1-Wire

From Wikipedia, the free encyclopedia

1-Wire is a device communications bus system designed by Dallas Semiconductor that provides low-speed (16.3 kbit/s[1]) data, signaling, and power over a single conductor.

1-Wire is similar in concept to I²C, but with lower data rates and longer range. It is typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments. A network of 1-Wire devices with an associated master device is called a **MicroLAN**. The protocol is also used in small electronic keys known as a **Dallas key** or **iButton**.

One distinctive feature of the bus is the possibility of using only two wires — data and ground. To accomplish this, 1-Wire devices include an 800 pF capacitor to store charge and power the device during periods when the data line is active.

An iButton in a plastic fob, as used for Istanbul Akbil smart ticket

Usage example [edit]

1-Wire devices are available in different packages: integrated circuits, a TO-92-style transistor, and a portable form called an iButton or Dallas key which is a small stainless-steel package that resembles a watch battery. Manufacturers also produce devices more complex than a single component that use the 1-Wire bus to communicate.

A Java ring with an embedded iButton

1-Wire devices can fit in different places in a system. It might be one of many components on a circuit board within a product. It also might be a single component within a device such as a temperature probe. It could be attached to a device being monitored. Some laboratory systems connect to 1-Wire devices using cables with modular connectors or CAT-5 cable. In such systems, RJ11 (6P2C or 6P4C modular plugs, commonly used for telephones) are popular.

Systems of sensors and actuators can be built by wiring together many 1-Wire components. Each 1-Wire component contains all of the logic needed to operate on the 1-Wire bus. Examples include temperature loggers, timers, voltage and current sensors, battery monitors, and memory. These can be connected to a PC using a bus converter. USB, RS-232 serial, and parallel port interfaces are popular solutions for connecting a MicroLan to the host PC. 1-Wire devices can also be interfaced directly to microcontrollers from various vendors.

iButtons are connected to 1-Wire bus systems by means of sockets with contacts that touch the "lid" and "base" of the canister. Alternatively, the connection can be semi-permanent with a socket into which the iButton clips, but from which it is easily removed.

Each 1-Wire chip has a unique identifier code. This feature makes the chips, especially iButtons, suitable electronic keys. Some uses include locks, burglar alarms, computer systems, manufacturer-approved accessories, time clocks and courrier and maintenance keys for smart safes. iButtons

have been used as Akbil smart tickets for the public transport in Istanbul.

Power supplies [edit]

Apple MagSafe and MagSafe 2 connector-equipped power supplies, displays, and Mac laptops use the 1-Wire protocol to send and receive data to and from the connected Mac laptop, via the middle pin of the connector. Data include power supply model, wattage, and serial number; and laptop commands to send full power, and illuminate the red or green light-emitting diodes in the connector. [2]

Genuine Dell laptop power supplies use the 1-Wire protocol to send data via the third wire to the laptop computer about power, current and voltage ratings. The laptop will then refuse charging if the adapter does not meet requirements.[3]

Communication protocol [edit]

In any MicroLan, there is always one master in overall charge, which may be a personal computer or a microcontroller . The master initiates activity on the bus, simplifying the avoidance of collisions on the bus. Protocols are built into the master's software to detect collisions. After a collision, the master retries the required communication.

A 1-Wire network is a single open drain wire with a single pull-up resistor . The pull-up resistor pulls the wire up to 3 or 5 volts. The master device and all the slaves each have a single open-drain connection to drive the wire, and a way to sense the state of the wire. Despite the "1-Wire" name, all devices must also have a second wire, a ground connection to permit a return current to flow through the data wire.[4] Communication occurs when a master or slave briefly pulls the bus low, *i.e.*, connects the pull-up resistor to ground through its output MOSFET. The data wire is high when idle, and so it can also power a limited number of slave devices. Data rates of 16.3 kbit/s can be achieved. There is also an overdrive mode that speeds up the communication by a factor of 10.

A short 1-Wire bus can be driven from a single digital I/O pin on a microcontroller. A universal asynchronous receiver-transmitter (UART) can also be used.[5] Specific 1-Wire driver and bridge chips are available. Universal Serial Bus "bridge" chips are also available. Bridge chips are particularly useful to drive cables longer than 100 m. Up to 300-meter twisted pairs , *i.e.*, telephone cables, have been tested by the manufacturer. These extreme lengths require adjustments to the pull-up resistances from 5 to 1 kΩ.

The master starts a transmission with a *reset* pulse, which pulls the wire to 0 volts for at least 480 µs. This resets every slave device on the bus. After that, any slave device, if present, shows that it exists with a "presence" pulse: it holds the bus low for at least 60 µs after the master releases the bus.

