

How to use 1-Wire bus in Linux

HARDWARE

frist you **need 1-wire master device.**

Master device may be :

Name	Bus	Native Linux driver	OWFS support	Example image
DS9490	USB	Yes	Yes	
DS9790E	SERIAL	No	Yes	
DS2480B	SERIAL	No	Yes	
DS2482	I2C	Yes	Yes	
General GPIO	GPIO (use any pin with linux control)	Yes	Yes	
And others...				

METHODS TO USE 1-WIRE DEVICES

In Linux is two current method to use 1-wire devices.

1. Native Linux kernel driver.

Main advantage is that consume less memory and is part of kernel (nothing to install).

Suitable for small systems. No every slave devices is supported. Slave devices is controlled by read or write into special files into directory /sys/bus/w1. Every devices has here own directory.

2. OWFS(OneWireFileSystem) project.

Is more often use, because support more slave devices, use cache and all is more complex than linux kernel drivers. But is need install separate software. Included more interfaces e.g. mount as file system or control from web, ftp and have interface for some program language e.g python, php, c,...

1. Native Linux kernel driver

Make work master device

First must be make work master device. This is some example by type master device:

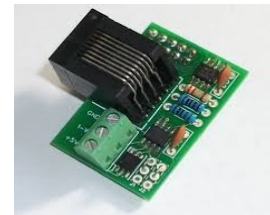
DS9490

This device has native support in linux just plug in. Linux automatically load these modules:
wire,w1_smem,w1_therm,ds2490



DS2482

This device has also native support in linux, but start is more difficult. Because on your system must have functional I2C bus on which DS2482 is connect. DS2482 exist in two variants DS2482-100 and DS2482-800. DS2482-100 have only one port of 1-Wire bus and DS2482-800 have eight ports of 1-Wire bus.



Make work I2C bus:

This example is for raspberryPI miniPC on other computers may be similar, but probably use another I2C driver (you must know what chip you use and how kernel module is needed load, or maybe your I2C bus is function automatically). Attention some computers maybe have more I2C bus, than you must find on which DS2482 is connect.

For raspberryPI:

1. Remove *i2c-bcm2708* kernel module from black list
In */etc/modprobe.d/raspi-blacklist.conf* comment this line (add "#" at the line start) ***#blacklist i2c-bcm2708***
2. Load modules *i2c-bcm2708* and *i2c-dev* needed for I2C bus (good is add this module to list in */etc/modules*)

```
Shell:~$ sudo modprobe i2c-bcm2708  
Shell:~$ sudo modprobe i2c-dev
```

3. Install I2C tools for check and control I2C devices from shell.

```
Shell:~$ sudo apt-get install i2c-tools
```

4. Check that something is "alive" at i2C bus and find I2C address of DS2482.
(Number 1 in next command is something like identification of I2C port my by different depends on version raspbeeryPI)

```
Shell:~$ sudo i2cdetect -y 1
```

this men I2C bus is function and chip have I2C address 18.

After I2C work

Load kernel module for DS2482 device (force=identification I2C port, chip I2C address)

```
Shell:~$ sudo modprobe ds2482  
Shell:~$ sudo sh -c "echo ds2482 0x18 >  
/sys/bus/i2c/devices/i2c-1/new_device"
```

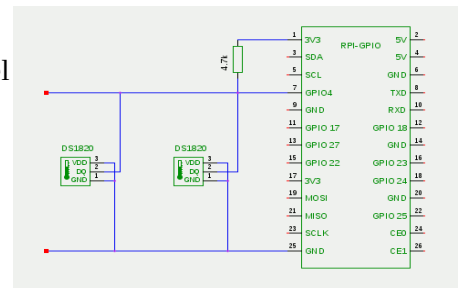
And load basic kernel module for 1-Wire bus

```
Shell:~$ sudo modprobe wire
```

This is all now is 1-Wire bus with ds2482 function.

General GPIO

This example is for raspberryPI. On other PC may be similar, but you must know witch pin use for 1-Wire and for this pin must be load kernel module as GPIO.



In raspberryPI for 1W is used pin GPIO4 and this pin must be connected cross 4,7KOhm resistor to 3,3V. Important is the parameter **pullup = 1**, which tells the module that a parasitic power supply via a pull-up resistor is present. Then load kernel module for *w1-gpio* .

```
Shell:~$ sudo modprobe w1-gpio pullup=1
```

More at: <https://github.com/raspberrypi/firmware/issues/348>

And load basic kernel module for 1-Wire bus

```
Shell:~$ sudo modprobe wire
```

How to use slaves

The *wire* modul create a subdirectory for each sensor found in */sys/bus/w1/devices* directory. Each sensor has family code:

3a is ds2413
28 is ds18b20

...

