用する他のPCオペレーティングシステム上では、特定の 1-Wireアダプタ(DS9097Uシリアルアダプタ やDS9490 USBアダプタ)をサポートしています。
Java 用 1-Wire API	OWAPI	ほぼすべての 1-Wire デバイスをサポートする完全なオープンソースの高レベル Java API。本来の 1-Wire マスタのサポートに加えて、クロスプラットフォームライブラリを通じて、DS9097U シリアルアダプタと DS9490 USB アダプタもサポートしています。
.NET 用 1-Wire API	OW.NET	Microsoft の.NET フレームワーク用の J#でコンパイルした OWAPI コードベース。Compact .NET Framework (Windows CE マシン用)の場合、低レベルの 1-Wire リンク層は C#に移植されており、ダウンロードが可能です。
1-Wire COM	OWCOM	OWAPI 用の Windows コンポーネントオブジェクトモデル(COM)のラッパー。Java Script および Visual Basic Script などの標準言語およびスクリプト言語から利用可能。注:OWCOM の適正な動作は、Microsoft Java 仮想マシン (MSJVM)の有無に依存しています。Microsoft は、MSJVM をサポートしなくなりました。
TMEX API	TMEX	Windows 32 ビットプラットフォームで 1-Wire マスタアダプタをすべてサポートしています。リンクおよびファイル I/O 機能を提供していますが、デバイス機能は提供していません。ドライバはクローズドソースです。この API は、すべての 1-Wire アダプタタイプにアクセスできるように他の API によって呼び出されます。

表 3. API のオペレーティングシステムおよび言語の適用範囲

Language / OS	TMEX/OWCOM/OW.NET (Microsoft Windows language independent)	C	Java
Windows XP	TMEX/OWCOM/OW.NET	PD	OWAPI
Windows 2000	TMEX/OWCOM/OW.NET	PD	OWAPI
Windows ME	TMEX/OWCOM/OW.NET	PD	OWAPI
Windows 98	TMEX/OWCOM/OW.NET	PD	OWAPI
Windows 95	TMEX	PD	OWAPI
Win3.1		PD	
DOS		PD	
Palm®		PD	
VISOR®		PD	
Pocket PC/CE	(OW.NET)	PD	
Linux and other UNIX®-based OS's		PD	OWAPI
TINI*		PD - without TINI OS	OWAPI

( ) - サポートを予定しているが未完了

*TINI®は、ダラスセミコンダクタが開発したJavaベースのOSを備えた組み込みプラットフォームです。

各デバイスファミリのサポートは、API によっても異なります。**表 4** は、各 API で利用可能なサポートを示すフラグとともに、現在利用可能な 1-Wire デバイスをすべて紹介しています。表 4 のフラグに対する「要点」は、表の下部にあります。

濃淡の影がないデバイス欄は、API によって完全サポートされることに留意してください。淡い影付きの欄は部分的サポートで、濃い影付きの欄は最小限のサポートを示しています。

表 4. デバイス別 API サポート

Device	FC	Description	TMEX	PD	OWAPI	OW.NET	OWCOM
DS1425	02	Multikey iButton, 1152-bit secure memory	AB	ABI	ABC	ABC	ABC
DS1427	04	4kb NV RAM memory and clock, timer, alarms	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1820	10	Temperature and alarm trips	AB	ABI	ABCI	ABCI	ABCI
DS18B20	28	Adjustable resolution temperature	AB	AB	ABI	ABI	ABI
DS18S20	10	Temperature and alarm trips	AB	ABI	ABCI	ABCI	ABCI
DS1982	09	1kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDE
DS1985	0B	16kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDE
DS1986	0F	64kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDEFGH
DS1904	24	RTC	AB	AB	ABI	ABI	ABI
DS1920	10	Temperature and alarm trips	AB	ABI	ABCI	ABCI	ABCI
DS1921G DS1921H DS1921Z	21	Thermochron temperature logger	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1922L DS1922T DS1923	41	High-capacity Thermochron (temperature) and/or Hygrochron (humidity) loggers	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1961S	33	1kb EEPROM memory with SHA-1 engine	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1963L	1A	4kb NV RAM memory with write-cycle counters	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1963S	18	4kb NVRAM memory and SHA-1 engine	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1971	14	256-bit EEPROM memory and 64-bit OTP register	ABD	ABCDI	ABCDI	ABCDI	ABCDI
DS1973	23	4kb EEPROM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1977	37	Password-protected 32kB (bytes) EEPROM	AB	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1990A DS1990R DS2411	01	1-Wire address only	AB	AB	AB	AB	AB
DS1991	02	Multikey iButton, 1152-bit secure memory	AB	ABC	ABC	ABC	ABC
DS1992	08	1kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1993	06	4kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1994	04	4kb NV RAM memory and clock, timer, alarms	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS1995	0A	16kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS1996	0C	64kb NV RAM memory	ABDE	ABCDE	ABCDEFGH	ABCDEFGH	ABCDEFGH
DS2401	01	1-Wire address only	AB	AB	AB	AB	AB
DS2405	05	Single switch	AB	ABI	ABI	ABI	ABI

DS2404	04	4kb NV RAM memory and clock, timer, alarms	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS2406 DS2407	12	1kb EPROM memory, 2-channel addressable switch	ABCE	ABCDEI	ABCDEI	ABCDEI	ABCDEI
DS2408	29	8-channel addressable switch	AB	ABI	ABI	ABI	ABI
DS2409	1F	Dual switch, coupler	AB	ABI	ABI	ABI	ABI
DS2413	3A	Dual-channel addressable switch	AB	ABI	ABI	ABI	ABI
DS2415	24	RTC	AB	AB	ABI	ABI	ABI
DS2417	27	RTC with interrupt	AB	AB	ABI	ABI	ABI
DS2422	41	High-capacity Thermochron (temperature)/Hygrochron (humidity) logger	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS2423	1D	4kb NV RAM memory with external counters	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS2430A	14	256-bit EEPROM memory and 64-bit OTP register	ABD	ABCDI	ABCDI	ABCDI	ABCDI
DS2431	2D	1024-bit EEPROM memory	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS2432	33	1kb EEPROM memory with SHA-1 engine	ABDE	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI
DS2450	20	Quad ADC	AB	ABI	ABI	ABI	ABI
DS2438	26	Temperature, ADC	AB	ABI	ABCI	ABCI	ABCI
DS2502	09	1kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDE
DS2505	0B	16kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDE
DS2506	0F	64kb EPROM memory	ABCE	ABCDE	ABCDE	ABCDE	ABCDE
DS2760	30	Temperature, current, ADC	AB	AB	ABI	ABI	ABI
DS2890	2C	Single-channel digital potentiometer	AB	AB	ABI	ABI	ABI
DS2760	30	Temperature, current, ADC	AB	AB	ABI	ABI	ABI
DS28E04 -100	1C	4096-bit EEPROM memory, two-channel addressable switch	AB	ABCDEI	ABCDEFGHI	ABCDEFGHI	ABCDEFGHI

欄の濃淡別サポート

完全サポート
部分サポート
最小サポート

サポートのフラグ

A. 1-Wire リンクの基本サポート
B. 1-Wire ネットワークのサポート
C. 転送メモリバイトの読取り/書込みサポート
D. 転送メモリパケットの読取り/書込みサポート
E. 1-Wireファイル構造タイプAAのサポート(ファイル構造タイプについては、アプリケーションノート 114を参照)
F. 1-Wire ファイル構造タイプ AB のサポート
G. 1-Wire ファイル構造タイプ BA のサポート
H. 1-Wire ファイル構造タイプ BB のサポート
I. その他のデバイス固有のサポート

1-Wire パブリックドメイン(PD)の概要

1-Wire PD APIで提供される機能は「C」で完全記述され、TMEX APIでサポートされていないプラットフォームでの使用を目的としています。「1-Wireネット」(またはMicroLAN)は、1 つのマスタと 1 つ以上のスレーブデバイスを備える一本の配線とグランドのネットワークです。このAPIはスレーブデバイスを識別し、スレーブデバイスと通信することが可能な 1-Wireマスタを創造します。また、iButtonを含めたすべてのダラスセミコンダクタの 1-Wireデバイスと通信するように、あらゆる 1-Wire、転送、およびファイルレベルサービスを提供します。このAPIキットおよびプラットフォームの構築(ビルド)例は、iButtonのWebサイトにあります。

この API の「C」ソースコードは、移植することができるように設計されています。特定プラットフォームを完成するための「TODO (要実行事項)」テンプレートが提供されています。Windows 32 ビット、Windows 16 ビット、DOS 16 ビット、Linux、Palm、および Pocket PC などを含めた複数のプラットフォームの実装例が提供されています。また、こうしたプラットフォーム実装を使用する複数のアプリケーション例もあります。

