1-Wire

From Wikipedia, the free encyclopedia

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

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**.

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 to power the device during periods when the data line is active.

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

A **Java Ring** with embedded iButton

Contents

- 1 Usage example
  - 1.1 Dell power source
- 2 Communication protocol
- 3 Example communication with a device
- 4 Development tools
- 5 See also
- 6 References
- 7 External links

Usage example

Dependent on function, native 1-Wire devices are available as single components in integrated circuit and TO-92 packaging, and in some cases a portable form called an *iButton* that resembles a watch battery. Manufacturers also produce devices more complex than a single component that use the 1-Wire bus to communicate.

1-Wire devices may be one of many components on a circuit board within a product, may be a single component within a device such as a temperature probe, or may be attached to a device being monitored. Some laboratory systems and other data acquisition and control systems connect to 1-Wire devices using cables with modular connectors or with CAT-5 cable, with the devices themselves mounted in a socket, incorporated in a small PCB, or attached to the object being monitored. 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 1-Wire components. Each 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 the MicroLan to the host PC. 1-Wire devices can also be interfaced directly to microcontrollers from various vendors.

The **iButton** (also known as the **Dallas Key**) is a mechanical packaging standard that places a 1-Wire component inside a small stainless steel "button" similar to a disk-shaped watch battery. 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.

The **Java Ring**, a ring-mounted iButton with a Java virtual machine compatible with the Java Card 2.0 specification within, was given to attendees of the 1998 JavaOne conference.[2]

Each 1-Wire chip has a unique identifier code. This feature makes the chips, especially in an iButton package, suitable for use as a key to open a lock, arm and deactivate burglar alarms, authenticate computer system users, operate time clock systems, etc. iButtons have been used as Akbil smart tickets for the public transport in Istanbul.

Dell power source

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

Communication protocol

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

Many devices can share the same bus. Each device on the bus has a unique 64-bit serial number. 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.[4]

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 (described later), 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 address roster also produces a reliable inventory of the devices on the bus. The 64-bit address space is searched as a binary tree, allowing up to 75 devices to be found per second.

The Dallas 1-Wire network is physically implemented as an open drain master device connected to one or more open drain slaves.[5] A single pull-up resistor is common to all devices and acts to pull the bus up to 3 or 5 volts, and may provide power to the slave devices. Communication occurs when a master or slave asserts the bus low, i.e. connects the pull up resistor to ground through its output MOSFET. Specific 1-Wire driver and bridge chips are also available. Data rates of 16.3 kbit/s can be achieved. There is also an overdrive mode which speeds up the communication by a factor of 10.

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 "1", the bus master sends a very brief (1–15 µs) low pulse. To send a "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 clocks to read the data line about 30 µs after the falling edge. The slave's multivibrator unavoidably has analog tolerances that affect its timing accuracy, which is why the "0" pulses have to be 60 µs long, and the "1" pulses can't be longer than 15 µs.

When a dedicated 1-Wire interface peripheral is not available, a UART can be used to implement a 1-wire bus master.[6] Serial or USB "bridge" chips are also available that handle the timing and waveform requirements of the 1-Wire bus, and are particularly useful in utilizing long (greater than 100 m) cables effectively. Up to 300 meter long buses consisting of simple twisted pair telephone cable have been tested by the manufacturer. It will however require adjustment of pull-up resistances from 5 to 1 kΩ.

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 is sent or received in groups of 8-bits.

