Paket OW
Version 3.10.3

Klaus der Tiger, Karl M. Weckler und Roland Franke
E-Mail: fli4l@franke-prem.de

Das fli4l-Team
E-Mail: team@fli4l.de

26. Juli 2015

Inhaltsverzeichnis

1. Dokumentation des Paketes OW

1.1. OW – 1–Wire Bus

1.1.1. Einleitung

Dieses Paket installiert das OWFS (siehe Kapitel 1.1.1.2) und bietet so lesenden und schreibenden Zugriff auf einen an den fli4l kontaktierten 1–Wire Bus. Hierzu wird ein 1–Wire Busmaster an eine serielle Schnittstelle[1] oder einen USB Port[2] des PC angeschlossen. Darüber hinaus unterstützt das OPT auch I^2C Adapter und die Anbindung an OWServer. Näheres hierzu in den man-pages im Anhang (Kapitel: A.1.6). An die 1–Wire-seitige Buchse des entsprechenden Adapters wird dann der eigentliche 1–Wire Bus angelegt.

1.1.1.1. Hardware

1.1.1.1.1. Der 1–Wire Standard Der 1–Wire ® bzw. One-Wire oder Eindraht-Bus von Maxim (Maxim/Dallas) beschreibt eine serielle Schnittstelle, die mit einer Datenader auskommt und sowohl als Stromversorgung als auch als Sende-und Empfangsleitung genutzt wird. Gleichwohl ist jedoch eine „Rückleitung“ (GND) erforderlich. Jeder 1–Wire Chip hat eine unikale Identnummer über die er angesprochen wird. So können mehrere 1–Wire Geräte an einem einzelnen Bus betrieben werden.

1.1.1.1.2. Die 1–Wire-Bauteile Maxim bietet eine Vielzahl von 1–Wire-Bauteilen an, als da sind: Serielle-, USB-, I^2C-Adapter, Thermometer, Schalter (bis 8 Kanäle), EEPROMs, Uhren, A/D-Wandler, digitale Potentiometer. Man bekommt eigentlich alles, was man so für die Hausautomation braucht. Eine Übersicht über die wichtigsten Bauteile findet sich im Anhang unter A.1.2. An den Bus können auch iButton ® Bauteile (NV-RAM, EPROM, EEPROM, Temperatur, Feuchtigkeit, RTC, SHA, Logger) angeschlossen werden.

1.1.1.1.3. Der 1–Wire-Bus Der 1–Wire-Bus besteht im Prinzip aus zwei verdrillten Leitungen, wobei unter Beachtung entsprechender Topologien auch längere Strecken bis 150 m kein Problem sein sollten. Häufig wird für die Verkabelung normales Cat.5 Twisted Pair Ethernet-Kabel genommen. Zur Belegung der einzelnen Adern gibt es unterschiedliche Ansätze. Maxim benutzt 6-polige Modular-Buchsen und -Stecker (RJ-11) und hat einen eigenen Standard kreiert, der jedoch nicht zum 8-poligen RJ-45 Steckzeug passt. Weitere Standards sind im Anhang (Kapitel A.1.1) beschrieben. Auch zum Thema Topologie des Busses wird man bei Maxim fündig, wie überhaupt deren Webseite alles Nötige bietet, um gut mit 1–Wire zu recht zu kommen.

[1] DS9097U COM Port Adapter.

[2] DS9490R USB Bridge, auch zusammen mit DS1402D-DR8 (Blue Dot™) für iButton. Alle DS9490 Adapter basieren auf dem DS2490 USB–1–Wire-Baustein.

1.1.1.2. OWFS

OWFS steht für „One Wire File System“. Dabei handelt es sich um eine von Paul H. Alfille entwickelte Software die unter der GPL lizenziert ist. Auf Grundlage einer 1Wire- Protokoll verständigen Systembibliothek (OWLib) bildet OWFS den 1–Wire-Bus als Dateisystem ab. Darüber hinaus bietet das Programm noch weitere Implementierungen, wie owserver, owshell, owhttpd, owftpd, owtap und Sprachmodule für capi, perl, tcl, php, die jedoch in der hier vorliegenden Anpassung für fli4l nicht berücksichtigt wurden. Alles Weitere zu OWFS und viel Interessantes zu 1–Wire findet man auf:http://owfs.org/ und http://sourceforge.net/projects/owfs/.

1.1.1.3. Fuse

Fuse steht für „Filesystem in userspace“. Fuse ermöglicht die Implementierung eines voll funktionsfähigen Dateisystems im Userspace. Mit der Installation von OPT_OW wird das mit fli4l ausgelieferte Fuse automatisch als Kernelmodul beim Systemstart geladen. Alles weitere zu Fuse findet man auf: http://fuse.sourceforge.net/ und http://sourceforge.net/projects/fuse/.

1.1.1.4. libusb

Bei libusb handelt es sich um eine freie, GPL-lizensierte USB-Bibliothek. Diese wird benötigt, um über einen USB-Adapter auf den 1–Wire-Bus zuzugreifen. Alles weitere zu libusb findet man auf: http://libusb.sourceforge.net/

1.1.2. Lizenz

Dieses Programm ist durch die GNU General Public License, Version 2, Juni 1991, lizenziert und kann unter den angegebenen Bedingungen frei verwendet, vervielfältigt und verändert werden. Der Text der GNU General Public License ist im Internet veröffentlicht unter: http://www.gnu.org/licenses/gpl.txt Eine inoffizielle deutsche Übersetzung findet sich unter: http://www.gnu.de/gpl-ger.html

Diese Übersetzung soll nur zu einem besseren Verständnis der GPL verhelfen, rechtsverbindlich ist alleine die englischsprachige Version.

1.1.3. Gewährleistungs-und Haftungsausschluss

Die Veröffentlichung dieses Programms erfolgt mit dem Willen und in der Hoffnung, dass es von Nutzen sein wird. Dennoch wird jegliche Gewährleistung -auch die implizite Gewährleistung der Marktreife oder der Eignung für einen bestimmten Zweck abgelehnt. Details hierzu finden Sie in der GNU General Public License (GPL). Für Datenverlust, Schäden an Hard-oder Software oder sonstige Schäden wird keine Haftung übernommen.

1.1.4. Systemvoraussetzungen

Auf Grund der Größe des OPT_OW benötigt man eine Festplatte bzw. Flashkarte. Näheres dazu unter OPT_HD. Eine Installation auf FD ist nicht möglich.

Für die Anzeige im Browser ist der im fli4l-Paket „httpd“ enthaltene Mini-Webserver erforderlich. Weitere Hinweise hierzu unter Kapitel 1.1.7.1.

Zur Beachtung:

Die USB Ansteuerung über die W1-Kernelmodule klappt nach Auskunft von Paul Alfille, dem Maintainer von OWFS, noch nicht und wurde im Opt bisher nicht getestet. (Testhalber hatte ich (Roland Franke) die W1 Module mit der Version vom OWFS V2.8 p16 bzw. p19 bereits einmal am laufen, da jedoch die Anbindung und Auswertung völlig anders ist, als dies bei der Standardversion der Fall ist, werde ich dies vermutlich nicht weiter verfolgen)

Für die Verwendung der USB-Adapter muss im System die Angaben des üdevïm "rules.d"vorhanden sein. Nur wenn diese Einstellungen stimmen, funktioniert auch die Anbindung im Zusammenhang OWSERVER und OWFS.

Die Verwendung der beiden Programme *owshell* und *owhttpd* funktionierte im Testbetrieb auf einigen Hardwareumgebungen bisher nicht zufrieden stellend. Im Falle von Fehlern kann man versucehn mit setaillierter Beschreibung in den Newsgroups für fli4l zu posten.

1.1.5. Installation

Nach dem Entpacken des tar.gz-Archivs in das fli4l-Verzeichnis ist die Textdatei config/ow.txt zu bearbeiten. Für die Verwendung des Webinterface muss in config/ httpd.txt die Variable `OPT_HTTPD`='yes' gesetzt werden (siehe Kapitel 1.1.6.1). Wird RRDTool für die Aufzeichnung von Messwerten benötigt, so ist die Konfiguration der Textdatei config/rrdtool.txt erforderlich (siehe Kapitel 1.1.8.4).