To send a binary number "1", the bus master sends a very brief (1–15 µs) low pulse. To send a binary number "0", the master sends a 60 µs low pulse. The falling (negative) edge of the pulse is used to start a monostable multivibrator in the slave device. The multivibrator in the slave reads the data line about 30 µs after the falling edge. The slave's internal timer is an inexpensive analog timer. It has analog tolerances that affect its timing accuracy. Therefore, the pulses are calculated to be within margins. Therefore, the "0" pulses have to be 60 µs long, and the "1" pulses can't be longer than 15 µs.

When receiving data, the master sends a 1–15-µs 0-volt pulse to start each bit. If the transmitting slave unit wants to send a "1", it does nothing, and the bus goes to the pulled-up voltage. If the transmitting slave wants to send a "0", it pulls the data line to ground for 60 µs.

The basic sequence is a reset pulse followed by an 8-bit command, and then data are sent or received in groups of 8 bits.

When a sequence of data is being transferred, errors can be detected with an 8-bit CRC (weak data protection).

Many devices can share the same bus. Each device on the bus has a 64-bit serial number, of which 8 bits are used as a checksum, thus allowing a "universe" of 2^{56} (over 7.2×10^{16}) unique device identities. The least significant byte of the serial number is an 8-bit number that tells the type of the device. The most significant byte is a standard (for the 1-Wire bus) 8-bit CRC.[6]

There are several standard broadcast commands, as well as commands used to address a particular device. The master can send a selection command, then the address of a particular device. The next command is executed only by the addressed device.

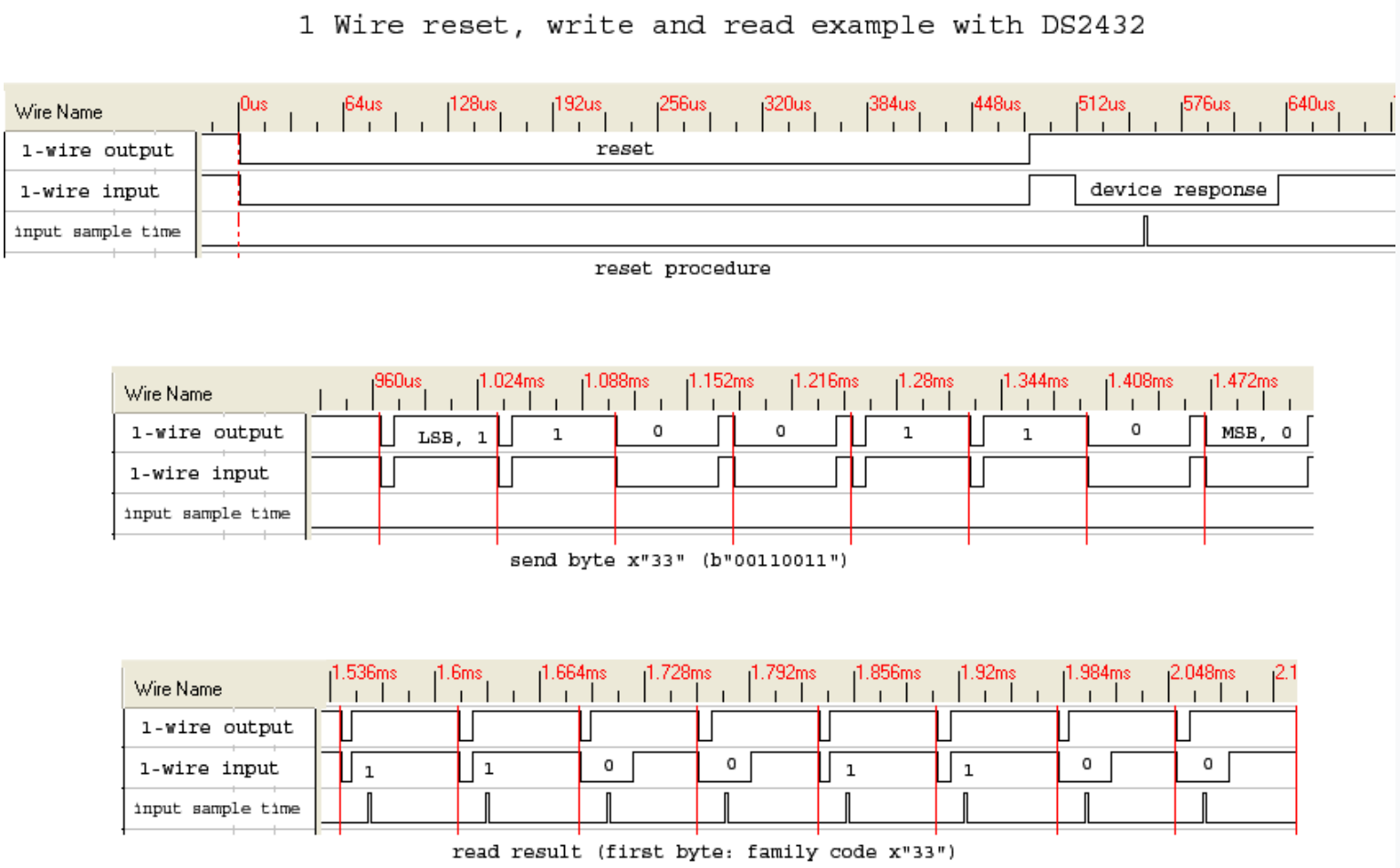
The 1-Wire bus enumeration protocol, like other singulation protocols, is an algorithm the master uses to read the address of every device on the bus. Since the address includes the device type and a CRC, recovering the roster of addresses also produces a reliable inventory of the devices on the bus. To find the devices, the master broadcasts an enumeration command, and then an address, "listening" after each bit of an address. If a slave's address matches all the address bits sent so far, it returns a 0. The master uses this simple behavior to search systematically for valid sequences of address bits. The process is much faster than a brute force search of all possible 56-bit numbers, because as soon as an invalid bit is

