



## APPLICATION NOTE 1100

# White Paper 5: Using 1-Wire APIs for Data Sheet Commands

*Abstract: All 1-Wire® device data sheets describe two sets of commands. The first set referred to as ROM Function Commands are used for device identification and selection. The second set is often called Memory Function Commands but may contain other non-memory operations. A ROM Function Command must be completed each time a device is selected to get it ready for a Memory Function Command. The 1-Wire APIs created by Dallas Semiconductor utilize these commands to do operations with 1-Wire devices. Sometimes it is not always obvious what commands are being called. This document will map the commands presented in the data sheets to the API functions. Where specific API functions are not available, a technique will be presented to translate the commands using the generic communications API functions.*

## Introduction

All 1-Wire device data sheets describe two sets of commands. The first set referred to as *ROM Function Commands* are used for device identification and selection. The second set is often called *Memory Function Commands* but may contain other non-memory operations. A *ROM Function Command* must be completed each time a device is selected to get it ready for a *Memory Function Command*. The 1-Wire APIs created by Dallas Semiconductor utilize these commands to do operations with 1-Wire devices. Sometimes it is not always obvious what commands are being called. This document will map the commands presented in the data sheets to the API functions. Where specific API functions are not available, a technique will be presented to translate the commands using the generic communications API functions.

See [Application Note 155](#) for a detailed description of the various APIs including the abbreviations discussed in this document (PD, TMEX, OWAPI, OWCOM). (*Special terms, commands, or codes are shown in italics for clarity.*)

## ROM Function Commands

The *ROM Function Commands* are used to either discover the ROM ID or use the ROM ID to select a device in various modes. The ROM ID is a unique 64-bit number that contains a family code, serialization field, and a cyclic redundancy check (CRC). The 1-Wire master transmits one of these functions after it has issued a 1-Wire reset and received a presence.

The *Read ROM* command reads the ROM ID directly. It can only be used on a 1-Wire network where there is only one device attached. With networks of more than one device, the ROM ID must be discovered with the *Search ROM* command. This search algorithm is discussed in detail in [Application Note 187](#). The *Conditional Search ROM* command works the same as *Search ROM* except only 1-Wire devices that are in some kind of alarm state will respond. This is used to discover only devices that need attention.

The *Skip ROM* command is used to select all devices regardless of its ROM ID. This could be used to gang program memory devices provided there is sufficient energy. The *Overdrive Skip* command is similar but it not only selects all devices it also puts those devices at the Overdrive communication rate. This is most often used to move all capable 1-Wire devices to Overdrive speed. After the devices are communicating in Overdrive, the ROM IDs can be discovered using the conventional *Search ROM sequence*.

The *Match ROM* command selects a specific device by broadcasting a selected ROM ID. The *Overdrive Match* is similar but it also switches the device to the Overdrive communication speed. The *Resume Command* is used to reselect the last device that was selected. This is a shortcut command when repeatedly accessing the same

device.

Table 1 maps the APIs to a particular *ROM Function Command*. Note that since the various 1-Wire APIs are designed with the idea of multiple 1-Wire devices on a network, the commands that require or are most useful in a single device network are not supported directly. However any command can be constructed using the basic communication functions as will be discussed later in *Custom Commands*.

**Table 1. ROM Function Commands**

Command	PD	TMEX
Read ROM	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Match ROM	owAccess	TMAccess
Search ROM	owFirst, owNext	TOMFirst, TMNext
Conditional Search ROM	owFirst, owNext	TMFirstAlarm, TMNextAlarm
Skip ROM	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Overdrive Skip*	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Overdrive Match	owOverdriveAccess	TMOverAccess
Resume Command**	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Command	OWAPI	OWCOM
Read ROM	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Match ROM	(package com.dalsemi.onewire.adapter) DSPortAdapter.select	DSPortAdapter.select
Search ROM	(package com.dalsemi.onewire.adapter) DSPortAdapter.getFirstDeviceContainer, DSPortAdapter.getNextDeviceContainer	DSPortAdapter.getFirstDeviceContainer, DSPortAdapter.getNextDeviceContainer
Conditional Search ROM	(package com.dalsemi.onewire.adapter) DSPortAdapter. setSearchOnlyAlarmingDevices (then same as Search ROM)	DSPortAdapter. setSearchOnlyAlarmingDevices (then same as Search ROM)
Skip ROM	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Overdrive Skip*	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.
Overdrive Match*	(package com.dalsemi.onewire.container) OneWireContainer.setSpeed OneWireContainer.doSpeed	OneWireContainer.setSpeed OneWireContainer.doSpeed
Resume Command**	No predefined API, see Custom Commands.	No predefined API, see Custom Commands.