1.1.6. Konfiguration

Beispielkonfiguration ohne Kommentare, nähere Erläuterungen weiter unten:

```
OPT_OW='yes'                        # install OPT_OW (yes/no)
OW_USER_SCRIPT=''                   # e.g. 'usr/local/bin/ow-user-script.sh'

OW_OWFS='yes'                       # start owfs (yes/no)
OW_OWFS_DEV='usb'                   # usb*, ttyS*, ip:port, etc.
OW_OWFS_GROUP_N='4'                             # number of groups
OW_OWFS_GROUP_1_NAME='1--Wire an USB'           # name of first group
OW_OWFS_GROUP_1_PORT_N='2'                      # number of ports of device
OW_OWFS_GROUP_1_PORT_1_ID='81.70D42A000000/ID'      # ID of device
OW_OWFS_GROUP_1_PORT_1_ALIAS='ID'                   # alias of ID
OW_OWFS_GROUP_1_PORT_2_ID='81.70D42A000000/Admin/*' # admin-access
OW_OWFS_GROUP_1_PORT_2_ALIAS='Admin/'               # alias of admin

OW_OWFS_GROUP_2_NAME='Heizung'
OW_OWFS_GROUP_2_PORT_N='7'
OW_OWFS_GROUP_2_PORT_1_ID='3A.F6E401000000/PA'
OW_OWFS_GROUP_2_PORT_1_ALIAS='1. Umwälzpumpe'
OW_OWFS_GROUP_2_PORT_2_ID='3A.F6E401000000/PB'
OW_OWFS_GROUP_2_PORT_2_ALIAS='2. Ladepumpe'
OW_OWFS_GROUP_2_PORT_3_ID='10.651BA9010800/temp'
OW_OWFS_GROUP_2_PORT_3_ALIAS='4. Rücklauftemperatur'
OW_OWFS_GROUP_2_PORT_4_ID='10.DEF0A8010800/temp'
OW_OWFS_GROUP_2_PORT_4_ALIAS='3. Vorlauftemperatur'
OW_OWFS_GROUP_2_PORT_5_ID='3A.F6E401000000/Admin/*'
OW_OWFS_GROUP_2_PORT_5_ALIAS='Admin/Switch-'
OW_OWFS_GROUP_2_PORT_6_ID='10.DEF0A8010800/Admin/*'
OW_OWFS_GROUP_2_PORT_6_ALIAS='Admin/VLT-'
```

```
    OW_OWFS_GROUP_2_PORT_7_ID='10.651BA9010800/Admin/*'
    OW_OWFS_GROUP_2_PORT_7_ALIAS='Admin/RLT-'

    OW_OWFS_GROUP_3_NAME='Solaranlage'
    OW_OWFS_GROUP_3_PORT_N='3'
    OW_OWFS_GROUP_3_PORT_1_ID='1C.7F6CF7040000/P0'
    OW_OWFS_GROUP_3_PORT_1_ALIAS='1. Ladepumpe'
    OW_OWFS_GROUP_3_PORT_2_ID='1C.7F6CF7040000/P1'
    OW_OWFS_GROUP_3_PORT_2_ALIAS='2. Ventil'
    OW_OWFS_GROUP_3_PORT_3_ID='1C.7F6CF7040000/Admin/*'
    OW_OWFS_GROUP_3_PORT_3_ALIAS='Admin/Switch-'

OW_OWSHELL='yes'
OW_OWSHELL_RUN='yes'
OW_OWSHELL_DEV='usb'
OW_OWSHELL_PORT='127.0.0.1:4304'

OW_OWHTTPD='yes'
OW_OWHTTPD_RUN='yes'
OW_OWHTTPD_DEV='127.0.0.1:4304'
OW_OWHTTPD_PORT='8080'
```

Die folgenden Variablen in der Datei /config/ow.txt sind zu konfigurieren:

OPT_OW Die Standardeinstellung von OPT_OW='no', das Paket wird nicht installiert. Mit `OPT_OW`='yes' wird das Paket aktiviert.

OW_USER_SCRIPT Hinter dieser Variablen verbirgt sich Pfad und Dateiname einer optionalen Hintergrundsteuerung, mit der zum Beispiel die Heizungsanlage geregelt werden kann. Nähere Erläuterungen finden sich in Kapitel 1.1.8.3.

OW_OWFS OWFS bietet einfachen Zugriff auf den 1–Wire Bus über die fli4l-Weboberfläche. Mit der Auswahl `OW_OWFS`='yes' wird mittels Fuse ein Dateisystem unter dem Standardpfad '/var/run/ow' erzeugt. Dort wird der 1–Wire-Bus abgebildet. Die im Dateisystem angelegten Verzeichnisse sind nach den Identnummern (siehe Anhang A.1.2) der Chips geordnet. Über den family code der Bauteile ist eine entsprechende Systematik leicht herzustellen.

OW_OWFS_DEV Mit der Variablen `OW_OWFS_DEV` legt man die PC-Schnittstelle fest, an der der 1–Wire-Adapter angeschlossen wird.

PC-Schnittstelle	**Variablenbelegung**	**Beispiel**
seriell	ttyS*	ttyS0 = COM1, ttyS1 = COM2
USB	ttyUSB*	ttyUSB1 = erster USB-Adapter
	usb	usb = erster USB-Adapter
	usb[2-9]	usb3 = dritter USB-Adapter
I^2C	i^2c-[0-9]	i^2c-0 = erster I^2C-Port
Simulation	fake	Für die Verwendung der Modi '`FAKE`' und '`TESTER`' müssen die Variablen `OW_OWFS_FAKE` oder `OW_OWFS_TESTER` auf gültige family codes gesetzt werden, siehe Kapitel 1.1.6.1
	tester	

OW_OWFS_GROUP_N Die Variable `OW_OWFS_GROUP_N` bestimmt die Anzahl der im Browser angezeigten Gruppen in die zusammen gehörige Ein-und Ausgänge für zum Beispiel die Steuerung einer Solaranlage zusammen gefasst werden und deren Namen mit OW_OWFS_GROUP_NAME festgelegt wird.

OW_OWFS_GROUP_x_PORT_N OW_OWFS_GROUP_x_PORT_x_ID OW_OWFS__GROUP_x_
Die Variable `OW_OWFS_GROUP_x_PORT_N` bestimmt die Anzahl Ports einer Gruppe. Mit den beiden untergeordneten Variablen `OW_OWFS_GROUP_x_PORT_x_ID` und `OW_OWFS_GROUP_x_PORT_x_ALIAS` weist man den jeweiligen Aus-, bzw. Eingängen des 1–Wire Bauteils einen Decknamen zu.

Möchte man bestimmte Anzeigen im Webinterface unterdrücken, weil zum Beispiel der Port eines Bauteils nicht belegt wurde oder der Admin-Zweig nach Abschluss der Konfiguration nicht mehr benötigt wird, dann kann man dem Namen ein Rufzeichen (!) voranstellen.

OW_OW_SHELL Aktivierung des SServerdienstes"vom OWFS, zur Bereitstellung des OWFS-BUS gleichzeitig für mehrere Anwendungen (OWFS und OWHTTPD). Dabei darf dann keine der anderen Anwendungen auf die direkte Schnittstelle des Adapters eingestellt sein, sondern muss hier auf den Server verknüpft werden.

OW_OW_SHELL_RUN Soll der Serverdienst beim Booten sofort gestartet werden?

OW_OW_SHELL_DEV Device, auf welches der Server zugreift (Hardware)

OW_OW_SHELL_PORT IP-Adresse und Port, auf dem der Serverdienst arbeitet. Sinnvoll ist hier nur die Localhost-Adresse 127.0.0.1 Standardmäßig sollte der Port 4304 (Standard bei OWFS) für den Serverdienst eingestellt bleiben. Diese Addresse ist fest im Paket RRDTool hinterlegt. Wenn also Werte per RRDTool erfasst werden sollen, muss dies so bleiben.

OW_OWHTTPD Aktivierung des vom OWFS selbst bereitgestellten Webservers.

OW_OWHTTPD_RUN Soll der Webserver beim Booten gestartet werden?

OW_OWHTTPD_READONLY Darf ein schreibender Zugriff auf die Bauteile im OWFS über den Webserver erfolgen.

OW_OWHTTPD_DEV Device, auf das der Webserver zugreift. Im Zusammenhang mit OW_OWSHELL (Server) kann hier auch gleichzeitig auf ein einzelnes Device zugegriffen werden.

OW_OWHTTPD_PORT HTTP-Port, auf dem der Webserver seinen Dienst bereit stellt.