「general」、「userial」、および「other」などの 3 つの移植可能なソースファイルのセットがあります。1 つ目のセットは汎用的なもので、基本的なリンクの 1-Wire 通信機能(general)をすでに備えているプラットフォームを対象にしています。これは、ハードウェアに依存する最小レベルのセットです。移植可能なソースファイルの 2 つ目のセットは、ユーザがシリアルポート(RS232)を備え、「Universal Serial 1-Wire Line Driver Master: DS2480B」(userial)の利用を望んでいるものと想定しています。このチップはシリアルポートに対するコマンドを受け、1-Wire オペレーションを実行し、その結果をシリアルポートに返送します。ソースコードは、目的の 1-Wire 操作を DS2480B へのシリアル通信パケットに変換します。プラットフォームに提供する必要があるモジュールは、シリアルポートの読取り/書込みの基本モジュールだけです。DS2480B は、すべての DS9097U シリーズのシリアルアダプタに使用されているインタフェースチップです。最後に、移植可能なソースファイルの 3 つ目のセットは、特定の 1-Wire アダプタ機能および又は、先の 2 つのカテゴリにはまったく入らないもの(other)を取り扱っています。この一例として、USB ポートを取り扱う例があり、具体的には、DS2490「USB と 1-Wire 間のブリッジチップ」を利用します。多くの面で、これは general のビルドと似ていますが、DS2490 の特異性に合わせて修正されています。いずれのファイルセット(general、userial、または other)を使用する場合でも、最終的には、同じ API が実際にソフトウェア開発者に対して公開されます。

特に、このアプリケーションノートでは、1-Wireパブリックドメインキット(「other」ビルドに分類される)の、オープンソース型クロスプラットフォームのlibusbビルドについても取り上げています。libusbは、多くの種類のOSに移植されているオープンソースのUSBライブラリです。この特別な 1-Wire PD APIのビルドは、その後 1-Wire PD APIを構築するのに必要となるlibusb関数にコンパイルされます。

利用可能なさまざまなファイルセットについては、以下の表を参照してください。これはすでに表に示した各プラットフォーム用にビルドされたものであり、ダウンロードして入手可能です。

表 5. 1-Wire ファイルセットと事前にビルドされたバイナリ

移植可能なソースファイルのセット(ビルド)	プラットフォーム	ポート	説明
userial	Win32、Win16、DOS、WinCE/Pocket PC、Linux、(その他の UNIX)、DS550	COM	DS9097U 1-Wire シリアルポートアダプタとその他の DS2480B ベースのソリューションをサポートしています(組み込みを含む)。
general	Win32	LPT	Win32 LPT ビルドは、Windows 上で DS1410E パラレルポートアダプタをサポートしています。
	DS550	マイクロプロセッサ(µP)のポートピン	DS550 の general ビルドは、µP のポートピンを使用しています。
Other「libusb」	Win32、Linux、Macintosh、(その他の UNIX)	USB	DS9490 USB 1-Wire アダプタと、特に適切な libusb ドライバをインストール済みのその他の DS2490 ベースの USB ソリューションをすべてサポートしています。
Other「usb」	Win32	USB	DS9490 USB 1-Wire アダプタと、DS2490.SYS デバイスドライバを特別に装備した Win32 の下で、その他の DS2490 ベースの USB アダプタをすべてサポートしています。
Other「wrapper」(TMEX)	Win32	USB, COM, LPT	TMEX API を包括し、これによって、Windows の下でマルチポートをサポートするようにします(それぞれ DS9490、DS9097U、DS1410E)。
Other「multi-port」	Win32	USB, COM, LPT	低レベルのネイティブ Win32 ドライバと直接通信することでマルチポートをサポートしています。

注:最新のプラットフォームのビルドについては、1-WireパブリックドメインのWebサイトを参照してください。

移植可能なソースコードファイルのこれらのセットは、同じ 1-Wire API 機能を実装し、また互換性を持っています。**図 4** は、1-Wire PD コードベースのバージョン 3.00 用の利用可能な API を紹介しています。非メモリデバイス固有の機能は多数存在す

るため、詳述していないことに注意してください。図 4 は、機能を提供するソースファイルと新規プラットフォームに必要なモジュールを対応付けています。

ダウンロードした PD キット内の移植可能な「C」モジュールだけでなく、1-Wire 通信を実行するための限定されたマイクロプロセッサのアセンブリ例もあります。

このAPIのファイル機能は、アプリケーションノート 114の規定にあるように、1-Wireファイル構造タイプ「AA」を実装しています。

この API の名前が示すように、提供されるソースコードは、できるだけパブリックドメインに近いライセンスを所持しています。開発者は、制限なく自由にこのコードを使用し、自らのアプリケーションに組み込むことができます。

図 4. PD API 機能

セッション
owAcquire - 1-Wire ネットを取得。 *owRelease* - 以前取得した 1-Wire ネットを開放。
リンク
owHasOverDrive - アダプタがオーバドライブ能力を備えているかどうかを指定。 *owHasPowerDelivery* - アダプタが電力を供給可能かどうかを指定。 *owHasProgramPulse* - EPROM プログラミング電圧が利用可能かどうかを指定。 *owLevel* - 1-Wire ネットラインレベルを Normal (5V の弱プルアップ)、Power Delivery (5V の強プルアップ)、または Program Level (12V の EPROM プログラミングレベル)に設定。 *owProgramPulse* - EPROM 1-Wire デバイスの書込み用に、時間設定されたプログラミングパルスを送信。 *owReadBitPower* - 1 ビットを読み取り、電力を随意に供給。 *owReadByte* - オール 1 (0xFF)を送信して、1-Wire ネットから 8 ビットを受信。 *owSpeed* - 1-Wire ネットの速度を Normal (16k ビット)または Overdrive (142k ビット)に設定。 *owTouchBit* - 1-Wire ネットで 1 ビットを送受信。 *owTouchByte* - 1-Wire ネットで 8 ビットを送受信。 *owTouchReset* - 1-Wire ネット上の全デバイスをリセットし、結果を返す。 *owWriteByte* - 1-Wire ネットに 8 ビットを送信し、受信したエコーが一致していることを確認。 *owWriteBytePower* - 1-Wire ネットに 8 ビットの通信を送信し、電力を供給。
ねっとわーく
owAccess - 現行デバイスを選択し、デバイス固有のコマンドに対してデバイスを準備させる。 *owFamilySearchSetup* - 特定のファミリタイプを検出するために次の検索(*owNext*)を設定。 *owFirst* - 1-Wire ネット上の「最初の」1-Wire デバイスを検出するために検索。 *owNext* - 1-Wire ネット上の「次の」1-Wire デバイスを検出するために検索。 *owOverdriveAccess* - 現行デバイスを選択し、速度を Overdrive に設定。 *owSerialNum* - 現在選択中のデバイスのシリアル番号(ROM 番号)を回収、または設定。 *owSkipFamily* - 最後の検索で検出されたファミリタイプの 1-Wire デバイスをすべてスキップ。 *owVerify* - 現行デバイスを選択し、現行デバイスが存在していることを検証(警報発生はオプション)。

転送
owBlock - リセットはオプションで 1-Wire ネットに 1 データブロックを送受信。 *owCanLockPage* - 特定のメモリバンクにロック可能なページがあるかどうかを確認。 *owCanLockRedirectPage* - リダイレクトしないようにロック可能なページが特定のメモリバンクにあるかどうかを確認。 *owGetAlternateName* - 代替の部品番号や部品名を取得。 *owGetBankDescription* - メモリバンクの文字列記述を取得。 *owGetDescription* - 1-Wire デバイスタイプの短い記述を取得。 *owGetExtraInfoDesc* - その他情報の内容に関する記述を取得。 *owGetExtraInfoLength* - このメモリバンク内のその他情報をバイトで表した長さを取得。 *owGetMaxPacketDataLength* - パケットのバイトで表した最大データ長を取得。 *owGetName* - 1-Wire デバイスの部品番号を文字列として取得。 *owGetNumberBanks* - 特定 1-Wire ファミリグループのメモリバンクの番号を取得。 *owGetNumberPages* - 特定メモリバンクのページ数を取得。 *owGetPageLength* - 特定メモリバンクの、バイトで表したロー(raw)ページ長を取得。 *owGetSize* - バイトで表した特定メモリバンクのサイズを取得。
owGetStartingAddress - 特定メモリバンクの物理開始アドレスを取得。 *owHasExtraInfo* - 読取り時にこのメモリバンクのページがその他情報を提供するかどうかを確認。 *owHasPageAutoCRC* - ページ読取り時にメモリバンクがデバイス生成 CRC 検査を備えているかどうかを確認。 *owIsGeneralPurposeMemory* - メモリバンクが汎用ユーザメモリであるかどうかを確認。 *owIsNonvolatile* - 現行メモリバンクが不揮発性であるかどうかを確認。 *owIsReadOnly* - メモリバンクが読取り専用であるかどうかを確認。 *owIsReadWrite* - メモリバンクが読取り/書込みであるかどうかを確認。 *owIsWriteOnce* - メモリバンクが EPROM などのように追記型であるかどうかを確認。 *owNeedsPowerDelivery* - このメモリバンクが書込みに「電力供給」を必要とするかどうかを確認。メモリバンクが書込みに電力供給を要求するかどうかを確認 *owNeedsProgramPulse* - このメモリバンクが書込みに「プログラムパルス」を必要とするかどうかを確認。 *owRead* - ローモードでメモリバンクの一部を読み取る(パケット、CRC なし)。 *owReadPage* - ローモードでメモリバンクの 1 ページ全体を読み取る(パケット、CRC なし)。 *owReadPageCRC* - デバイス生成 CRC 検査付きのメモリバンクの 1 ページ全体を読み取る。 *owReadPageExtra* - 「その他」情報を含んだ、メモリバンクの 1 つのローページ全体を読み取る(パケット、CRC なし)。 *owReadPageExtraCRC* - 「その他」情報とデバイス生成 CRC 検査を含んだメモリバンクの 1 つのローページ全体を読み取る。 *owReadPagePacket* - メモリバンクの 1 ページからUniversal Data Packetを読み取る(Universal Data Packet構造の説明については、アプリケーションノート 114を参照)。 *owReadPagePacketExtra* - 「その他」情報付きのメモリバンクの 1 ページから Universal Data Packet を読み取る。 *owRedirectPage* - メモリバンクが転送可能なページを備えているかどうかを確認。 *owWrite* - ローモードでメモリバンクの一部に書き込む。 *owWritePagePacket* - メモリバンクの 1 ページに Universal Data Packet を書き込む。
ファイル
owAttribute - ファイルの属性を変更。 *owChangeDirectory* -現在のディレクトリを変更。 *owCloseFile* - ファイルを閉じる。 *owCreateDir* - ディレクトリを作成。 *owCreateFile* - 書込み用にファイルを作成。 *owCreateProgramJob* - EPROM プログラミング保留ジョブのロギング用書込みバッファを作成。 *owDeleteFile* - ファイルを削除。 *owDoProgramJob* - 保留の EPROM プログラミングジョブを書き込む。 *owFirstFile* - 現在のディレクトリ内の最初のファイルを検出。 *owFormat* - 1-Wire ファイル構造ファイルシステムをフォーマット。