\*Note: Only applies to 1-Wire devices that support Overdrive communication speed.

\*\*Note: Only applies to 1-Wire devices that support the Resume Command.

## Memory Function Commands

The *Memory Function Commands* vary slightly from one device type to another. However their primary objective is the same, which is to read and write the memory areas of the device. To deal with these diverse command structures, the 1-Wire APIs were constructed to abstract out these differences. For example, a generic write memory API may use a Write Scratchpad, Read Scratchpad, and Copy Scratchpad sequence or it may use an EPROM Write Memory sequence. To the API user, it looks the same.

There are three levels of memory commands in most of the APIs. The first allows reading and writing to the memory without any structure (raw). The second uses a packet structure called the Universal Data Packet (UDP). The third type combines multiple UDP structures into a file structure. See [Application Note 114](#) for a description of the UDP and file structure. Table 2 maps the APIs to the three types of memory operations.

**Table 2. Abstract Memory Functions**

Command	PD	TMEX
Write Raw	owWrite	TMProgramBlock (EPROM only) (see Custom Commands)
Read Raw	owRead	TMProgramBlock (EPROM only) (see Custom Commands)
Write UDP	owWritePagePacket	TMWritePacket
Read UDP	owReadPagePacket	TMReadPacket
Write File	owCreateFile owWriteFile	TMCreateFile TMWriteFile
Read File	owOpenFile owReadFile	TMOpenFile TMReadFile

Command	OWAPI	OWCOM
Write Raw	(package com.dalsemi.onewire.container) MemoryBank.write	MemoryBank.write
Read Raw	MemoryBank.read	MemoryBank.read
Write UDP	(package com.dalsemi.onewire.container) PagedMemoryBank.writePagePacket	PagedMemoryBank.writePagePacket
Read UDP	(package com.dalsemi.onewire.container) PagedMemoryBank.readPagePacket	PagedMemoryBank.readPagePacket
Write File	(package com.dalsemi.onewire.application.file) OWFileOutputStream.write	OWFileOutputStream.write
Read File	(package com.dalsemi.onewire.application.file) OWFileInputStream.read	OWFileInputStream.read

Some of the commands that are included under *Memory Function Commands* in data sheets are actually custom device commands. See the following section for a guide on how to deal with these commands.

## Custom Commands

Almost all of the custom commands can be derived by first selecting the device with a *Match ROM* equivalent API and then send a bidirectional block of data to the 1-Wire network. The block is constructed by putting in the write commands that are required into the block and putting in FF (hex) bytes into the block that are reads from the 1-Wire device. For example, the DS1994 has a memory-mapped real-time clock (RTC) register that can be accessed with the *Read Memory* command. **Figure 1** below is taken from the DS1994's data sheet.

The DS1994 is no longer recommended for new designs.