Konfigurationsbeispiel:

```
OW_OWFS_GROUP_x_PORT_x_ID='29.57D305000000/P6'
OW_OWFS_GROUP_x_PORT_x_ALIAS='EA-Modul/!P6'        # Signal unterdrückt
OW_OWFS_GROUP_x_PORT_x_ID='29.57D305000000/Admin/*'
OW_OWFS_GROUP_x_PORT_x_ALIAS='EA-Modul/Admin/!'    # Admin-Pfad komplett
                                                   # abgeschaltet
```

Eine weitergehende Beschreibung zur Konfiguration von OWFS gibt es im Anhang „A.1.6“ und hier: http://owfs.org/index.php?page=owfs.

1.1.6.1. Sonstige Variablen

Die folgenden Variablen können bei Bedarf über die Datei config/ow.txt angepasst werden:

OW_LOG_DESTINATION Ziel für Status-und Fehlerausgaben.

```
0 = mixed (1 und 2)
1 = syslog
2 = stderr
3 = off
```

Standardwert ist '1'.

OW_LOG_LEVEL Der Log-Level (1-9) bestimmt die Anzahl der Status-und Fehlerausgaben, wobei:

```
1 = leise und 9 = geschwätzig
```

Standardwert ist '1'.

OW_TEMP_SCALE Die zur Verfügung stehenden Temperaturskalen.

```
C = "Celsius"
F = "Fahrenheit"
K = "Kelvin"
R = "Rankine"
```

Standardwert ist 'C'.

OW_REFRESH_INTERVAL Refresh der fli4l-HTTP-Anwendung in Sekunden. '0' = kein Refresh.

Standardwert ist '10'.

OW_OWFS_FAKE Aktiviert die zufällige Simulation von 1–Wire-Bauteilen. Es können mehrere Bauteile mit dem family code und durch Leerzeichen getrennt angegeben werden. Die simulierten Zustände sind rein zufällig. Die Option kann nicht gleichzeitig mit dem 'TESTER'-Modus aktiviert werden.

OW_OWFS_TESTER Aktiviert die systematische Simulation von 1–Wire- Bauteilen. Es können mehrere Bauteile mit dem family code und durch Leerzeichen getrennt angegeben werden. Die simulierten Zustände folgen natürlichen Werten. Die Option kann nicht gleichzeitig mit dem 'FAKE'-Modus aktiviert werden.

OW_OWFS_RUN Gibt an, ob owfs beim Boot des Routers automatisch gestartet werden soll. Standardwert ist 'yes', wohingegen nach dem Setzen von 'no' die Anwendung manuell gestartet werden muss.

OW_OWFS_READONLY Legt mit 'yes' fest, dass Bauteilzustände über owfs nur gelesen, aber nicht geschrieben werden können.

Standardwert ist 'no'.

OW_OWFS_PATH Gibt das Wurzelverzeichnis der Fuse-Verzeichnisstruktur an. Standardwert ist '/var/run/ow'. Das gewählte Verzeichnis sollte aus Gründen der Systemleistung unbedingt auf der RAMdisk liegen!

OW_CACHE_SIZE Dient zur Anpassung der maximalen Größe des cache in [bytes] auf Systemen mit sehr kleiner RAMdisk.

Der Standardwert '0' hebt jegliche Limitierung auf.

OW_USER_SCRIPT_INTERVAL Gibt in Sekunden an, wie lange zwischen zwei Durchläufen des user-script gewartet wird. Der Wert '0' sollte nur verwendet werden, wenn innerhalb des user-script ein 'sleep' ausgeführt wird.

OW_DEVICE_LIB Legt den absoluten Pfad und Dateinamen der Bauteilbibliothek auf dem Router fest. Durch die Verwendung eines anderen Werts als des Standardwerts '/srv/www/include/owdevice.lib' kann sichergestellt werden, dass bei einer Aktualisierung des Opt die Bibliothek nicht überschrieben wird und so persönliche Änderungen an der Bauteilbibliothek erhalten bleiben.

OW_INVERT_PORT_LEDS Invertiert den Status der Port-Leds von i/o Ports (latch*, sensed*, PIO*).

Standardwert ist 'no'.

1.1.6.2. Nicht dokumentierte Variablen

Die folgende Variablen sind nicht dokumentiert:

OW_MODULE_CONF_FILE

OW_USER_SCRIPT_STOP

OW_SCRIPT_WRAPPER

OW_MENU_ITEM

OW_RIGHTS_SECTION

OW_OWFS_PID_FILE

OW_OWFS_GROUP_x_NAME

OW_REFRESH_FILE

OW_REFRESH_TEMP

OW_ALIAS_FILE

OW_CSS_FILE

OW_OWHTTPD_FAKE

OW_OWHTTPD_TESTER

OW_OWHTTPD_PID_FILE

1.1.7. Bedienung im Browser und auf der Konsole

1.1.7.1. Browser

1.1.7.1.1. Webserver Der in fli4l optional zu installierende Webserver (opt_httpd) bietet die Möglichkeit, eigene Shell/CGI Skript Anwendungen über jeden Browser im Netz zu bedienen. Davon wurde im vorliegenden Falle Gebrauch gemacht. Um den Webserver zu nutzen muss config/httpd.txt entsprechend konfiguriert werden.

Im `OPT_OW` wird eine Browser Applikation mitgeliefert. Sie wird nur installiert, wenn in /config/ow.txt `OW_OWFS`='yes' gesetzt wird. Das zugehörige Skript befindet sich gemäß Vorgaben auf dem fli4l unter /srv/www/admin/ow.cgi und im fli4l-Installationsverzeichnis unter `fli4l-version\opt\files\srv\www\admin\ow.cgi`. Der zugehörige Menüpunkt erscheint unter „Opt / 1–Wire-Bus“.

1.1.7.1.2. Darstellung Im Reiter „Status“ werden die am 1–Wire-Bus angeschlossenen Bauteile gemäß der in config/ow.txt vorgenommenen Konfiguration gruppenweise in einer Baumstruktur angezeigt. Mittels ‚Klick‘ wird die jeweilige Gruppe geöffnet. Die konfigurierten Werte werden angezeigt. In der Admin-Struktur werden alle in der Bauteilbibliothek (siehe 8.4) für das jeweilige Bauteil definierten Parameter angezeigt. Hinsichtlich der Bedeutung dieser Parameter wird auf die Datenblätter von Maxim und die beiliegenden Man-Pages verwiesen.

Im Reiter „Admin“, der nur im Admin-Modus angezeigt wird, können die gewählten Applikationen ein-und ausgeschaltet werden.

Die angezeigten LEDs signalisieren mit ihren Farben folgende Zustände: LED grün = inaktiv (Ruhe) LED rot = aktiv (Betrieb) LED gelb = inaktiv (Warnung)

Die Schaltknöpfe dienen zum Umschalten der zugeordneten Ports. Das Symbol zeigt zusätzlich den aktuellen Schaltzustand. Hinsichtlich der Berechtigungen siehe 8.1.

1.1.7.2. Konsole

Die Abfrage und Steuerung der Sensoren und Aktoren ist auch auf der Konsole des fli4l oder über einen Remotezugriff (WinSCP, Putty o.ä.) möglich.

Mit z.B.:

- cat /var/run/ow/10.DEF0A8010800/temperature
  ruft man die Temperaturmessung eines DS19S20 ab.
- echo "1" > /var/run/ow/1C.7F6CF7040000/PIO.O
  schaltet man den Ausgang 1 eines DS28E04-100 (dual switch) ein.
- echo "0" > /var/run/ow/1C.7F6CF7040000/PIO.O
  schaltet man den Ausgang 1 wieder aus.

Eine weitere Beschreibung gibt es im Anhang „A.1.6“ und hier:
`http://owfs.org/index.php?page=owfs`

1.1.8. Erweiterte Funktionen

1.1.8.1. Rechtevergabe

In der fli4l-Webanwendung ist die Vergabe von Benutzerrechten implementiert, siehe hierzu die Erläuterungen in doc/deutsch/pdf/httpd.pdf. Dieser Rechtevergabe bedient sich auch `OPT_OW`.