E.g.:

```
Shell:~$ cd /sys/bus/w1/devices  
Shell:~$ ls
```

If no load kernel module for specific 1-Wire slave device, than you may see directory for specific device contain only general files.

Example:

```
Shell:~$ cd /sys/bus/w1/devices/3a-00000002f140  
Shell:~$ ls
```

This is not for use.

But after load specific module for use device, than directory contain specific files (every device has

another files). Writing or reading these files you control 1-Wire device. When module *wire* start scan devices and try load specifick modules automaticaly later on no.

Some example as upper bat with load kernel module:

Now directory contain extra files state a output. By this files maybe device control.

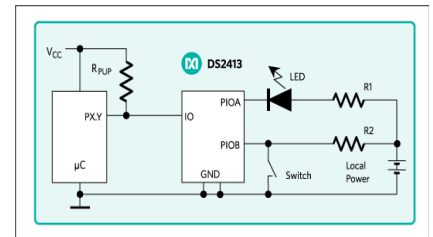
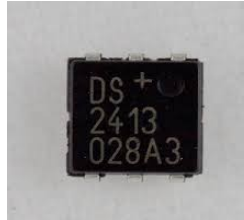
```
Shell:~$ sudo modprobe w1_ds2413  
Shell:~$ cd /sys/bus/w1/devices/3a-00000002f140  
Shell:~$ ls
```

If you can see which kernel modules is supported in your system, then list directory */lib/modules/XX-version-kernel/kernel/drivers/w1/slaves*.

List of slave devices and description how to control

DS2413

Is small device contain two pins names as A,B with open collector. Pins can be used as logical output or input. Often use as switch or binary input.



Load kernel module:

```
Shell:~$ sudo modprobe w1_ds2413
```

List device directory:

```
Shell:~$ cd /sys/bus/w1/devices/3a-00000002f140
Shell:~$ ls
```

file **output** – control device (write only 1B with binary number).

Bit map PIO:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	PIO B	PIO A

(n.u = not used)

This mean:

write 0	PIO A = 0	PIO B = 0
write 1	PIO A = 1	PIO B = 0
write 2	PIO A = 0	PIO B = 1
write 3	PIO A = 1	PIO B = 1 (in this case port can, be used as input)

file **state** - contain status of port A and B. Port A is mapped as bit 0 and port B as bit 1.

bit 2 and 3 is always 0. Bit 4-7 is inverse of bit 0-3, can by use for verify.

Shell example write:

This is more difficult in shell, because into output file must be write only 1B.

Set port A=0, B=0 (b=00000000 h=00)

```
Shell:~$ echo -e '\x00' |dd of=/sys/bus/w1/devices/3a-00000002f140/output
```

Set port A=1, B=0 (b=00000001 h=01)

```
Shell:~$ echo -e '\x01' |dd of=/sys/bus/w1/devices/3a-00000002f140/output
```

Set port A=0, B=1 (b=00000010 h=02)

```
Shell:~$ echo -e '\x02' |dd of=/sys/bus/w1/devices/3a-00000002f140/output
```


Set port A=1, B=1 (b=00000011 h=03)

Shell example read:

Read 1B from file *state*.

```
Shell:~$ dd if=/sys/bus/w1/devices/3a-00000002f140/state | hexdump -x
```

Important is number **0f**. This mean 00001111 binary => PIO A=1, PIO B=1 and inversion bits is OK read was successfully.

Python example:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import re, os, time, struct

# function: read and parse sensor data file
def read_sensor(path):
    try:
        f = open(path+"/state", "r")
        status = f.read(1)
        status=struct.unpack('B', status)[0]
        print "%s : %x" % (path, status)
        f.close()
    except (IOError), e:
        print time.strftime("%x %X"), "Error reading", path, ": ", e
        return

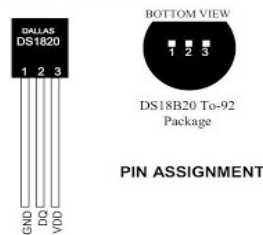
# function: read and parse sensor data file
def write_sensor(status,path):
    try:
        f = open(path+"/output", "w")
        status = f.write(struct.pack('B',status))
        f.close()
    except (IOError), e:
        print time.strftime("%x %X"), "Error reading", path, ": ", e
        return