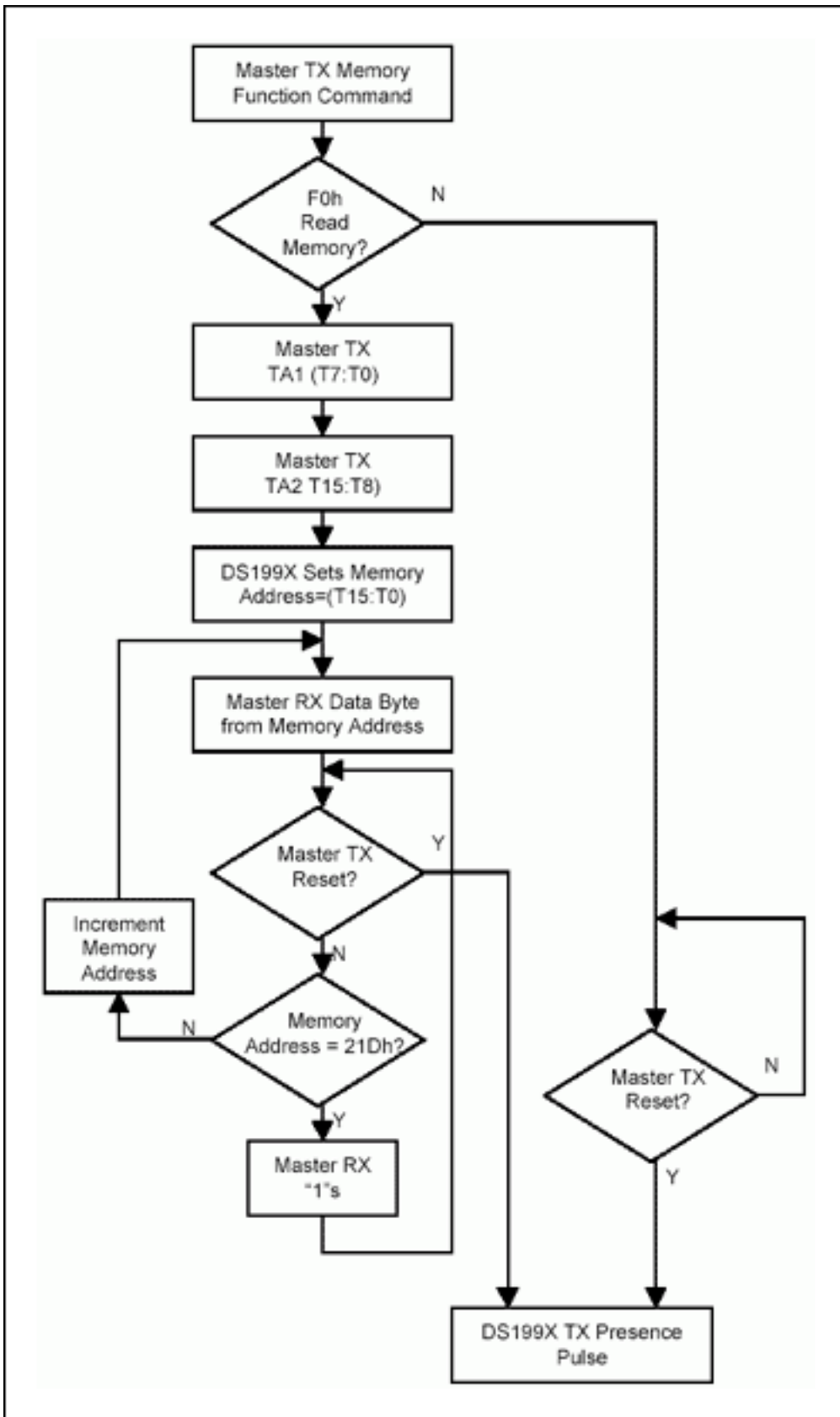


Figure 1. DS1994 read memory flow (data sheet).

As the DS1994 data sheet specifies, the RTC register is five bytes long starting at address 0202(hex). The memory command flow as seen in Figure 1 starts after the device has been selected with a *ROM Function Command* like *Match ROM*.

Table 3 below lists the eight bytes that make up a block of bidirectional data to be sent to the 1-Wire bus based on the flow chart. While this data could be sent a byte at a time, it is often more efficient to create a block and

sent it all at once.

**Table 3. Read RTC Block**

Block Offset	Byte Value (hex)	Description
0	F0	Master TX Read Memory command
1	02	Master TX TA1 (address, least significant byte, T7:T0)
2	02	Master TX TA2 (address, most significant byte, T15:T8)
3	FF	Master RX byte 0 of RTC (address 0202h)
4	FF	Master RX byte 1 of RTC (address 0203h)
5	FF	Master RX byte 2 of RTC (address 0204h)
6	FF	Master RX byte 3 of RTC (address 0205h)
7	FF	Master RX byte 4 of RTC (address 0206h)

**Figure 2** is a 'C' example written for the 1-Wire Public Domain (PD) API that uses the block outlined in Table 3 to read the RTC of the DS1994.

## Figure 2. PD Example Reading RTC

```
unsigned char datablock[] = { 0xF0,0x02,0x02,0xFF,0xFF,0xFF,0xFF,0xFF };
int portnum=0;

// select the current device (Match ROM)
if (owAccess(portnum))
{
    // send the read memory command and address, receive the RTC value
    if (owBlock(portnum, 1, datablock, 8))
    {
        // RTC is now in bytes 3-7 of datablock
        ...
    }
}
```

Table 4 shows the bidirectional block commands for each of the APIs. For completeness, the single byte and bit commands are also included.