Zur Nutzung der OW-Rechte können in der Datei config/ httpd.txt für den Bereich „ow“ die folgenden Stufen vorgegeben werden:
admin = alle Rechte
exec = Befehle ausführen, Ein-und Ausgänge schalten, Ansicht der Daten
view = Ansicht der Daten
Die Admin-Tabelle, über die owfs und das user-script ein-und ausgeschaltet werden können, wird in den Stufen „exec“ und „view“ nicht angezeigt. Weiterhin werden in diesen Berechtigungsstufen alle Anzeigen, die ein „Admin“ enthalten, ausgeblendet.

1.1.8.2. Bauteilebibliothek

Auf Grund der Vielzahl der von MAXIM angebotenen 1–Wire Bauteile wurde eine eigene Bauteilebibliothek angelegt. Das entsprechende Library-script liegt auf dem fli4l unter /srv/www/include/ow-device.lib und im fli4l-Installationsverzeichnis unter `fli4l-version\opt\files\srv\w`
Die Bibliothek enthält bereits einige wichtige Bauteile. Eigene Devices können entsprechend der verwendeten Systematik nachgetragen und über die fli4l-Newsgroup 'spline.fli4l.opt' den übrigen fli4l-Nutzern zur Verfügung gestellt werden. Es werden nur in der Bibliothek angelegte Bauteile im Browser angezeigt. Das Libraryscript kann wunschgemäß entweder mittels Programmen wie „WinSCP“ auf dem fli4l selbst zu Testzwecken oder als dauerhafte Änderung im fli4l-Installationsverzeichnis editiert werden.

1.1.8.3. OW_USER_SCRIPT

Das Skript findet sich auf dem fli4l unter /usr/local/bin/ow-user-script.sh und im fli4l-Installationsverzeichnis unter `fli4l-version\opt\files\usr\local\bin\ow-userscript.sh`. Es kann nach eigenen Wünschen und Bedürfnissen an die zu überwachenden und zu steuernden Anwendungen angepasst werden. Der Vorteil des Skript ist darin zu sehen, dass auch umfangreiche und komplexe Steuerungen auf bestehender Hardware möglich sind.

1.1.8.4. RRDTool

1.1.8.4.1. Schnittstelle Die über den 1–Wire-Bus erfassten Daten können mittels fli4l-Opt „RRDTool“ aufgezeichnet und graphisch aufbereitet werden. Das vorliegende Opt bringt die erforderlichen Schnittstellen bereits mit. Hierzu muss owfs (siehe /config/ow.txt) installiert sein. Bei der Installation von RRDTool sind in der /config/rrdtool.txt die gewünschten Einträge vorzunehmen. Dabei ist es zwingend erforderlich, dass beim Paket OW das `OW_SHELL` auf den Port 127.0.0.1:4304 aktiviert wird, das dass Collectd-Plugin von RRDTool auf diesen Port lauscht und die Daten aller Sensoren in getrennte Grafiken abbildet.

1.1.9. Feedback

Wir freuen uns über jede, auch kurze Rückmeldung, selbst wenn das Paket ohne jegliche Probleme laufen sollte.
Viel Spaß mit 1–Wire wünschen:
Klaus der Tiger E-Mail: der.tiger.opt-ow@arcor.de
Karl M. Weckler E-Mail: news4kmw@web.de
Roland Franke E-Mail: fli4l@franke-prem.de

A. Anhang zum Paket OW

A.1. OW

A.1.1. Pin-, Adernbelegung für TP-Kabel und RJ-45

Eine Übersicht über die Belegungsvarianten findet sich hier:
http://www.1wire.org/media/A_Guide_to_the_1WRJ45_Standard_Draft.zip

Wer sich nicht durch alle Varianten quälen möchte, verwendet folgendes universelle und zukunftsfähige Schema für die RJ45 Pin-Belegung.

Farbangaben nach:			
Pin	EIA/TIA 568A		EIA/TIA 568B
1	Grün/Weiß	Hauptversorgung GND	Orange/Weiß
2	Grün	Hauptversorgung +5V/50mA für 1–Wire Bauteile	Orange
3	Orange/Weiß	Sekundärer 1–Wire Bus GND	Grün/Weiß
4	Blau	Primärer 1–Wire Bus	Blau
5	Blau/Weiß	Primärer 1–Wire Bus GND	Blau/Weiß
6	Orange	Sekundärer 1–Wire Bus	Grün
7	Braun/Weiß	Hilfsversorgung z.B. +12V/200mA für andere Verbraucher	Braun/Weiß
8	Braun	Hilfsversorgung GND	Braun

Hinweis: Bei langen Busstrecken wirkt sich der ohmsche Widerstand negativ auf die Versorgungsspannungen aus. Insofern ist, soweit möglich, eine anwendungsseitige Stromversorgung zu bevorzugen.

A.1.2. Family Code Referenz

(aus Maxim AppNote 155)

Family Code (hex)	Bauteil / Chip	Beschreibung (Speichergröße in bits, wnaa)
01	(DS1990A), (DS1990R), DS2401, DS2411	1–Wire net address (registration number) only
02	(DS1991)*	Multikey iButton, 1152-bit secure memory
04	(DS1994), DS2404	4Kb NV RAM memory and clock, timer, alarms
05	DS2405*	Single addressable switch
06	(DS1993)	4Kb NV RAM memory
08	(DS1992)	1Kb NV RAM memory
09	(DS1982), DS2502	1Kb EPROM memory
0A	(DS1995)	16Kb NV RAM memory
0B	(DS1985), DS2505	16Kb EPROM memory
0C	(DS1996)	64Kb NV RAM memory
0F	(DS1986), DS2506	64Kb EPROM memory
10	(DS1920)	Temperature with alarm trips
12	DS2406, DS2407*	1Kb EPROM memory, 2-channel addressable switch
14	(DS1971), DS2430A*	256-bit EEPROM memory and 64-bit OTP register
1A	(DS1963L)*	4Kb NV RAM memory with write cycle counters
1C	DS28E04-100	4096-bit EEPROM memory, 2-channel addressable switch
1D	DS2423*	4Kb NV RAM memory with external counters
1F	DS2409*	2-channel addressable coupler for sub-netting
20	DS2450	4-channel A/D converter (ADC)
21	(DS1921G), (DS1921H), (DS1921Z)	Thermochron® temperature logger
23	(DS1973), DS2433	4Kb EEPROM memory
24	(DS1904), DS2415	Real-time clock (RTC)
27	DS2417	RTC with interrupt
29	DS2408	8-channel addressable switch
2C	DS2890*	1-channel digital potentiometer
2D	(DS1972), DS2431	1024-bit, 1-Wire EEPROM
37	(DS1977)	Password-protected 32KB (bytes) EEPROM
3A	(DS2413)	2-channel addressable switch
41	(DS1922L), (DS1922T), (DS1923), DS2422	High-capacity Thermochron (temperature) and Hygrochron™ (humidity) loggers
42	DS28EA00	Programmable resolution digital thermometer with sequenced detection and PIO
43	DS28EC20	20Kb 1-Wire EEPROM

Die Liste erhebt keinen Anspruch auf Vollständigkeit
* Diese Chips werden nicht mehr für Neuentwicklungen empfohlen.

A.1.3. Links zum Thema 1–Wire

http://www.maxim-ic.com/auto_info.cfm/
http://www.1wire.org/

http://www.simat.org.uk/

http://owfs.org/
http://sourceforge.net/projects/owfs/
http://fuse.sourceforge.net/
http://sourceforge.net/projects/fuse/

A.1.4. Bezugsquellen für 1–Wire Bauteile

http://www.maxim-ic.com/auto_info.cfm
MAXIM verschickt auch kostenlose Muster (Sample)

http://www.fuchs-shop.com/
http://www.spezial.com/
http://www.1-wire.de/
http://www.reichelt.de/
Das 1–Wire Angebot ist eher dürftig, aber sonst...

http://www.homechip.com/catalog/
http://www.aagelectronica.com/aag/index.html
http://www.hobby-boards.com/catalog/main_page.php
http://www.tme.eu/de/

A.1.5. Schaltplanskizzen

A.1.5.1. Passiver serieller zu 1–Wire Adapter

Links:
http://www.hobby-boards.com/catalog/main_page.php
http://www.rockenberg.net/ow.html