owGetCurrentDir - 現在のディレクトリを取得。
owNextFile - 現在のディレクトリ内の次のファイルを検出。
owOpenFile - 読取り用にファイルを開く。
owReadFile - 開いたファイルを読み取る。
owReadFile - ファイルからデータを読み取る。
owRemoveDir - ディレクトリを削除。
owReNameFile - ファイル名を変更。
owWriteFile - 作成されたファイルに書き込む。

デバイス

DoAtoDConversion - DS2450 で A/D 変換を実行。
ReadSwitch12 - DS2406 のスイッチ状態を読み取る。
readCounter - DS2423 1-Wire チップ上の特定のメモリページに関連するカウンタ値を読み取る。
… (以上のほかに、多数のデバイス固有の機能があります。)

例 1 は、図 3 で概説した「API の使用法フロー」にしたがった PD コード部分を紹介しています。簡単にするために、ワークループを通過するごとに 1-Wire ネットワーク上の各デバイスを検出しています。より高度なアプリケーションであれば、1 つのデバイスタイプだけを検出したり、以前の検索で検出されたデバイスを選択したりすることができます。

例 1. PD コード例

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

図 5a と**図 5b** は、2 組の 1-Wire PD ライブラリで各々構成される C 言語モジュールを列記しています。**図 5c** は、他の実装の 1 つ、具体的には libusb ビルドについても示していることに注意してください。また、新規プラットフォームへのライブラリ移植のために提供すべき「TODO」機能も紹介しています。「TODO」機能を実装するプラットフォームリンクファイルの数例がキットで提供されています。

図 5a. PD「USERIAL」の実装

セッション					
owsesu.c					
リンク					
owllu.c	ds2480ut.c	ds2480.h			
ネットワーク					
ownetu.c	crcutil.c (コンパイルに必要)				
転送					
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			
ファイル					
owcache.c	owfile.c owfile.h	owpgrw.c	owprgm.c		
デバイス					
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	thermo21.h	time04.c	time04.h	weather.c	weather.h
その他ユーティリティ					
ioutil.c	owerr.c	findtype.c	ownet.h	screenio.c	sprintf.c
crcutil.c					
TODO (要実行事項)					
以下の機能を実装する SERIAL インタフェースモジュールを提供。 *BreakCOM** - 最低 2ms 間、シリアルポート上に「BREAK」を送信。 *CloseCOM* - 以前開いたシリアルポートを閉じる(一部プラットフォームではオプション)。 *FlushCOM** - 保留中の書込み操作を完了させ、入力バッファをクリアする。 *msDelay** - 指定のミリ秒以上遅延させる。 *msGettick* - 増分ミリ秒カウンタを返す(一部の例ではオプション)。 *OpenCOM* - 通信用の指定シリアルポートを開く(一部プラットフォームではオプション)。 *ReadCOM** - シリアルポートから指定数のバイトを読み取る。 *SetCOMBaud* - シリアル BAUD 速度を指定速度に変更(オーバドライブはオプション)。 *WriteCOM** - シリアルポートに指定数のバイトを書き込む。 *基本操作に最低限必要な機能					

図 5b. PD「GENERAL」の実装

セッション
(TODO 参照)
リンク
(TODO 参照)
ネットワーク
ownet.c　　crcutil.c (コンパイルに必要)
転送
「owtrnu.c」が「owtran.c」に置き換えられる以外は、USERIAL の実装と同じ。
ファイル
USERIAL の実装と同じ。
デバイス
USERIAL の実装と同じ。
その他ユーティリティ
USERIAL の実装と同じ。
TODO (要実行事項)
以下の機能を実装する LINK および SESSION インタフェースモジュールを提供: *owAcquire* - 1-Wire ネットを取得。 *owRelease* - 以前取得した 1-Wire ネットを開放。 *owHasOverDrive* - アダプタがオーバドライブ機能を備えるかどうかを指定。 *owHasPowerDelivery* - アダプタが電力を供給可能かどうかを指定。 *owHasProgramPulse* - EPROM プログラミング電圧が利用可能かどうかを指定。 *owLevel* - 1-Wire ネットラインレベルを Normal (5V の弱プルアップ)、Power Delivery (5V の強プルアップ)、または Program Level (12V の EPROM プログラミングレベル)に設定。 *owProgramPulse* - EPROM 1-Wire デバイスの書込み用に、時間設定されたプログラミングパルスを送信。 *owReadBitPower* - 1 ビットを読み取り、随意に電力を供給。 *owReadByte* - オール 1 (0xFF)を送信して、1-Wire ネットから 8 ビットを受信。 *owSpeed* - 1-Wire ネットの速度を Normal (16K ビット)または Overdrive (142K ビット)に設定。 *owTouchBit** - 1-Wire ネットで 1 ビットを送受信。 *owTouchByte* - 1-Wire ネットで 8 ビットを送受信。 *owTouchReset** - 1-Wire ネット上の全デバイスをリセットし、結果を返す。 *owWriteByte* - 1-Wire ネットに 8 ビットを送信し、受信したエコーが一致していることを確認。 *owWriteBytePower* - 1-Wire ネットに 8 ビットの通信を送信し、電力を供給。 * 基本操作に最低限必要な機能

図 5c. PD「LIBUSB」の実装

セッション
libusbses.c
リンク
libusbwlink.c (Linux/Macintosh/UNIX の場合には libusbllink.c)、libusbds2490.c、libusbds2490.h、usb.h (libusb ヘッダファイル)
ネットワーク
libusbnet.c　　crcutil.c (コンパイルに必要)
転送
「owtrnu.c」が「libusbtran.c」に置き換えられる以外は、USERIAL の実装と同じ。
ファイル
USERIAL の実装と同じ。
デバイス
USERIAL の実装と同じ。
その他ユーティリティ
USERIAL の実装と同じ。
TODO (要実行事項)
以下の機能を実装する LINK および SESSION インタフェースモジュールを提供: *owAcquire* - 1-Wire ネットを取得。 *owRelease* - 以前取得した 1-Wire ネットをリリース。 *owHasOverDrive* - アダプタがオーバドライブ機能を備えるかどうかを指定。 *owHasPowerDelivery* - アダプタが電力を供給可能かどうかを指定。 *owHasProgramPulse* - EPROM プログラミング電圧が利用可能かどうかを指定。 *owLevel* - 1-Wire ネットラインレベルを Normal (5V の弱プルアップ)、Power Delivery (5V の強プルアップ)、または Program Level (12V の EPROM プログラミングレベル)に設定。 *owProgramPulse* - EPROM 1-Wire デバイスの書込み用に、時間設定されたプログラミングパルスを送信。 *owReadBitPower* - 1 ビットを読み取り、電力を任意に供給。 *owReadByte* - オール 1 (0xFF)を送信して、1-Wire ネットから 8 ビットを受信。 *owSpeed* - 1-Wire ネットの速度を Normal (16K ビット)または Overdrive (142K ビット)に設定。 *owTouchBit** - 1-Wire ネットで 1 ビットを送受信。 *owTouchByte* - 1-Wire ネットで 8 ビットを送受信。 *owTouchReset** - 1-Wire ネット上の全デバイスをリセットし、結果を返す。 *owWriteByte* - 1-Wire ネットに 8 ビットを送信し、受信したエコーが一致していることを確認。 *owWriteBytePowe*r - 1-Wire ネットに 8 ビットの通信を送信し、電力を供給。 * 基本操作に最低限必要な機能