**Table 4. Generic 1-Wire IO Functions**

Command	PD	TMEX
block (bidirectional)	owBlock	TMBlockStream
byte (bidirectional)	owTouchByte	TMTouchByte
bit (bidirectional)	owTouchBit	TMTouchBit
read byte	owReadByte	TMTouchByte(data = FF hex)
write byte	owWriteByte	TMTouchByte(data to write)
reset	owTouchReset	TMTouchReset
Command	OWAPI	OWCOM
block (bidirectional)	(package com.dalsemi.onewire.adapter) DSPortAdapter.dataBlock	DSPortAdapter.dataBlock
byte (bidirectional)	(package com.dalsemi.onewire.adapter) DSPortAdapter.dataBlock (single byte block)	DSPortAdapter.dataBlock (single byte block)
bit (bidirectional)	Not available	Not available
read byte	(package com.dalsemi.onewire.adapter) DSPortAdapter.getBytes	DSPortAdapter.getBytes
write byte	(package com.dalsemi.onewire.adapter) DSPortAdapter.putByte	DSPortAdapter.putByte
1-Wire reset + presence detect	(package com.dalsemi.onewire.adapter) DSPortAdapter.reset	DSPortAdapter.reset

By looking at the data sheet for each 1-Wire device type, it is possible to do any of the functions by constructing the appropriate block and using the generic 1-Wire IO functions. Some of the 1-Wire devices require special power delivery constraints or program pulses that are addressed by the special API functions in Table 5.

**Table 5. Special 1-Wire Power Functions**

Command	PD	TMEX
EPROM programming pulse	owProgramPulse	TMProgramPulse
Power delivery (strong pullup) after bit	owReadBitPower (read bit only)	TMOneWireLevel (prime for next bit) TMTouchBit
Power delivery (strong pullup) after byte	owWriteBytePower (write byte only)	TMOneWireLevel (prime for next byte) TMTouchByte
Command	OWAPI	OWCOM
EPROM programming pulse	(package com.dalsemi.onewire.adapter) DSPortAdapter.startProgramPulse	DSPortAdapter.startProgramPulse
Power delivery (strong pullup) after bit	(package com.dalsemi.onewire.adapter) DSPortAdapter.startPowerDelivery (prime for next bit) DSPortAdapter.putBit / DSPortAdapter.getBit	DSPortAdapter.startPowerDelivery (prime for next bit) DSPortAdapter.putBit / DSPortAdapter.getBit
Power delivery (strong pullup) after byte	(package com.dalsemi.onewire.adapter) DSPortAdapter.startPowerDelivery (prime for next byte) DSPortAdapter.putByte / DSPortAdapter.getBytes	DSPortAdapter.startPowerDelivery (prime for next byte) DSPortAdapter.putByte / DSPortAdapter.getBytes

1-Wire is a registered trademark of Dallas Semiconductor Corp.

Dallas Semiconductor is a wholly owned subsidiary of Maxim Integrated Products, Inc.

---

Application Note 1100: [www.maxim-ic.com/an1100](http://www.maxim-ic.com/an1100)

### More Information

For technical questions and support: [www.maxim-ic.com/support](http://www.maxim-ic.com/support)

For samples: [www.maxim-ic.com/samples](http://www.maxim-ic.com/samples)

Other questions and comments: [www.maxim-ic.com/contact](http://www.maxim-ic.com/contact)

---

### Keep Me Informed

Preview new application notes in your areas of interest as soon as they are published. Subscribe to [EE-Mail - Application Notes](#) for weekly updates.

---

### Related Parts

DS1822: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS1822-PAR: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS18B20: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS18B20-PAR: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS18S20: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS18S20-PAR: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1904: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1920: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1921G: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1963S: [QuickView](#)  
DS1971: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1973: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1982: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1985: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1990A: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1992: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS1993: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS1995: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS1996: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)  
DS2401: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2405: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2406: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2415: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2417: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2431: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2432: [QuickView](#) -- [Abridged Data Sheet](#)  
DS2433: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS2438: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2450: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2502: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2505: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2506: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)  
DS2760: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

AN1100, AN 1100, APP1100, Appnote1100, Appnote 1100

Copyright © by Maxim Integrated Products

Additional legal notices: [www.maxim-ic.com/legal](http://www.maxim-ic.com/legal)