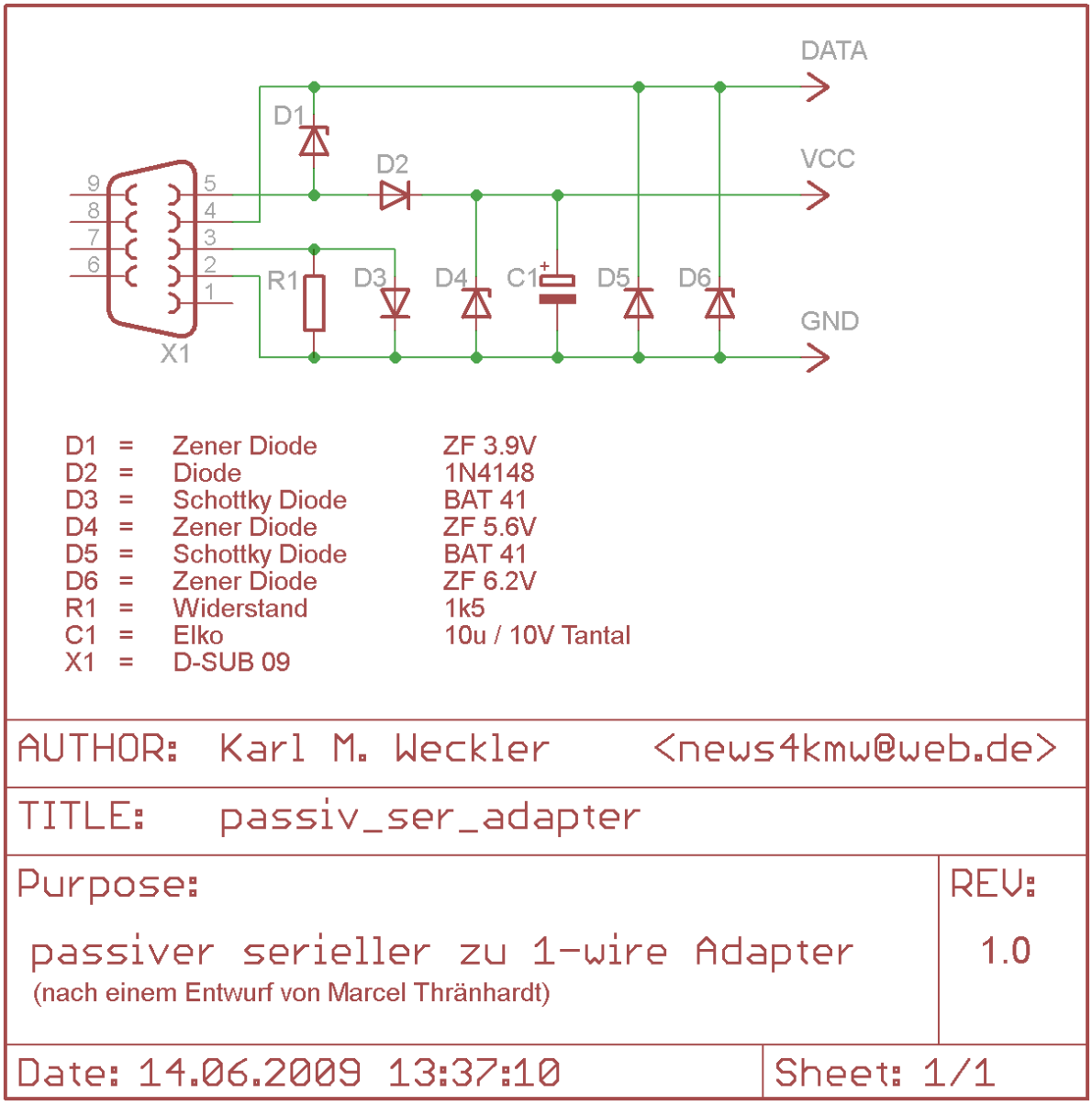

Abbildung A.1.: Passiver serieller zu 1–Wire Adapter

A.1.6. Man-Pages

Die komplette Manual Pages Liste findet sich unter:
http://owfs.org/index.php?page=software

A.1.6.1. OWFS

Name **owfs** - 1-wire filesystem

Synopsis **owfs** [*-c* config] *-d* serialport | *-u* | *-s* [host:]port *-m* mountdir

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

owfs **owfs (1)** is the filesystem client of the *OWFS* family of programs. It runs on linux, freebsd and Mac OS X, and requires the *fuse* kernel module and library. (http://fuse.sourceforge.net/) which is a user-mode filesystem driver.

Essentially, the entire 1-wire bus is mounted to a place in your filesystem. All the 1-wire devices are accessible using standard file operations (read, write, directory listing). The system is safe, no actual files are exposed, these files are virtual. Not all operations are supported. Specifically, file creation, deletion, linking and renaming are not allowed. (You can link from outside to a owfs file, but not the other way around).

Device Options (1-wire Bus Master) These options specify the device (bus master) connecting the computer to the 1-wire bus. The 1-wire slaves are connected to the 1-wire bus, and the bus master connects to a port on the computer and controls the 1-wire bus. The bus master is either an actual physical device, the kernel w1 module, or an **owserver (1)**.

At least one device option is required. There is no default. More than one device can be listed, and all will be used. (A logical union unless you explore the */bus.n/* directories.)

Linux and BSD enforce a security policy restricting access to hardware ports. You must have sufficient rights to access the given port or access will silently fail.

Serial devices *port* specifies a serial port, e.g. */dev/ttyS0*

-d port | ***–device=port (DS2480B)*** DS2480B-based bus master (like the DS9097U or the LINK in emulation mode). If the adapter doesn't respond, a passive type (DS9907E or diode/resistor) circuit will be assumed.

–serial_flextime | **–serial_regulartime (DS2480B)** Changes details of bus timing (see DS2480B datasheet). Some devices, like the *Swart* LCD cannot work with *flextime.*

–baud=**1200|9600|19200|38400|57600|115200 (DS2480B,LINK,HA5)** Sets the initial serial port communication speed for all bus masters. Not all serial devices support all speeds. You can change the individual bus master speed for the **LINK** and **DS2880B** in the interface/settings directory. The **HA5** speed is set in hardware, so the command line buad rate should match that rate.

Usually the default settings (9600 for **LINK** and **DS2480B**) and 115200 for the **HA5** are sane and shouldn't be changed.

–straight_polarity | ***–reverse_polarity (DS2480B)*** Reverse polarity of the DS2480B output transistors? Not needed for the DS9097U, but required for some other designs.

–link=port (LINK) **iButtonLink** *LINK* adapter (all versions) in non-emulation mode. Uses an ascii protocol over serial.

–ha7e=port (HA7E) **Embedded Data Systems** *HA7E* adapter (and *HA7S*) in native ascii mode.

–ha5=port | ***–ha5=port:a*** | ***–ha5=port:acg (HA5)*** **Embedded Data Systems** *HA5* mutidrop adapter in native ascii mode. Up to 26 adapters can share the same port, each with an assigned letter. If no letter specified, the program will scan for the first response (which may be slow).

–checksum | **–no_checksum (HA5)** Turn on (default) or off the checksum feature of the HA5 communication.

–passive=port | ***–ha2=port*** | ***–ha3=port*** | ***–ha4b=port (Passive)*** Passive 1-wire adapters. Powered off the serial port and using passive electrical components (resitors and diodes).

–8bit | **–6bit (Passive)** Synthesize the 1-wire waveforme using a 6-bit (default) serial word, or 8-bit word. Not all UART devices support 6 bit operation.

–timeout_serial=5 Timeout (in seconds) for all serial communications. 5 second default. Can be altered dynamically under */settings/timeout/serial*

USB devices The only supported true USB bus masters are based on the DS2490 chip. The most common is the DS9490R which has an included 1-wire ID slave with family code 81.

There are also bus masters based on the serial chip with a USB to serial conversion built in. These are supported by the serial bus master protocol.

-u | ***–usb*** DS2490 based bus master (like the DS9490R).

-u2 | ***–usb=2*** Use the second USB bus master. (The order isn't predicatble, however, since the operating system does not conssitently order USB devices).

-uall | ***–usb=ALL*** Use all the USB devices.

–usb_flextime | **–usb_regulartime** Changes the details of 1-wire waveform timing for certain network configurations.

–altusb Willy Robion's alternative USB timing.

–timeout_usb=5 Timeout for USB communications. This has a 5 second default and can be changed dynamically under */settings/timeout/usb*

I2C devices I2C is 2 wire protocol used for chip-to-chip communication. The bus masters: *DS2482-100,* DS2482-101 and *DS2482-800* can specify (via pin voltages) a subset of addresses on the i2c bus. Those choices are

i2c_address

0,1,2,3 0x18,0x19,0x1A,0x1B

4,5,6,7 0x1C,0x1D,0x1E,0x1F (DS2482-800 only)

port for i2c masters have the form */dev/i2c-0,* /dev/i2c-1, ...