インストール

1-Wire の PD API は C モジュールのセットであるため、正式なインストールがありません。「構築(ビルド)」例で紹介されているように、必要なモジュールはアプリケーションに直接コンパイルされます。これは、開発者が Windows DLL などのロード可能なライブラリにモジュールを統合することを阻むことはありません。

一部のビルドは、ネイティブの 1-Wire アダプタドライバまたはこれに相当するものを必要とすることに留意してください。たいていの OS プラットフォームは組み込まれたシリアルポートドライバを持っているため、1-Wire PD API の userial ビルドは、他のいずれのドライバも必要としません。ただし、USB とパラレルポートの 1-Wire アダプタのサポートについては、ネイティブまたはクロスプラットフォームのドライバをインストールすることが必要となります。適切なドライバのダウンロードについては、オンラインで実際に利用可能な 1-Wire PD キットのビルドの資料を参照してください。

Java 用 1-Wire API (OWAPI)の概要

Java 用 1-Wire API は、Java での 1-Wire アプリケーションの構築を目的として、最初から極めて堅牢で高度なオブジェクト指向の基盤となるように設計されています。この API は、移植したりクロスプラットフォームのソフトウェア開発をしたりするプログラマの能力を拡大し、1-Wire が使用された製品の製品化までの時間を短縮します。

このAPIは、多数のJavaクラスとインタフェースから構成されています。1-Wire APIのJavaクラスのひとつの特別なグループがコンテナ(クラスOneWireContainer)です。iButton を含めて特定の 1-Wireデバイス用のサポートは、コンテナを通じて提供されています。このAPIは、大部分の 1-Wireデバイスを代表する 30 を超える各種コンテナタイプを備えています。各コンテナは個々のデバイスの機能を要約し、実装します。

「コンテナ」は、物理的な 1-Wireアダプタ(クラスDSPortAdapter)である 1-Wireアダプタクラスを通じて 1-Wireデバイスとやりとりします。アダプタのインスタンス(要求)は、プロバイダのクラス(クラスOneWireAccessProvider)から作成されます。1-Wireアダプタの実際の実装はプラットフォームによって変わりますが、すべて同じインタフェースを持っています。一部のプラットフォームではネイティブなドライバを使用していますが、大部分のプラットフォームは、少なくとも、RXTX (クロスプラットフォームのシリアルCOMポートAPI)を使用してDS9097U-XXXシリアルアダプタをサポートしています。このAPIは、RXTXのWebサイトから入手することができます。

Java用 1-Wire APIソフトウェア開発キットは、iButtonのWebサイトで入手することができます。1-Wire PDキットのように、OWAPIのJavaソース一式はパブリックドメイン形式のライセンスに基づいて提供されます。

図 6 は、この API の標準的なオブジェクト生成手順を紹介しています。「プロバイダ」は、デバイスの「コンテナ」のインスタンス(要求)を順次作成可能な「アダプタ」のインスタンス(またはエニュメレーション(列挙))を作成します。これによってデバイスとの通信は、ほぼ例外なく「コンテナ」を通じて行われます。

図 6. OWAPI オブジェクトの作成

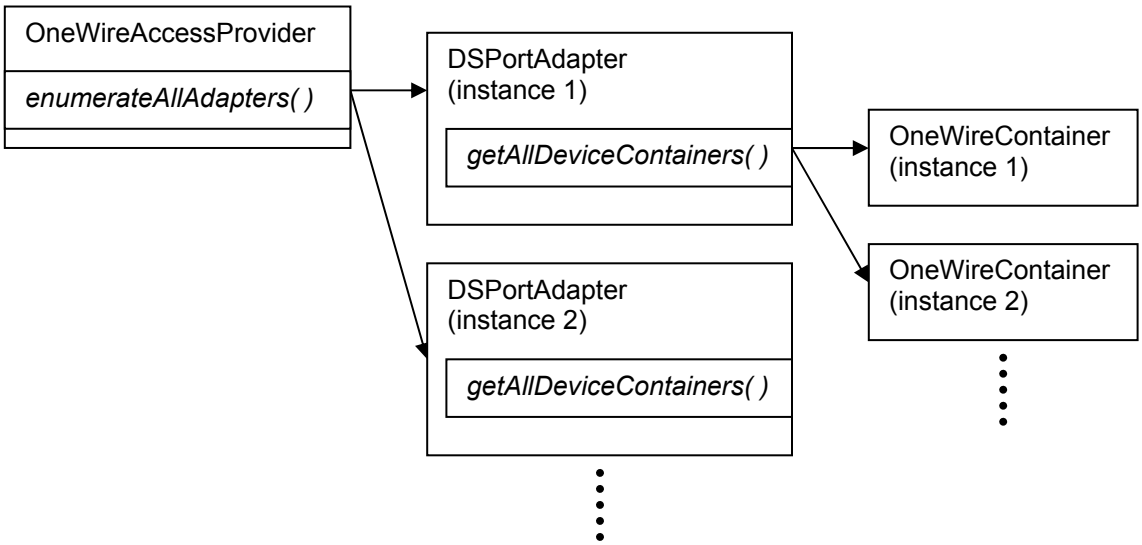


図 7 は、「コンテナ」の一般的な機能を紹介しています。メモリ搭載のデバイスは、各メモリバンク用に「メモリバンク」インスタンスを作成します。メモリは、バンクの機能セットに応じて、各バンクに分類されます。たとえば、揮発性バンクもあれば、不揮発性バンクもあります。または、バンクは汎用メモリもあれば、デバイスの機能を変更する「メモリ割当て」の場合もあります。

図 7. OWAPI ONEWIRECONTAINER の機能

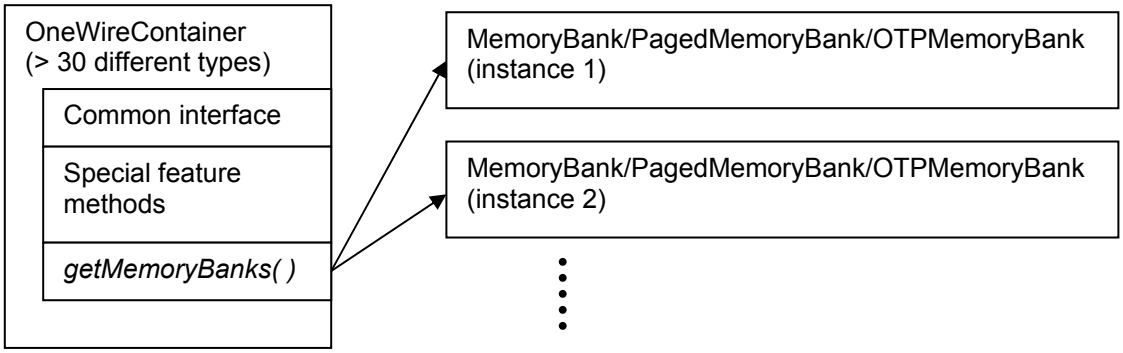


図 8 は、基本 OWAPI クラスで提供されたメソッドを紹介しています。クラスまたはパッケージは**太字**で表示しています。各コンテナは個々のデバイスタイプを操作するハイレベルなメソッドを備えているため、アダプタの LINK レベルメソッドは通常直接呼び出されないことに注意してください。

図 8. OWAPI の機能

セッション
com.dalsemi.onewire.adapter.DSPortAdapter *beginExclusive* - 1-Wire ネットの排他的使用を取得。 *endExclusive* - 1-Wire ネット上で排他的ロックを開放。
リンク
com.dalsemi.onewire.adapter.DSPortAdapter *canBreak* - 1-Wire「ブレーク」(long low)操作がアダプタによってサポートされているかどうかを確認。 *canDeliverPower* - 「強プルアップ」電力供給がアダプタによってサポートされているかどうかを確認。 *canDeliverSmartPower* - 「スマート」電力供給がアダプタによってサポートされているかどうかを確認。 「スマート」電力供給とは、電力消費の低減時を検出し、電力供給を自動停止する機能。 *canFlex* - フレキシブルな長距離通信タイミングがアダプタによってサポートされているかどうかを確認。 *canHyperdrive* - ハイパードライブ通信速度がアダプタによってサポートされているかどうかを確認。 *canOverdrive* - オーバドライブ通信速度がアダプタによってサポートされているかどうかを確認。 *canProgram* - 12V EPROM プログラミング電圧がアダプタによってサポートされているかどうかを確認。 *dataBlock* - 1-Wire ネットに 1 データブロックを送受信。 *getBit* - 1-Wire ネットから 1 ビットを読み取る。 *getBlock* - オール 1 (0xFF)を送信して、1-Wire ネットから 1 ブロックを読み取る。 *getByte* - オール 1 (0xFF)を送信して、1-Wire ネットから 1 バイトを読み取る。 *getSpeed* - 現在の 1-Wire 通信速度を読み取る。 *putBit* - 1-Wire ネットに 1 ビットを書き込む。 *putByte* - 1-Wire ネットに 1 バイトを書き込み、エコーが正しいことを確認。 *reset* - 1-Wire ネットの全デバイスをリセット。 *setPowerDuration* - 電力供給時間を設定。 *setPowerNormal* - 電力供給を停止。 *setProgramPulseDuration* - プログラムパルス時間を設定。 *setSpeed* - 1-Wire 通信速度を設定。 *startBreak* - 1-Wire ネット上でブレーク(low)を開始。 *startPowerDelivery* - 電力供給を開始。 *startProgramPulse* – プログラムパルスを開始。