# define pathes to 1-wire sensor
pathes = (
    "/sys/bus/w1/devices/3a-00000002f140",
    "/sys/bus/w1/devices/3a-00000002f265"
)

# read sensor data
for path in pathes:
    write_sensor(0,path)
    read_sensor(path)
    time.sleep(1)
    write_sensor(3,path)
    read_sensor(path)
    time.sleep(1)
```

DS18B20

Temperature sensor. Measures
Temperatures from -55°C to +125°C (-
67°F to +257°F).



No need load kernel, because *w1_therm* is automatically load with *wire* module.

List device directory:

```
Shell:~$ cd /sys/bus/w1/devices/28-000002da5fb3
Shell:~$ ls
```

Shell example:

Use ds18B20 is very easy only read file *w1_slave*.

```
Shell:~$ cat w1_slave
```

Its all temperature is $24937/1000 = 24,937^{\circ}\text{C}$

DS2408

Device with eight pin of PIO. Every pin can be use as input or output.
Often use as switch of relay array or buttons input.

Comment:

Me not workwith DS90C02 and ubuntu, but on rasoberry pi with DS2482 and rasbian work.



Load kernel module:

```
Shell:~$ sudo modprobe w1_ds2408
```

List device directory:

```
Shell:~$ cd /sys/bus/w1/devices/29-00000013b42f
Shell:~$ ls
```

file **output** – write to control device (write only 1B with binary number) or read last write value.
Bit map PIO:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PIO 8	PIO 7	PIO 6	PIO 5	PIO 4	PIO 3	PIO 2	PIO 1

If pin is set 1 then pin maybe used as input.

file **state** - contain status of PIO pin as input

files **activity**, **cond_search_mask**, **cond_search_polarity** and **status_control** affect reactinon if change status some of pins. More in documentation od DS2408 at
<http://datasheets.maximintegrated.com/en/ds/DS2408.pdf> .

Shell examples:

Write 1Byte=FF hex binary data (FF hex = 11111111 binary → set all PIO=1):

```
Shell:~$ cd /sys/bus/w1/devices/29-00000013b42f
Shell:~$ echo -e "\xff" > output
```

Read actual state:

```
Shell:~$ cd /sys/bus/w1/devices/29-00000013b42f
Shell:~$ cat state | hexdump -x
```

Important is number **fe**. This mean 11111110 binary => PIO 1=0, PIO 2 to PIO 8=1.

2. OWFS(OneWireFileSystem) project

OWFS is open source project for management 1-Wire bus and his slave devices. Its best choice for bigger systems. More info at www.owfs.org.

Install

On debian or ubuntu and rasbian just:

```
Shell:~$ sudo apt-get install owfs  
Shell:~$ sudo apt-get install owfs-doc  
Shell:~$ sudo apt-get install ow-shell
```

and create directory for mount 1-Wire file system e.g.:

```
Shell:~$ sudo mkdir /mnt/1wire
```

Configuration

Edit */etc/owfs.conf* file:

comment line (add # to beginning line):

```
#server: FAKE = DS18S20,DS2405
```

```
mountpoint = /mnt/1wire  
allow_other
```

Edit */etc/fuse.conf* file:

comment line:

```
user_allow_other
```

Next configuration is depends on master device you use.

For master device DS9490

First remove conflict with kernel modules

```
Shell:~$ sudo rmmod ds2490  
Shell:~$ sudo rmmod wire
```

Good idea is add these module to black list in */etc/modprobe.d/blacklist.conf*.

And edit */etc/owfs.conf* file:

comment line:

```
server: usb = all
```

For master device DS2480B OR DS9907E

Edit `/etc/owfs.conf` file:

uncomment line (`/dev/ttyS1` is used serial port may by another):

```
server: device = /dev/ttys1
```

For master device DS2482

For this device must by function I2C bus. Commissioning procedure is described above in chapter native linux driver DS2482.

If I2C bus is function and verify detect chip with I2C address 18 then can going configure OWFS.

Edit `/etc/owfs.conf` file:

edit line with device option (type correct I2C bus `/dev/i2c-0` or `/dev/i2c-1` or ..):

```
server: device = /dev/i2c-0
```

For all

Mount OWFS directory (`/mnt/1wire`):

```
Shell:~$ owfs
```

Good idea is add this command into init scripts for automatically start with system.

Or use init script