-d port | ***–device=port*** This simple form only permits a specific *port* and the first available *i2c_address*

–i2c=port | ***–i2c=port:i2c_address*** | ***–i2c=port:ALL*** Specific i2c *port* and the *i2c_address* is either the first, specific, or all or them. The *i2c_address* is 0,1,2,...

–i2c | ***–i2c=:*** | ***–i2c=ALL:ALL*** Search the available i2c buses for either the first, the first, or every i2c adapter.

The *DS2482-800* masters 8 1-wire buses and so will generate 8 */bus.n* entries.

Network devices These bus masters communicate via the tcp/ip network protocol and so can be located anywhere on the network. The *network_address* is of the form tcp_address:port

E.g. 192.168.0.1:3000 or localhost:3000

–link=network_address LinkHubE network LINK adapter by **iButtonLink**

–ha7net=network_address | **–ha7net** HA7Net network 1-wire adapter with specified tcp address or discovered by udp multicast. By **Embedded Data Systems**

–timeout_ha7=60 specific timeout for HA7Net communications (60 second default).

–etherweather=network_address Etherweather adapter

-s network_address | ***–server=network_address*** Location of an **owserver (1)** program that talks to the 1-wire bus. The default port is 4304.

–timeout_network=5 Timeout for network bus master communications. This has a 1 second default and can be changed dynamically under */settings/timeout/network*

Simulated devices Used for testing and development. No actual hardware is needed. Useful for separating the hardware development from the rest of the software design.

devices is a list of comma-separated 1-wire devices in the following formats. Note that a valid CRC8 code is created automatically.

10,05,21 Hexidecimal *family* codes (the DS18S20, DS2405 and DS1921 in this example).

10.12AB23431211 A more complete hexidecimal unique address. Useful when an actual hardware device should be simulated.

DS2408,DS2489 The 1-wire device name. (Full ID cannot be speciifed in this format).

–fake=devices Random address and random values for each read. The device ID is also random (unless specified).

–temperature_low=12 **–temperature_high=44** Specify the temperature limits for the *fake* adapter simulation. These should be in the same temperature scale that is specified in the command line. It is possible to change the limits dynamically for each adapter under */bus.x/interface/settings/simulated/[temperature_low|temperature_high]*

–tester=devices Predictable address and predictable values for each read. (See the website for the algorhythm).

w1 kernel module This a linux-specific option for using the operating system's access to bus masters. Root access is required and the implementation was still in progress as of owfs v2.7p12 and linux 2.6.30.

Bus masters are recognized and added dynamically. Details of the physical bus master are not accessible, bu they include USB, i2c and a number of GPIO designs on embedded boards.

Access is restrict to superuser due to the netlink broadcast protocol employed by w1. Multitasking must be configured (threads) on the compilation.

–w1 Use the linux kernel w1 virtual bus master.

–timeout_w1=10 Timeout for w1 netlink communications. This has a 10 second default and can be changed dynamically under */settings/timeout/w1*

Specific Options

-m –mountpoint=directory_path Path of a directory to mount the 1-wire file system
The mountpoint is required. There is no default.

–allow_other Shorthand for fuse mount option o allow_otherÄllows other users to see the fuse (owfs) mount point and file system. Requires a setting in /etc/fuse.conf as well.

–fuse-opt "options" Sends options to the fuse-mount process. Options should be quoted, e.g. o allow_other".

Temperature Scale Options

-C –Celsius

-F –Fahrenheit

-K –Kelvin

-R –Rankine Temperature scale used for data output. Celsius is the default.
Can also be changed within the program at */settings/units/temperature_scale*

Pressure Scale Options

–mbar (default)

–atm

–mmHg

–inHg

–psi

–Pa Pressure scale used for data output. Millibar is the default.
Can also be changed within the program at */settings/units/pressure_scale*

Format Options Choose the representation of the 1-wire unique identifiers. OWFS uses these identifiers as unique directory names.

Although several display formats are selectable, all must be in *family-id-crc8* form, unlike some other programs and the labelling on iButtons, which are *crc8-id-family* form.

-f –format="f[.]i[[.]c]" Display format for the 1-wire devices. Each device has a 8byte address, consisting of:

f family code, 1 byte

i ID number, 6 bytes

c CRC checksum, 1 byte

Possible formats are *f.i* (default, 01.A1B2C3D4E5F6), *fi* fic f.ic f.i.c and *fi.c*

All formats are accepted as input, but the output will be in the specified format.

The address elements can be retrieved from a device entry in owfs by the *family,* id and crc8 properties, and as a whole with *address.* The reversed id and address can be retrieved as *r_id* and *r_address.*

Job Control Options

-r –readonly

-w –write Do we allow writing to the 1-wire bus (writing memory, setting switches, limits, PIOs)? The *write* option is available for symmetry, it's the default.

-P –pid-file "filename" Places the PID – process ID of owfs into the specified filename. Useful for startup scripts control.

–background | –foreground Whether the program releases the console and runs in the *background* after evaluating command line options. *background* is the default.

–error_print=0|1|2|3

=0 default mixed destination: stderr foreground / syslog background

=1 syslog only

=2 stderr only

=3 /dev/null (quiet mode).

–error_level=0..9

=0 default errors only

=1 connections/disconnections

=2 all high level calls

=3 data summary for each call

=4 details level

>***4*** debugging chaff

–error_level=9 produces a lot of output

Configuration File

-c file | –configuration file Name of an **owfs (5)** configuration file with more command line parameters

Help Options See also this man page and the web site http://www.owfs.org/

-h –help=[device|cache|program|job|temperature] Shows basic summary of options.

device 1-wire bus master options

cache cache and communication size and timing

program mountpoint or TCP server settings

job control and debugging options

temperature Unique ID display format and temperature scale

-V –version *Version* of this program and related libraries.

Time Options Timeouts for the bus masters were previously listed in *Device* options. Timeouts for the cache affect the time that data stays in memory. Default values are shown.

–timeout_volatile=15 Seconds until a *volatile* property expires in the cache. Volatile properties are those (like temperature) that change on their own.
Can be changed dynamically at */settings/timeout/volatile*

–timeout_stable=300 Seconds until a *stable* property expires in the cache. Stable properties are those that shouldn't change unless explicitly changed. Memory contents for example.
Can be changed dynamically at */settings/timeout/stable*

–timeout_directory=60 Seconds until a *directory* listing expires in the cache. Directory lists are the 1-wire devices found on the bus.
Can be changed dynamically at */settings/timeout/directory*

–timeout_presence=120 Seconds until the *presence* and bus location of a 1-wire device expires in the cache.
Can be changed dynamically at */settings/timeout/presence*
There are also timeouts for specific program responses:

–timeout_server=5 Seconds until the expected response from the **owserver (1)** is deemed tardy.
Can be changed dynamically at */settings/timeout/server*

–timeout_ftp=900 Seconds that an ftp session is kept alive.
Can be changed dynamically at */settings/timeout/ftp*

Example

owfs -d /dev/ttyS0 -m /mnt/1wire Bus master on serial port

owfs -F -u -m /mnt/1wire USB adapter, temperatures reported in Fahrenheit

owfs -s 10.0.1.2:4304 -m /mnt/1wire Connect to an **owserver (1)** process that was started on another machine at tcp port 4304

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3)

Humidity **DS1922 (3)**

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.2. OWSHELL

Name owdir, owread, owwrite, owget, owpresent - lightweight owserver access

Synopsis **owdir** *-s* [host:]port [directory]
owread *-s* [host:]port filepath
owwrite *-s* [host:]port filepath value
owget *-s* [host:]port [directory] | filepath
owdir *–autoserver* [directory]
owread *–autoserver* filepath
owwrite *–autoserver* filepath value
owget *–autoserver* [directory] | filepath
owdir *-f* –format f[.]i[[.]c]] [*–dir*] *-s* [host:]port [directory] [directory2 ...]
owread *-C* –celsius *-K* –kelvin *-F* –fahrenheit *-R* –rankine [*–hex*] [*–start=* offset] [*–size=* bytes] *-s* [host:]port filepath [filepath2 ...]
owwrite *-C* –celsius *-K* –kelvin *-F* –fahrenheit *-R* –rankine [*–hex*] [*–start=* offset] *-s* [host:]port filepath value [filepath2 value2 ...]
owget *-f* –format f[.]i[[.]c] *-C* –celsius *-K* –kelvin *-F* –fahrenheit *-R* –rankine [*–hex*] [*–start=* offset] [*–size=* bytes] [*–dir*] *-s* [host:]port [directory] | filepath
owdir *-V* –version
owread *-V* –version
owwrite *-V* –version
owget *-V* –version
owdir *-h* | –help
owread *-h* | –help
owwrite *-h* | –help
owget *-h* | –help

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

OWSHELL programs **owdir owread owwrite** and **owget** are collectively called the **owshell** programs. They allow lightweight access to an **owserver (1)** for use in command line scripts.