ネットワーク

com.dalsemi.onewire.adapter.DSPortAdapter

excludeFamily - 検索からファミリグループを除外。
findFirstDevice - コンテナを自動生成せずに 1-Wire ネット上の最初のデバイスを検出。
findNextDevice - コンテナを自動生成せずに 1-Wire ネット上の次のデバイスを検出。
getAllDeviceContainers - コンテナとともに 1-Wire ネット上のすべてのデバイスを検索および検出。
getDeviceContainer - 検出した「現行」デバイス用のデバイスコンテナを取得。
getFirstDeviceContainer - 最初のデバイスを検出し、そのコンテナを生成。
getNextDeviceContainer - 次のデバイスを検出し、そのコンテナを生成。
setNoResetSearch - 1-Wire ネット検索が 1-Wire リセットを送出しないように設定。
setSearchAllDevices - 1-Wire ネット検索が全デバイスを含めるように設定(アラーム専用デバイスは除外)。
setSearchOnlyAlarmingDevices - 1-Wire ネット検索がアラームデバイスのみを含めるように設定。
targetAllFamilies - 1-Wire ネット検索が全デバイスを含めるように設定(除外なし)。
targetFamily - 1-Wire ネット検索で特定のファミリグループを対象とする。

(**com.dalsemi.onewire.container.***でも同様)

isAlarming - デバイスがアラーム状態にあるかどうかを確認。
isPresent - デバイスが 1-Wire ネット上に存在するかどうかを確認。
select - デバイス固有の操作コマンドに対応できるように、1-Wire ネットデバイスを選択。

転送

com.dalsemi.onewire.container.MemoryBank

getBankDescription - メモリバンクのテキスト記述を返す。
getSize - バイトで表したメモリバンクのサイズを取得。
getStartPhysicalAddress - メモリバンクの開始物理アドレスを取得。
isGeneralPurposeMemory - メモリバンクが汎用(メモリ割当てでない)かどうかを確認。
isNonVolatile - メモリバンクが不揮発性かどうかを確認。
isReadOnly - メモリバンクが読取り専用かどうかを確認。
isReadWrite - メモリバンクが読取り/書込み可能かどうかを確認。
isWriteOnce - メモリバンクが EPROM のような追記型かどうかを確認。
needsPowerDelivery - このメモリバンクが書込みに電力供給を必要とするかどうかを確認。
needsProgramPulse - このメモリバンクが書込みにプログラムパルスを必要とするかどうかを確認。
read - 解釈なしにメモリバンクを読み取る(パケット構造なし)。
setWriteVerification - API が書込み後に追加検証を実行するように設定。
write - メモリバンクのローを書き込む(パケット構造なし)。

com.dalsemi.onewire.container.PagedMemoryBank

getExtraInfoDescription - このバンクに関連するその他情報の記述を取得。
getExtraInfoLength - 各ページのその他情報のバイトで表した長さを取得。
getMaxPacketDataLength - このメモリバンクの各ページに収まる、「パケット」構造に収納可能なデータの最大長を取得。
getNumberPages - このメモリバンクのページ数を取得。
getPageLength - このメモリバンク内のローページの、バイトで表した長さを取得。
hasExtraInfo - このメモリバンクが各ページに関連したその他情報を保有しているかどうかを確認。
hasPageAutoCRC - このメモリバンクのページがデバイスによって提供される CRC 検証を備えているかどうかを確認。
readPage - メモリバンクから 1 ページを読み取る。
readPageCRC - デバイスが生成する CRC を利用して、メモリバンクから 1 ページを読み取る。
readPagePacket - メモリバンクの 1 ページからパケット構造を読み取る。
writePagePacket - メモリバンクの 1 ページにパケット構造を書き込む。

com.dalsemi.onewire.container.OTPMemoryBank

canLockPage - メモリバンクのページは追加書き込みの防止が可能かどうかを確認。

canLockRedirectPage - 追加リダイレクト防止のため、メモリバンクのリダイレクト機能をロック可能かどうかを確認。
canRedirectPage - 追記型ページの更新方法として、メモリバンクはリダイレクトされたページを持てるかどうかを確認。
getRedirectedPage - ページのリダイレクト先のページ番号を取得。
isPageLocked - 追加書込みをすることができないようにページがロックされているかどうかを確認。
isRedirectPageLocked - 追加リダイレクトすることができないようにページがロックされているかどうかを確認。
lockPage - ページをロック。
lockRedirectPage - リダイレクトされないようにページをロック。
redirectPage - ページを新規ページにリダイレクト。これは追記型デバイスの更新に使用。

ファイル

com.dalsemi.onewire.utils.OWFile
java.io.File (JDKのバージョン 1.2 用)と同じメソッドで、以下の追加メソッドを含みます。
close - ファイルを閉じ、このファイルに関連するリソースを開放。
format - この OWFile に規定されたデバイスに関連する 1-Wire File System をフォーマット。
getFD - このファイルがデバイスと同期することができるように、ファイルの OWFileDescriptor を取得。
getFreeMemory - 1-Wire File System 内の利用可能な空きメモリを取得。
getLocalPage - 1-Wire File System のページからメモリバンクのローカルページのリファレンスを取得。
getMemoryBankForPage - 提供された 1-Wire File System のページの読取り/書込みに使用可能なメモリバンクインスタンスを取得。
getOneWireContainer - ファイルシステムを構成するコンテナを取得。
getPageList - ファイルを構成している 1-Wire File System のページのリストを取得。

com.dalsemi.onewire.utils.OWFileDescriptor
java.io.FileDescriptor (JDKのバージョン 1.2 用)と同じメソッド。

com.dalsemi.onewire.utils.OWFileOutputStream
java.io.FileOutputStream (JDKのバージョン 1.2 用)と同じメソッド。

com.dalsemi.onewire.utils.OWFileInputStream
java.io.FileInputStream (JDKのバージョン 1.2 用)と同じメソッド。

デバイス

com.dalsemi.onewire.container*
6 種類の「センサ」タイプのインタフェースを含めて、30 を超える各種デバイス固有のコンテナの実装。
ADContainer - アナログ-ディジタルコンバータ
ClockContainer - クロック
SwitchContainer - スイッチ
TemperatureContainer - 温度センサ
PotentiometerContainer - ディジタルポテンショメータ
HumidityContainer - 湿度センサ
MissionContainer - 温度と湿度ロガーのミッション用
OneWireSensor - 1-Wire センサ
PasswordContainer - パスワードで保護されたメモリ

com.dalsemi.onewire.application*
SHA および 1-Wire タグ付けユーティリティクラス

下記の**例 2** は、図 3 で概説した「API の使用法フロー」にしたがった OWAPI コード部分を紹介しています。例 1 の PD コード例と同様、ワークループを通過するごとに 1-Wire ネットワーク上の各デバイスが検出され、より高度なアプリケーションであれば、1 つのデバイスタイプだけ、あるいは以前の検索で発見されたデバイスを検出することができます。

例 2. OWAPI のコード例

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

1-Wire のタグ付け

1-Wireセンサが増加して多種多様になるにつれて、1-Wireネットワークの管理が次第に困難になります。たとえば、ADCのようなセンサは各種数値を測定することができるため、センサの機能を規定するためにセンサに「タグ」付けできることが重要になります。XMLによる 1-Wireのタグ付け機構が作成され、Java用の 1-Wire APIに実装されています。これらのタグは、アプリケーションが動的にロードし、センサを環境に適合させることができるようにします。詳細については、『アプリケーションノート 158：1-Wire Tagging with XML (XMLによる 1-Wireのタグ付け)』を参照してください。

インストール

図 8 に記載されている API 呼び出しを構成する必須モジュールのすべては、単一の jar ファイルである OneWireAPI.jar にあります。この単一のモジュールを適切な場所またはクラスパスに配置すると、API 全体が提供されます。ただし、以下のようなよく知られた 2 つの例外があります。すなわち、特定のプラットフォームにインストールされる必要があるネイティブまたは通信 API があるということ、あるいはプラットフォームにはサイズ制限があるため、API 全体を有効にすることは好ましくないということです。こうした 2 つの例外は、OWAPI キットで詳細に考察されます。

