

# **W1: Dallas' 1-wire bus**

**David Fries** <David@Fries.net>

---

# **W1: Dallas' 1-wire bus**

by David Fries

Copyright © 2013

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For more details see the file COPYING in the source distribution of Linux.

---

## Table of Contents

1. W1 API internal to the kernel .....	1
W1 API internal to the kernel .....	1
drivers/w1/w1.h .....	1
drivers/w1/w1.c .....	8
drivers/w1/w1_family.h .....	10
drivers/w1/w1_family.c .....	12
drivers/w1/w1_int.c .....	14
drivers/w1/w1_netlink.h .....	16
drivers/w1/w1_io.c .....	21

---

# Chapter 1. W1 API internal to the kernel

## W1 API internal to the kernel

**drivers/w1/w1.h**

W1 core functions.







## Name

enum w1\_master\_flags — bitfields used in w1\_master.flags

## Synopsis

```
enum w1_master_flags {  
    W1_ABORT_SEARCH,  
    W1_WARN_MAX_COUNT  
};
```

## Constants

W1\_ABORT\_SEARCH    abort searching early on shutdown

W1\_WARN\_MAX\_COUNT    emit warning when the maximum count is reached











## Name

struct w1\_family\_ops — operations for a family type

## Synopsis

```
struct w1_family_ops {  
    int (* add_slave) (struct w1_slave *);  
    void (* remove_slave) (struct w1_slave *);  
    const struct attribute_group ** groups;  
};
```

## Members

add_slave	add_slave
remove_slave	remove_slave
groups	sysfs group



## Name

`wl_register_family` — register a device family driver

## Synopsis

```
int wl_register_family (struct wl_family * newf);
```

## Arguments

*newf* family to register

## Name

`w1_unregister_family` — unregister a device family driver

## Synopsis

```
void w1_unregister_family (struct w1_family * fent);
```

## Arguments

*fent* family to unregister

## drivers/w1/w1\_int.c

W1 internal initialization for master devices.

## Name

`w1_add_master_device` — registers a new master device

## Synopsis

```
int w1_add_master_device (struct w1_bus_master * master);
```

## Arguments

*master* master bus device to register



## Name

enum w1\_cn\_msg\_flags — bitfield flags for struct cn\_msg.flags

## Synopsis

```
enum w1_cn_msg_flags {  
    W1_CN_BUNDLE  
};
```

## Constants

W1_CN_BUNDLE	Request bundling replies into fewer messages. Be prepared to handle multiple struct cn_msg, struct w1_netlink_msg, and struct w1_netlink_cmd in one packet.
--------------	---









## Name

`w1_write_8` — Writes 8 bits.

## Synopsis

```
void w1_write_8 (struct w1_master * dev, u8 byte);
```

## Arguments

*dev*    the master device

*byte*   the byte to write

## Name

`w1_read_8` — Reads 8 bits.

## Synopsis

```
u8 w1_read_8 (struct w1_master * dev);
```

## Arguments

*dev* the master device

## Return

the byte read

