Unlike **owserver (1)** owhttpd (1) owftpd (1) owhttpd (1) there is not persistent connection with the 1-wire bus, no caching and no multithreading. Instead, each program connects to a running **owserver (1)** and performs a quick set of queries.

owserver (1) performs the actual 1-wire connection (to physical 1-wire busses or other **owserver** programs), performs concurrency locking, caching, and error collection.

owshell programs are intended for use in command line scripts. An alternative approach is to mount an **owfs (1)** filesystem and perform direct file lists, reads and writes.

owdir **owdir** performs a *directory* listing. With no argument, all the devices on the main 1-wire bus will be listed. Given the name of a 1-wire device, the available properties will be listed. It is the equivalent of

***ls* directory**

in the **owfs (1)** filesystem.

owread **owread** obtains for value of a 1-wire device property. e.g. 28.0080BE21AA00/temperature gives the DS18B20 temperature. It is the equivalent of

***cat* filepath**

in the **owfs (1)** filesystem.

owwrite **owwrite** performs a change of a property, changing a 1-wire device setting or writing to memory. It is the equivalent of

***echo* "value"> filepath**

in the **owfs (1)** filesystem.

owget **owget (1)** is a convenience program, combining the function of **owdir (1)** and **owread (1)** by first trying to read the argument as a directory, and if that fails as a 1-wire property.

Standard Options

–autoserver Find an *owserver* using the Service Discovery protocol. Essentially Apple's Bonjour (aka zeroconf). Only the first *owserver* will be used, and that choice is probably arbitrary.

-s [host:]port Connect via tcp (network) to an *owserver* process that is connected to a physical 1-wire bus. This allows multiple processes to share the same bus. The *owserver* process can be local or remote.

Data Options

–hex Hexidecimal mode. For reading data, each byte of character will be displayed as two characrters 0-9ABCDEF. Most useful for reading memory locations. No spaces between data.

Writing data in hexidecimal mode just means that the data should be given as one long hexidecimal string.

–start=offset Read or write memory locations starting at the offset byte rather than the beginning. An offset of 0 means the beginning (and is the default).

–size=bytes Read up to the specified number of bytes of a memory location.

Help Options

-h –help Shows basic summary of options.

-V –version *Version* of this program and related libraries.

Display Options

–dir Modify the display of directories to indicate which entries are also directories. A directory member will have a trailing '/' if it is a directory itself. This aids recursive searches.

-f –format "f[.]i[[.]c]" Display format for the 1-wire devices. Each device has a 8 byte address, consisting of:

f family code, 1 byte

i ID number, 6 bytes

c CRC checksum, 1 byte

Possible formats are *f.i* (default, 01.A1B2C3D4E5F6), *fi* fic f.ic f.i.c and *fi.c*
All formats are accepted as input, but the output will be in the specified format.

Example

owdir -s 3000 –format fic Get the device listing (full 16 hex digits, no dots) from the local *owserver* at port 3000

owread -F –autoserver 51.125499A32000/typeK/temperature Read temperature from the DS2751-based thermocouple on an auto-discovered *owserver* Temperature in fahrenheit.

owwrite -s 10.0.1.2:3001 32.000800AD23110/pages/page.1 "Passed" Connect to a OWFS server process (*owserver*) that was started on another machine at tcp port 3001 and write to the memory of a DS2780

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3)

Humidity **DS1922 (3)**

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.3. OWFS.CONF

Name **owfs.conf** - owfs programs configuration file

Synopsis An OWFS configuration file is specified on the command line:

owfs -c config_file [other options] The file name is arbitrary, there is no default configuration file used.

Usage A configuration file can be invoked for any of the OWFS programs (**owfs (1)** owhttpd (1) owserver (1) owftpd (1)) or any of the language bindings (**owperl (1)** owcapi (1) owtcl (1) owphp owpython) to set command line parameters.

Syntax Similar to Unix shell script or perl syntax

Comments # Any # marks the start of a comment

blank lines are ignored

Options **option** # some options (like 'foreground') take no values

option = value # other options need a value

option value # '=' can be omitted if whitespace separates

Option # Case is ignored (for options, not values)

opt # non-ambiguous abbreviation allowed

-opt –opt # hyphens ignored

owserver **server:** opt = value # only *owserver* effected by this line

! server: opt = value # *owserver* NOT effected by this line

owhttpd **http:** opt = value # only *owhttpd* effected by this line

! http: opt = value # *owhttpd* NOT effected by this line

owftpd **ftp:** opt = value # only *owftpd* effected by this line

! ftp: opt = value # *owftpd* NOT effected by this line

owfs **owfs:** opt = value # only *owfs* effected by this line

! owfs: opt = value # *owfs* NOT effected by this line

Limits # maximum line length of 250 characters

no limit on number of lines

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Configuration **owfs.conf (5)** allows a uniform set of command line parameters to be set.

Not all OWFS programs use the same command line options, but the non-relevant ones will be ignored.

Command line and configuration options can mixed. They will be invoked in the order presented. Left to right for the command line. Top to bottom for the configuration file.

Configuration files can call other configuration files. There is an arbitrary depth of 5 levels to prevent infinite loops. More than one configuration file can be specified.

Sample

Here is a sample configuration file with all the possible parameters included. # **Sources**

device = /dev/ttyS0 # serial port: DS9097U DS9097 ECLO or LINK

device = /dev/i2c-0 # i2c port: DS2482-100 or DS2482-800

usb # USB device: DS9490 PuceBaboon

usb = 2 # Second DS9490

usb = all # All DS9490s

altUSB # Willy Robison's tweaks
LINK = /dev/ttyS0 # serial LINK in ascii mode
LINK = [address:]port # LINK-HUB-E (tcp access)
HA7 # HA7Net autodiscovery mode
HA7 = address[:port] # HA7Net at tcp address (port 80)
etherweather = address[:port] # Etherweather device
server = [address:]port # **owserver** tcp address
FAKE = 10,1B # Random simulated device with family codes (hex)
TESTER = 28,3E # Predictable simulated device with family codes
#
Sinks
owfs specific
mountpoint = filelocation # *FUSE* mount point
allow_other # Short hand for *FUSE* mount option "
owhttpd owserver owftpd specific
port = [address:]port # tcp out port
#
Temperature scales
Celsius # default
Fahrenheit
Kelvin
Rankine
#
Timeouts (all in seconds)
cache for values that change on their own
timeout_volatile = value # seconds "volatile"values remain in cache
cache for values that change on command
timeout_stable = value # seconds ßtable"values remain in cache
cache for directory lists (non-alarm)
timeout_directory = value # seconds "directory"values remain in cache
cache for 1-wire device location
timeout_presence = value # seconds "device presence"(which bus)
timeout_serial = value # seconds to wait for serial response
timeout_usb = value # seconds to wait for USB response
timeout_network = value # seconds to wait for tcp/ip response

timeout_ftp = value # seconds inactivity before closing ftp session
#
Process control
configuration = filename # file (like this) of program options
pid_file = filename # file to store PID number
foreground
background # default
readonly # prevent changing 1-wire device contents
write # default
error_print = 0-3 # 0-mixed 1-syslog 2-stderr 3-suppressed
error_level = 0-9 # increasing noise
#
zeroconf / Bonjour
zero # turn on zeroconf announcement (default)
nozero # turn off zeroconf announcement
annouce = name # name of announced service (optional)
autoserver # Add owservers descovered by zeroconf/Bonjour
noautoserver # Don't use zeroconf/Bonjour owservers (default)
#
tcp persistence
timeout_persistent_low = 600 # minimum time a persistent socket will stay open
timeout_persistent_high = 3600 # max time an idle client socket will stay around

#
Display
format = f[.]i[[.]c] # 1-wire address *f* amily *i* d code *c* rc
#
Cache
cache_size = 1000000 # maximum cache size (in bytes) or 0 for no limit (default 0) #
Information
(silly in a configuration file)
version
help
morehelp

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3)

Humidity **DS1922 (3)**

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.4. DS18S20

Name DS18S20 - High-Precision 1-Wire Digital Thermometer
DS1920 - iButton version of the thermometer

Synopsis Thermometer.
10 [.]XXXXXXXXXXXX[XX][/[**die** | **power** | **temperature** | **temphigh** | **templow** | **trim** | **trimblanket** | **trimvalid** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *10*

Special Properties

power *read-only,yes-no*
Is the chip powered externally (=1) or from the parasitically from the data bus (=0)?

temperature *read-only,* floating point
Temperature read by the chip at high resolution (12 bits). Units are selected from the invoking command line. See **owfs(1)** or **owhttpd(1)** for choices. Default is Celsius. Conversion takes 1000 msec.