1-Wire .NET (OW.NET)の概要

あらゆる目的や用途に応えるため、ここで提供する.NETサポートは、MicrosoftのJ#言語だけでコンパイルされるJava用 1-Wire APIです。すべての.NET 1-Wireアプリケーションは、OneWire.NET.dllへの参照のみを必要とします。これは、C#、J#、およびVB.NETなどのような最新の.NET言語のサポートも含んでいます。1-Wire .NETの例については、iButtonのWebサイトで該当するSDKをダウンロードして参照してください。

1-Wire アプリケーションが正しく動作するためには、OneWire.NET.dll は、次の再配布可能なモジュールも PC にインストールされている必要があることを忘れないでください。

1. ネイティブな 1-Wireポートアダプタのデバイスドライバ。これらは 1-Wireドライバと呼ばれ、iButtonのWebサイトからダウンロードすることができます。
2. Microsoft .NETのフレームワーク
3. Visual J# .NET再配布可能モジュール

Pocket PC、パーソナルディジタルアシスタンス(PDA)、携帯電話、およびセットトップボックスなどのデバイス用としてCompact .NETフレームワークの 1-Wireが必要な場合には、OneWire.NET.dllの新しいリンク層(専用)バージョンが利用可能です。これは、すべてがC#で記述されています。現時点では、OneWireContainersは記述されていませんが、DSPortAdapterクラスは利用可能です。詳細については、1-Wireソフトウェアのディスカッションフォーラムを参照してください。

1-Wire COM (OWCOM)の概要

1-Wire Windows コンポーネントオブジェクトモデル(COM)のフレームワークは、Java 用 1-Wire API (OWAPI)の単なるラッパーです。32 ビットの Microsoft Windows オペレーティングシステム上で稼動するほぼすべてのプログラミング言語は、COM オブジェクトを使用することができます。これは OWAPI を利用しているため、ダラスセミコンダクタの 1-Wire デバイスの極めて豊富なサポートクラスのコレクションは、C++ 、VisualBasic、JavaScript、JScript、Perl、および VBScript などのさまざまなプログラミング環境で利用することができます。

Java と COM は完全互換ではないため、既存の Java コードベースを COM インタフェースで機能させるために、複数の解決策が講じられています。こうした解決策は、次のセクションで概説されています。

OWCOM (およびTMEX)を含むSoftware Development Kitは、iButtonのWebサイトにあります。OWCOMを生成するJavaソースコードは、自由なPD形式のライセンスに基づいて提供されています。未提供のソースは、各種 1-Wireアダプタのサポート用に呼び出されるTMEX APIドライバ用のみです(TMEX APIのセクションを参照)。ただし、こうしたドライバは無制限に再配布可能です。

注:OWCOM の適正な動作は、Microsoft Java 仮想マシン(MSJVM)の有無に依存しています。Microsoft は、MSJVM をサポートしなくなりました。

ただし、MSJVM は次のリンクからダウンロードすることができます。

- http://www.linktivity.com/get_java.asp
- http://www.big-foot.de/dateien/msjava.exe

OWAPI と OWCOM の相違点

全インタフェースの基本構造は OWAPI を使用して実際に実装されているため、そのドキュメントは同じです。ただし、Java のドキュメントを COM のインタフェースに適用する際の経験則がいくつかあります。

1) Java は 64 ビット long を使用し、COM で使用のインタフェースは 32 ビットより大きな数をサポートしません。パラメータとして long を取得したり、long を返したりする Java のどの機能についても、文字列が使用されます。ルーズなタイプのプログラミング言語(VisualBasic や JavaScript など)では、long と文字列間の変換は自動的に行われ、おそらくコードには影響をまったく及ぼしません。その他すべての言語については、パラメータが文字列に変換され、戻り値が 64 ビット long に変換されるように特別な注意を払う必要があります。

2) 特定のプログラミング言語(特に厳密なチェック方式に依存している言語)で、さまざまなコレクションを取り扱うことは困難な場合があります。たとえば、VBは、java.util.Enumeration型のオブジェクトのランタイムプロパティを自動的に検出することができますが、VC++では不可能です。さらに、一部のコレクション(配列型など)は、COMをサポートしている各言語で同様には取り扱われません。これを回避するため、ヘルパー付のOWCOMラッパーオブジェクトにおける各配列型と列挙はリサイジング用に機能し、またエレメントあるいは列挙コンポーネントの取得/設定が行われるようにしています。

ByteArray、CharArray、IntArray、LongArray、DoubleArray、およびFloatArrayなどの配列型が追加されています。これらの各ヘルパーオブジェクトは、同一のテンプレートにしたがっています。

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

  // returns the primite value at the given index
  value 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();
}
```

OneWireContainerEnumeration、MemoryBankEnumeration、OWPathEnumeration、およびStringEnumerationの列挙型が追加されています。これらの各列挙ヘルパーオブジェクトは、以下に示すテンプレートに適合しています。

```
TypeEnumeration
{
  // Returns true if there are more elements left to enumerate
  bool hasMoreElements();

  // returns the next object in the enumeration
  value getNextElement();
}
```

3) Java では、過負荷メソッドはパラメータ数だけでなく、パラメータのタイプが異なるメソッドに対応しています。COM では、パラメータ数のみが、同じ名前の 2 つのメソッドを識別することができます。この必要条件に相反するすべてのメソッドは、同一パターンに収められます。こうしたメソッドは、文字列、long、またはバイト配列のいずれかを要求しています。文字列を要求するメソッドは手を付けられず、long を要求するメソッドは、末尾に「FromLong」を付けた新たな名前を持ちます。バイト配列を要求するメソッドは「FromBytes」を末尾に付加します。例を以下に示します。

Java	**COM**
isPresent(String) -->	isPresent(String)
isPresent(long) -->	isPresentFromLong(String) //By rule #1 and #3
isPresent(byte[ ]) -->	isPresentFromBytes(ByteArray) //By rule #2 and #3

4) OWAPI の一部のコンテナは、その機能のポリモフィズム（多様性）に依存しています。基本的に、ポリモフィズムとは、現実に親クラスのインスタンスであるかのように処理されるクラスの機能です。COM では、継承をサポートしないため、まったく同じようにポリモフィズムが顕在化することはありません。当社のアーキテクチャの各 COM コンポーネントは OWAPI のオブジェクトの実インスタンスを示していますが、通常、実際にはハイレベルのコンポーネント(MemoryBank など)はローレベルのオブジェクト(PagedMemoryBank や OTPMemoryBank など)のインスタンスを持っています。Java では、オブジェクトを単純に特定タイプに割り当てます。

残念ながら、この処理は COM ではそれほど容易ではないため、ヘルパーメソッドが OneWireContainer、MemoryBank、および DSPortAdapter などの 3 つの最上位コンテナに付加されました。このヘルパーメソッドのインタフェースは単純で、以下のとおりです。

```
Component getMostSpecificComponent();
```

戻り値は、どのクラスが呼び出されているかに依存しています。たとえば、Thermochron オブジェクト(OneWireContainer21)を実際持っており、getMostSpecificComponent を呼び出す OneWireContainer のインスタンスは、OneWireContainer21 COM インタフェースのインスタンスを返します。

5) ポリモフィズムによって喪失した別の機能は、特定の 1-Wire デバイスが特定のインタフェースをサポートしているかどうかをどのように検出するかということです。ポリモフィズムは、クラスをあたかもクラスを実装しているインタフェースの実際のインスタンスのように取り扱うようにもできます。たとえば OneWireContainer21 は、インタフェース TemperatureContainer を実装します。つまり、OneWireContainer21 は TemperatureContainer(「readTemperature」など)で一般的な全メソッドを実装します。

   Java では、「instanceof」のキーワードを使って、1-Wire オブジェクトのインスタンスを特定のインタフェースと比較するのが一般的です。「myOneWireDevice instanceof TemperatureContainer」ステートメントが true を返した場合、デバイスは温度読取りに必要なすべてのメソッドを実装しています。COM のソリューションは、すべての OneWireContainer で指定されたこれらのメソッドを使用することです。

```
boolean isTemperatureContainer();
boolean isADContainer();
boolean isSwitchContainer();
boolean isClockContainer();
boolean isPotentiometerContainer();
boolean isHumidityContainer();[NOT YET IMPLEMENTED]
```

6) java.util.Calendar オブジェクトを返す Java の機能は、ここでは、グリニッジ標準時 1970 年 1 月 1 日からの現在時刻をミリ秒単位で返します。これは、時刻を表す場合の UNIX の標準です。

7) このインタフェースは Java 形式のパッケージにはありませんが、まったく同一レベルで現在提供されています。たとえば、OneWireContainer21 はもはやパッケージ com.dalsemi.onewire.container.OneWireContainer21 にはなく、単に OneWireContainer21 として COM パッケージから入手することができます。

例 3 は、図 3 で概説した「API の使用法フロー」にしたがった OWCOM コード部分を紹介しています。今回も同様、ワークループを通過するごとに 1-Wire ネットワーク上の各デバイスが検出されます。より高度なアプリケーションであれば、1 つのデバイスタイプだけを検出したり、以前の検索で検出されたデバイスを選択したりすることができます。

例 3. OWCOM「Java Script」コードの例