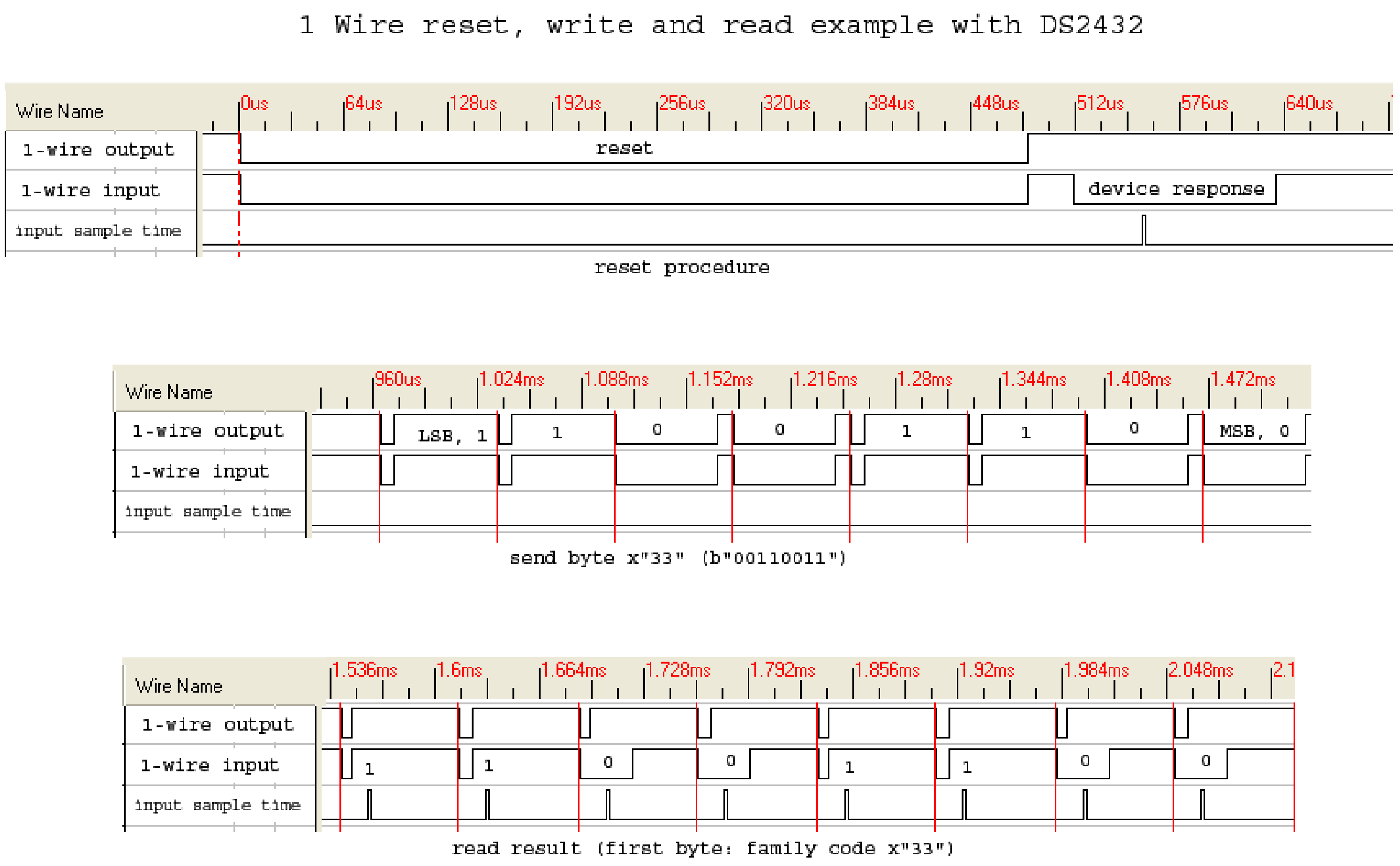
When a sequence of data is being transferred, errors can be detected with an 8-bit CRC (weak data protection).

To find the devices, the master broadcasts an enumeration command, and then an address, "listening" after each bit of an address. If a slave has all the address bits 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 64-bit numbers because as soon as an invalid bit is detected, all subsequent address bits are known to be invalid. An enumeration of 10 to 15 devices finishes very quickly.

The location of devices on the bus is sometimes significant. For these situations, the manufacturer has a special device that either passes through the bus or switches it off. Software can therefore explore sequential *bus domains*.[4]

Example communication with a device

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

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

See also

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

References

1. *1-Wire Products (http://www.maxim-ic.com/design_guides/en/1_WIRE_PRODUCTS_4.pdf)*
2. *An introduction to the Java Ring (http://www.javaworld.com/javaworld/jw-04-1998/jw-04-javadev.html?page=1)*, by Stephen M. Curry, JavaWorld.com, April 1st, 1998.
3. "Hacking Dell Laptop Charger Identification". hackaday.com. Retrieved 2015-11-30.

4. "iButton Overview" (PDF). Archived from the original (PDF) on 27 January 2009. Retrieved 18 December 2008. 081218 maxim-ic.com
5. "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 2009-04-25. Retrieved 2009-03-13.
6. "Using a UART to Implement a 1-Wire Bus Master".

External links

- 1-Wire Device (https://www.maximintegrated.com/en/products/digital/one-wire.html)
- Accessing, Reading, and Writing to 1-Wire devices using a UART (http://www.spider-e.com/wp/?p=231)
- Using a UART to Implement a 1-Wire Bus Master (http://www.maximintegrated.com/en/app-notes/index.mvp/id/214)
- iButton (http://www.maxim-ic.com/products/ibutton/index.cfm?CMP=ELK2), iButtonLink (http://www.ibuttonlink.com)
- Guidelines for Reliable Long Line 1-Wire Networks (http://www.maxim-ic.com/appnotes.cfm/an_pk/148/CMP/ELK5)
- Choosing the Right 1-Wire Master for Embedded Applications (http://www.maxim-ic.com/appnotes.cfm/an_pk/4206/CMP/ELK6)
- OWFS - 1-wire filesystem for Linux (http://www.owfs.org/)
- Guides to working with 1-Wire, for programmers and engineers (http://sheepdogguides.com/dst1main.htm)
- Getting 1-wire sensors working in Linux using OWFS (http://www.technotes.se/?p=26)
- 1-Wire Arduino tutorial (http://www.hacktronics.com/Tutorials/arduino-1-wire-tutorial.html)
- Guide to writing software for 1-Wire/ MicroLan (http://sheepdogguides.com/lut/dstl2hello_ds18b20.htm) using Lazarus, "the free Delphi".

Retrieved from "https://en.wikipedia.org/w/index.php?title=1-Wire&oldid=751189426"

Categories: Serial buses