```
Shell:~$ /etc.init.d/owfs start
```

Now every sensor have our own directory with his options e.g.:

`/mnt/1wire/28.B35FDA020000` (temperature sensor)

`/mnt/1wire/3A.40F102000000` (dual channel switch)

control devices is very easy only read or write options representation as normal file.

Shell examples for slave DS2413 (dual channel switch):

Write 0 to port A

```
Shell:~$ echo "0" > /mnt/1wire/3A.40F102000000/PIO.A
```

Read from port A

```
Shell:~$ cat /mnt/1wire/3A.40F102000000/PIO.A
```

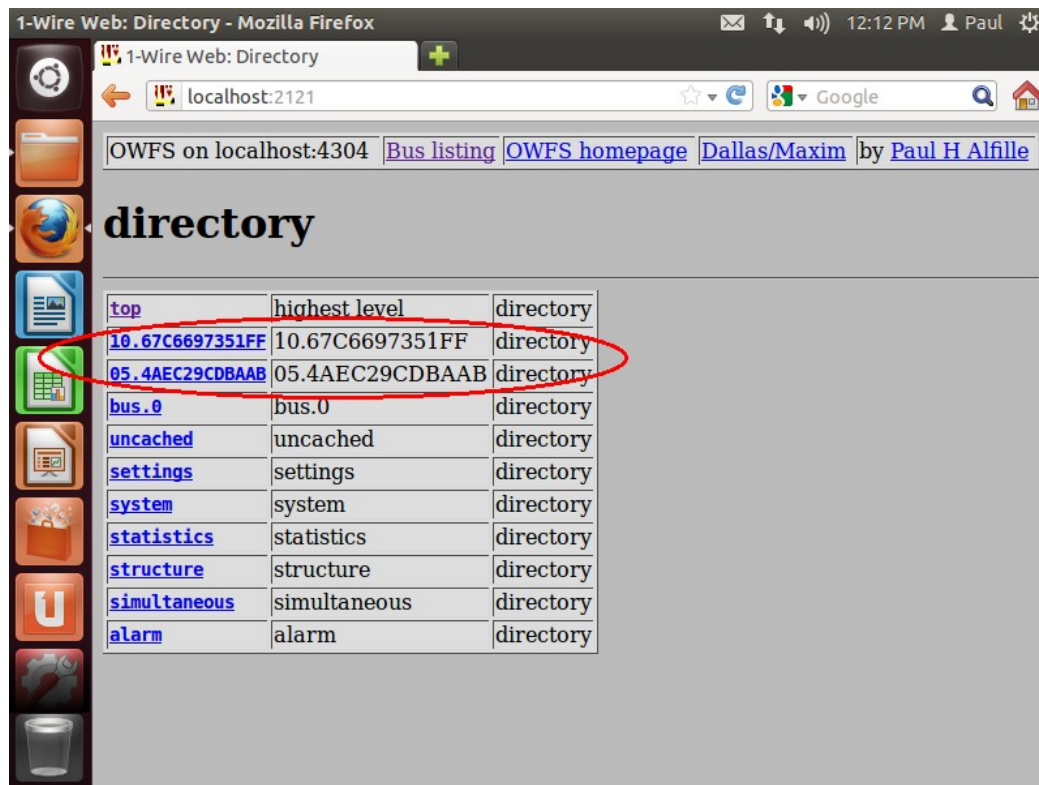
One of the next comfort options OWFS is run `owhttpd` (OWFS over web server).

Standard port is 2121 can you change in `/etc/owfs.conf`.

```
Shell:~$ /etc.init.d/owhttpd start
```

And open web browser at localhost:2121

Examples owhttpd:



1-Wire Web: Directory - Mozilla Firefox

1-Wire Web: Directory

localhost:2121

OWFS on localhost:4304 [Bus listing](#) [OWFS homepage](#) [Dallas/Maxim](#) by [Paul H Alfille](#)

directory

top	highest level	directory
10.67C6697351FF	10.67C6697351FF	directory
05.4AEC29CDBAAB	05.4AEC29CDBAAB	directory
bus.0	bus.0	directory
uncached	uncached	directory
settings	settings	directory
system	system	directory
statistics	statistics	directory
structure	structure	directory
simultaneous	simultaneous	directory
alarm	alarm	directory

Use easy prepared device

If you search easy use products for home automation look at www.seahu.cz.

SH017 (PiToDIN)

Is preinstalled PLC computer based on raspberry PI with OWFS.



You can easy add more other models and sensors.

DIN case.



Case mount on or into wall



OEM modules