```
boolean doing_work=true;

// get the 1-Wire access provider
var access = WScript.CreateObject("owapi.OneWireAccessProvider");

// get the default adapter
var adapter = access.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
   owd = owd_enum.nextElement();

   // do SOMETHING with device found (TRANSPORT/FILE/DEVICE)
   // . . .
  }

  // end exclusive use of adapter (SESSION)
  adapter.endExclusive();

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

インストール

I OWCOMライブラリおよび必要なドライバは、1-Wireドライバのインストールパッケージでインストールすることができるようになりました。1-Wireドライバは、iButtonのWebサイトからダウンロードすることができます。

1-Wireドライバのカスタムインストールを作成する方法の詳細については、『アプリケーションノート 1740：White Paper 6：1-Wire Drivers Installation Guide for Windows (白書 6：1-WireドライバのWindows用インストールガイド)』を参照してください。

TMEX API (TMEX)の概要

TMEX API は、メモリデバイスの限定された 1-Wire File Structure サポートを含めて、全 1-Wire デバイスに基本機能を提供する言語非依存の Windows 32 ビット DLL セットです。この API は、同じ 1-Wire ポートまたは異なる 1-Wire ポートで全面的に競合する多重プロセスのマルチスレッドアプリケーションで機能するように設計されています。この API は、それぞれ 16 個の別個のポートとともに、最大 16 種類の 1-Wire アダプタをサポートすることができます。これはダラスセミコンダクタ製のすべての 1-Wire アダプタをサポートします。

この API の最低レベル(最低レベルのデバイス-ドライバ層)は、Win32 プラットフォーム上で Java 用 1-Wire API 用の「ネイティブ」ドライバとして使用します(1-Wire ドライバとして知られています)。1-Wire .NET API と 1-Wire COM API は、Java 用の 1-Wire API にも基づいているため、Win32 プラットフォーム上でも使用することができます。**図 9** は、TMEX API が提供する

Win32 ネイティブサポートの利点をその他の API が利用する方法を図示しています。実ドライバファイル名およびその階層化方法がこの図に紹介されています。

1-Wireドライバのインストールパッケージ(TMEX API、OW.NET、およびOWCOMライブラリを含む)およびTMEX (および付随するAPI)にリンクされた多くのプログラム例を含むSoftware Developer's Kitのどちらも、iButtonのWebサイトで入手可能です。

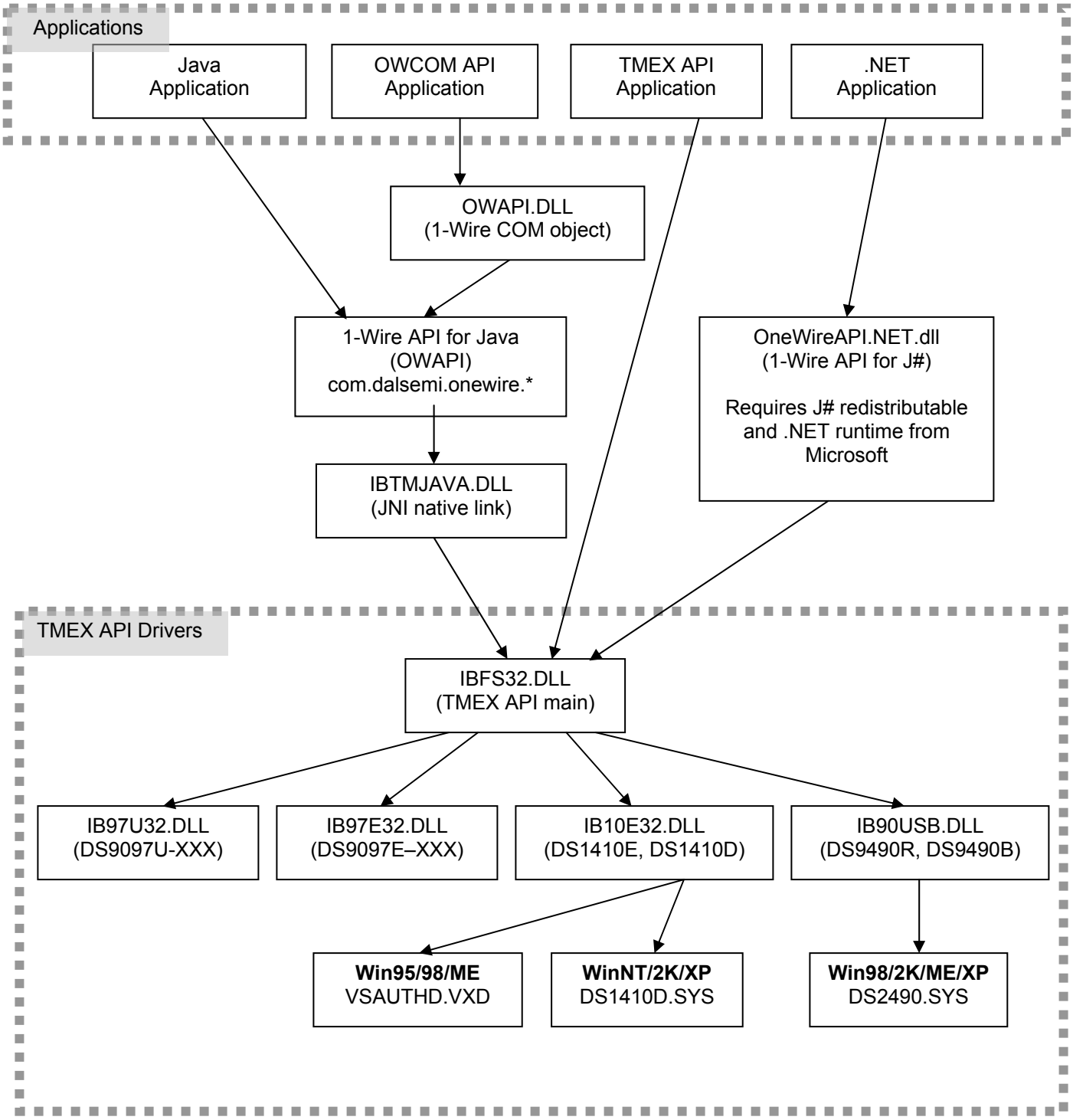
TMEX SDK で提供されているすべての例はソースコードを備えていますが、ローレベルのドライバのソースは現在提供されていません。ただし、こうしたドライバは無制限に再配布可能です。

表 6 は、各アダプタの機能とサポート対象の Windows プラットフォームとともに、現在サポート中の 1-Wire アダプタを紹介しています。

表 6. サポート対象の TMEX アダプタ

Adapter	Port	Features	Win95	Win98	WinME	WinNT	Win2K	WinXP
DS9490R, DS9490B	USB	Power delivery Overdrive RJ-11 or iButton holder DS2401 ID	No USB stack available	X	X	No USB stack available	X	X
DS1410E	Parallel	Power delivery Overdrive Dual iButton holder DS2401 ID	X	X	X	X	X	X
DS1410D	Parallel legacy	Dual iButton holder DS2401 ID	X	X	X	X	X	X
DS9097U-009	Serial	Power delivery Overdrive RJ-11 connector DS2502 ID	X	X	X	X	X	X
DS9097U-S09	Serial	Power delivery Overdrive RJ-11 connector	X	X	X	X	X	X
DS9097U-E25	Serial	Power delivery Overdrive RJ-11 connector EPROM write	X	X	X	X	X	X
DS1411	Serial	Power delivery Overdrive Single iButton holder	X	X	X	X	X	X
DS9097E	Serial legacy	RJ-11 connector EPROM write	X	X	X	X	X	X
DS9097	Serial legacy	RJ-11 connector	X	X	X	X	X	X
DS1413	Serial legacy	Single iButton holder	X	X	X	X	X	X