detected, all subsequent address bits are known to be invalid. The 56-bit address space is searched as a binary tree, allowing up to 75 devices to be found per second. The order in which device addresses are discovered by this enumeration protocol is deterministic and depends only on the device type and serial number. Bit-reversing these 56 bits yields the order of discovery for devices using Maxim's published algorithm (algorithm defined in Application Note 187[7]). The search algorithm can be implemented in an alternative form, initially searching paths with address bits equal to 1, rather than 0. In this case, inverting the 56 address bits and then reversing them yields the order of discovery.

The location of devices on the bus is sometimes significant. For these situations, a microcontroller can use several pins, or the manufacturer has a 1-Wire device that can switch the bus off or pass it on. Software can therefore explore sequential *bus domains*.[6]

Example communication with a device [edit]

The following signals were generated by an FPGA , which was the master for the communication with a DS2432 (EEPROM) chip, and measured with a logic analyzer. A logic high on the 1-Wire output, means the output of the FPGA is in tri-state mode and the 1-Wire device can pull the bus low. A low means the FPGA pulls down the bus. The 1-Wire input is the measured bus signal. On input sample time high, the FPGA samples the input for detecting the device response and receiving bits.



Development tools [edit]

When developing and/or troubleshooting the 1-Wire bus, examination of hardware signals can be very important. Logic analyzers and bus analyzers are tools that collect, analyze, decode, and store signals to simplify viewing the high-speed waveforms.

See also [edit]

- SDI-12 , a single data wire communications scheme
- Single-wire transmission line , a technique for electric power transmission with only "1 wire" without a ground return wire path

Touch memory

References [edit]

1. ^ "Reading and Writing 1-Wire Devices Through Serial Interfaces" . *Maxim Integrated*. Retrieved 2022-12-21.
2. ^ ""Teardown and exploration of Apple's Magsafe connector"" . rightTo.com. Retrieved 2017-07-18.
3. ^ "Hacking Dell Laptop Charger Identification" . hackaday.com. Retrieved 2015-11-30.
4. ^ "1-Wire online tutorial. This tutorial will give you an overview of the 1-Wire protocol, its device operation and application solutions" . Archived from the original on 2009-05-02. Retrieved 2009-03-13.
5. ^ "Using a UART to Implement a 1-Wire Bus Master" .
6. ^ *a b* "iButton Overview" (PDF). Archived from the original (PDF) on 27 January 2009. Retrieved 18 December 2008. 081218 maxim-ic.com
7. ^ "1 Wire Search Algorithm (Application Note 187)" (PDF). Retrieved 2 October 2020.

External links [edit]

- 1-Wire Device
- Accessing, Reading, and Writing to 1-Wire devices using a UART
- Using a UART to Implement a 1-Wire Bus Master
- iButton , iButtonLink
- Guidelines for Reliable Long Line 1-Wire Networks
- Choosing the Right 1-Wire Master for Embedded Applications
- OWFS — 1-Wire file system for Linux
- Guides to working with 1-Wire, for programmers and engineers
- Getting 1-Wire sensors working in Linux using OWFS
- 1-wire Arduino tutorial
- Guide to writing software for 1-Wire/ MicroLan using Lazarus , "the free Delphi".

v · t · e **Technical and *de facto* standards for wired computer buses** [show]

v · t · e **Automation protocols** [show]

Category : Serial buses