図 9. TMEX API ドライバおよび他の API の接続性



* 基本操作に最低限必要な機能

図 10 は、TMEX API 提供の機能を紹介しています。この API は非メモリデバイス固有の機能を提供していないことに注意してください。

図 10. TMEX API 機能

セッション
TMEndSession - 1-Wire ネットを放棄。 *TMExtendedStartSession* - 1-Wire ネットの排他的使用を要求。 *TMValidSession* - 現行の 1-Wire ネットセッションが有効かどうかを確認。
リンク
TMClose - 開いているポートのリソースを開放(常時使用可能ではない)。 *TMOneWireCom* - 1-Wire ネットの速度を Normal (16k ビット)または Overdrive (142k ビット)に設定。 *TMOneWireLevel* - 1-Wire ネットラインレベルを Normal (5V の弱プルアップ)、Power Delivery (5V の強プルアップ)、または Program Level (12V の EPROM プログラミングレベル)に設定。 *TMProgramPulse* - EPROM プログラミング用に、時間設定されたプログラミングパルスを 1-Wire ネットに送信。 *TMSetup* - ポートおよびアダプタが機能しているかどうかを確認し、検証。 *TMTouchBit* - 1-Wire ネットで 1 ビットを送受信。 *TMTouchByte* - 1-Wire ネットで 1 バイトを送受信。 *TMTouchReset* - 1-Wire ネット上の全デバイスをリセットし、結果を返す。
ネットワーク
TMAccess - 現行デバイスを選択し、デバイス固有のコマンドに対して準備をさせる。 *TMAutoOverDrive* - デバイスが自動的にオーバドライブ速度の切り替えができるようにドライバを設定。 *TMFamilySearchSetup* - 次の検索(TMNext または TMNextAlarm)で指定されたファミリタイプを検出するために現行の検索状態を設定。 *TMFirst* - 1-Wire ネット上の「最初の」1-Wire デバイスを検出するために検索。 *TMFirstAlarm* - 1-Wire ネット上の「最初の」警報を発生している 1-Wire デバイスを検出するために検索。 *TMNext* - 1-Wire ネット上の「次の」1-Wire デバイスを検出するために検索。 *TMNextAlarm* - 1-Wire ネット上の「次の」警報を発生している 1-Wire デバイスを検出するために検索。 *TMOverAccess* - 現行デバイスを選択し、Overdrive 速度に設定。 *TMRom* - 現在選択中のデバイスのシリアル番号(ROM 番号)を回収、または設定。 *TMSkipFamily* - 最後の検索で検出されたすべてのファミリタイプをキップ。 *TMStrongAccess* - 現行デバイスを選択し、現行デバイスが存在していることを検証。 *TMStrongAlarmAccess* - 現行デバイスを選択し、現行デバイスが存在して警報を出していないかを検証。
転送
TMBlockIO - 1-Wire リセットに先行して 1-Wire ネットに 1 データブロックを送受信。 *TMBlockStream* - 1-Wire リセットなしで 1-Wire ネットに 1 データブロックを送受信。 *TMExtendedReadPage* - デバイス生成 CRC 検査付きのメモリバンクの 1 ページ全体を読み取る(全デバイスタイプに該当せず)。 *TMProgramByte* - EPROM ベースの 1-Wire デバイスに 1 バイトをプログラミング。 *TMReadPacket* - ページからUniversal Data Packetを読み取る(Universal Data Packet 構造の説明については、アプリケーションノート 114を参照)。 *TMWritePacket* - ページに Universal Data Packet を書き込む。
ファイル
TMAttribute - ファイルまたはディレクトリの属性を変更。 *TMChangeDirectory* - 現在の作業ディレクトリの読取りや変更を行う。 *TMCloseFile* - ファイルハンドルを解放するために開いたファイルや作成したファイルを閉じる。 *TMCreateFile* - 書込み用にファイルを作成。 *TMCreateProgramJob* - EPROM プログラミング保留ジョブのロギング用書込みバッファを作成。 *TMDeleteFile* - ファイルを削除。 *TMDirectoryMR* - サブディレクトリを作成または削除。 *TMDoProgramJob* - EPROM プログラミング保留ジョブを書き込む。 *TMFirstFile* - 現在のディレクトリの最初のファイルを検出。 *TMFormat* - 1-Wire ファイル構造ファイルシステムをフォーマット。 *TMNextFile* - 現在のディレクトリ内の次のファイルを検出。

TMOpenFile - 読取り用にファイルを開く。 *TMReadFile* - 開いたファイルを読み取る。 *TMReNameFile* - ファイルやディレクトリの名前を変更。 *TMTerminateAddFile* - 「AddFile」を終了。「AddFile」は、上書きせずに追加可能な EPROM デバイス上の特殊ファイルタイプです。 *TMWriteAddFile* - EPROM デバイス上に「AddFile」を追加、または変更。 *TMWriteFile* - 作成されたファイルに書き込む。
デバイス
なし
その他
TMGetTypeVersion - アダプタドライバ用のバージョン情報を取得。 *Get_Version* - 全ドライババージョンを取得。

例 4 は、図 3 で概説した「API の使用法フロー」にしたがった TMEX コード部分を紹介しています。先の 3 つの例と同様、簡単なプロセスはワークループを通過するごとに 1-Wire ネットワーク上の各デバイスを検出するようにしており、より高度なアプリケーションであれば、1 つのデバイスタイプだけを検出したり、以前の検索で検出されたデバイスを選択したりすることができます。

例 4. TMEX「C」コードの例

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

インストール

TMEX APIは、前述の 1-Wireドライバのインストールパッケージ(オプションでOWCOM APIとOW.NET APIライブラリをインストール)でインストールすることができます。これはMicrosoftのインストーラパッケージであり、サポート対象のあらゆる 1-Wireアダプタについて Windowsのドライバとレジストリキーのすべてをロードします。ドライバとAPIファイル(インストールなし)は、カスタムインストール用に自由にオンラインで入手することが可能です。1-Wireドライバがインストールされれば、副次的に

OneWireViewerをインストールすることができます。OneWireViewerは、ほとんどの 1-Wireデバイスとi̲Buttonを調べて表示するデモプログラムです。

1-WireドライバとOneWireViewerデモの追加情報については、「ソフトウェアリソース」の下にあるiButtonのWebサイトにあります。

その他ツール

このドキュメントで取り上げた API は、1-Wire デバイスとの通信に利用可能なリソースの大部分を説明していますが、決して唯一のリソースではありません。このセクションでは、その他の利用可能なリソースを概説しています。

ソフトウェア認可 API

ソフトウェア認可とは、ハードウェアトークンの存在を必要とするソフトウェアアプリケーションの単なるロックです。この場合のトークンは、ユーザのワークステーションに接続されたi̲Buttonです。アプリケーションは、その実行前にi̲Buttonの存在と妥当性を照会します。これは、アプリケーションの実行中にも連続して問い合わせることができます。実際には、このドキュメントで概説されるどのAPIもこうしたアプリケーション用に使用することができますが、考慮すべき特定の問題があります。ユーザがドライバを交換し、ロックを無効にすることができるため、外部ドライバのセキュリティが脅かされます。1-Wire PDキット上で構築されたソフトウェア認可APIは、特にこうしたアプリケーション用に開発され、外部ドライバの代わりにリンク可能なモジュールを備えています。

ソフトウェア認可 API の設計は、最も簡単に使用することができるという点に重点を置いており、ソフトウェアプログラマの既存のコードに簡単に統合することができるようにしています。次の 3 つの主要なサービスが API によって提供されています。

- 1-Wire デバイスの初期化
- デバイスの認証
- デバイスの消去(これによってシステムの有効な要素ではなくなる)

1-Wireソフトウェア認可キットの情報は、i̲ButtonのWebサイトにあります。

検索エンジンのソフトウェア例

ダラスセミコンダクタは、この解説書で述べたAPIを実証する 1-Wireのソフトウェアアプリケーション例を継続的に開発しています。ソフトウェア開発キット(SDK)に含まれていない例は、オンラインのソフトウェア例の検索エンジンを利用して見つけることができます。これらの例の多くは (ソースコードを含めて) 、関連するSDKに追加されるまえに最初に検索エンジンに(個別のダウンロードとして)追加されます。検索エンジンは、以下の項目別にソフトウェア例を検索することができます。

- 1-Wireデバイス(Thermochron、SHA i̲Buttonなど)
- プラットフォーム(Win32、Linux、TINI など)
- API (TMEX、1-Wire パブリックドメイン、1-Wire .NET など)
- プログラミング言語(C、Java、C#、Visual Basic、Delphi など)

検索結果の内容は、プログラムの説明と該当するダウンロードのリンクです。

1-Wire ソフトウェア開発関係者グループ

1-Wireの開発者は「1-Wire Software Developer's Forum」に参加することをお勧めします。このグループの目的は、開発するアプリケーション、ツール、および 1-Wire/i̲Button系製品の使用法について調査、検討、および質疑応答を行うことです。ダラスセミコンダクタのアプリケーション技術者は、このグループ提起の質問に答えるためにこのフォーラムを管理しています。APIの更新およびキットのダウンロードに関する発表もこのグループに通知されます。

サードパーティ

多くのサードパーティ製のソフトウェアは 1-Wireデバイス用に利用することができます。その一部は、ソリューションとともに購入可能な、ダラスセミコンダクタ認定のソリューション開発者(ASD)が作成したアプリケーションです。「Source Forge」などのパブリックフォーラムに関するオープンソースプロジェクトはもちろんのこと、ASDおよびソリューションロケータのリストー式は、iButtonのサイトにあります。

終わりに

この解説書は、すべての 1-Wire API の特徴に関する概要を紹介してきました。また、サポートプラットフォームやプログラミング言語などを含めて、数種類の API についても詳述してきました。各デバイスタイプの API の適用範囲を紹介する早見表によって、使用する API の選択が容易になりました。適切な API を入手すると、1-Wire 利用のアプリケーションを容易に設計することができます。