Temperature Alarm Limits When the device exceeds either *temphigh* or *templow* temperature threshold the device is in the alarm state, and will appear in the alarm directory. This provides an easy way to poll for temperatures that are unsafe, especially if *simultaneous* temperature conversion is done.
Units for the temperature alarms are in the same *temperature* scale that was set for *temperature* measurements.
Temperature thresholds are stored in non-volatile memory and persist until changed, even if power is lost.

temphigh *read-write,* integer
Shows or sets the lower limit for the high temperature alarm state.

templow *read-write,* integer
Shows or sets the upper limit for the low temperature alarm state.

Temperature Errata There are a group of obscure internal properties exposed to protect against an hardware defect in certain batches of the B7 die of some DS18x20 chips. See http://www.1wire.org/en-us/pg_18.html or request AN247.pdf from Dallas directly.

errata/die *read-only,ascii*
Two character manufacturing die lot. "B6B7ör "C2"

errata/trim *read-write,unsigned* integer

32 bit trim value in the EEPROM of the chip. When written, it does not seem to read back. Used for a production problem in the B7 *die.*

Read allowed for all chips. Only the B7 chips can be written.

errata/trimblanket *read-write,yes-no*

Writing non-zero (=1) puts a default trim value in the chip. Only applied to the B7 *die.* Reading will be true (non-zero) if trim value is the blanket value. Again, only B7 chips will register true, and since the written trim values cannot be read, this value may have little utility.

errata/trimvalid *read-only,yes-no*

Is the *trim* value in the valid range? Non-zero if true, which includes all non-B7 chips.

Standard Properties

address

r_address *read-only,* ascii

The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).

address starts with the *family* code

r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii

The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii

The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii

The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).

r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii

Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.

If no **Link Locator** is between the device and the master, the *locator* field will be all FF.

r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no

Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii

Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds18s20 Ds1920 The **DS18S20 (3)** is one of several available 1-wire temperature sensors. It has been largely replaced by the **DS18B20 (3)** and **DS1822 (3)** as well as temperature/vlotage measurements in the **DS2436 (3)** and **DS2438 (3)**. For truly versatile temperature measurements, see the protean **DS1921 (3)** Thermachron (3).

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS18S20.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.5. DS2401

Name **DS2401** - Silicon Serial Number
DS1990A - Serial Number iButton
01 [.]XXXXXXXXXXXX[XX][/[**address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Synopsis Unique serial number only.

Family Code *01*

Special Properties None.

Standard Properties

address

r_address *read-only,* ascii
The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).
address starts with the *family* code
r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii
The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).
r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii
Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.
If no **Link Locator** is between the device and the master, the *locator* field will be all FF.
r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no
Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii
Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms

None.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.
Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.
The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.
Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2401 Ds1990a The **DS2401 (3)** and **DS1990A (3)** are the most basic of 1-wire devices. Their sole property is it's unique address. It can be used for unique identification. Nonetheless, many keylocks, night watchman systems, and tracking systems use this device.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS2401.pdf
http://pdfserv.maxim-ic.com/en/ds/DS1990A-F3-DS1990A-F5.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.6. DS2406 DS2407

Name DS2406, DS2407 - Dual Addressable Switch with 1kbit Memory

Synopsis Dual Switch, Write-once Memory
12 [.]XXXXXXXXXXXX[XX][/[**channels** | **latch.[A|B|ALL|BYTE]** | **memory** | **pages/page.[0-3|ALL]** | **PIO.[A|B|ALL|BYTE]** | **power** | **sensed.[A|B|ALL|BYTE]** | **set_alarm** | **TAI8570/[sibling|temperature|pressure]** | **T8A/volt.[0-7,ALL]** **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *12*

Special Properties

channels *read-only,* unsigned integer
Is this a 1 or 2 channel switch? The *DS2406* comes in two forms, one has only one *PIO* pin (PIO.A). Returns 1 or 2.

latch.A latch.B latch.ALL latch.BYTE *read-write,* yes-no

The activity latch is set to 1 with the first negative or positive edge detected on the associated PIO channel.

Writing any data will clear latch for all (both)) channels. This is a hardware "featureöf the chip.

ALL references both channels simultaneously, comma separated

BYTE references both channels simultaneously as a single byte, with channel A in bit 0.

memory *read-write,* binary

128 bytes of non-volatile, write-once data.

pages/page.0 ... pages/page.3 pages/page.ALL *read-write,* binary

Memory organized as 4 pages or 32 bytes. Memory is write-once.

ALL is the aggregate of all 4 pages, sequentially accessed.

Pio.a Pio.b Pio.all Pio.byte *read-write,* yes-no

State of the open-drain output (*PIO*) pin. 0 = non-conducting (off), 1 = conducting (on).

Writing zero will turn off the switch, non-zero will turn on the switch. Reading the *PIO* state will return the switch setting (flipflop in the data sheet). To determine the actual logic level at the switch, refer to the *sensed* property.

Note that the actual pin setting for the chip uses the opposite polarity – 0 for conducting, 1 for non-conducting. However, to turn a connected device on (i.e. to deliver power) we use the software concept of 1 as conducting or ön".

ALL references both channels simultaneously, comma separated.

BYTE references both channels simultaneously as a single byte, with channel A in bit 0.

power *read-only,* yes-no

Is the *DS2406* powered parasitically =0 or separately on the Vcc pin =1

sensed.A sensed.B sensed.ALL sensed.BYTE *read-only,* yes-no

Logic level at the *PIO* pin. 0 = ground. 1 = high (2.4V - 5V). Really makes sense only if the *PIO* state is set to zero (off), else will read zero.

ALL references both channels simultaneously, comma separated.

BYTE references both channels simultaneously as a single byte, with channel A in bit 0.

set_alarm *read-write,* unsigned integer (0-331)

A number consisting of three digits XYZ, where:

X channel selection

- *0* neither
- *1* A only
- *2* B only
- *3* A or B

Y source selection

0 undefined

1 latch

2 PIO

3 sensed

Z polarity selection

0 low

1 high

All digits will be truncated to the 0-3 (or 0-1) range. Leading zeroes are optional (and may be problematic for some shells).

Example:

311 Responds on Conditional Search when either latch.A or latch.B (or both) are set to 1.

<**100** Never responds to Conditional Search.

Tai8570/ *subdirectory*

Properties when the *DS2406* (3) is built into a *TAI8570.*

If the *DS2406* (3) is not part of a *TAI8570* or is not the controlling switch, attempts to read will result in an error.

TAI8570/pressure *read-only,* floating point

Barometric *pressure* in millibar.

TAI8570/sibling *read-only,* ascii

Hex address of the *DS2406* (3) paired with this chip in a *TAI8570.*

TAI8570/temperature *read-only,* floating-point

Ambient *temperature* measured by the *TAI8570* in prevailing temperature units (Centigrade is the default).

T8A/volt.[0-7|ALL] *read-only,* floating-point

Uses the T8A (by *Embedded* Data Systems) 8 channel voltage converter. Units in volts, 0 to 5V range. 12 bit resolution.

Standard Properties

address

r_address *read-only,* ascii

The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).

address starts with the *family* code

r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii
The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).
r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii
Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.
If no **Link Locator** is between the device and the master, the *locator* field will be all FF.
r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no
Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii
Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms Use the *set_alarm* property to set the alarm triggering criteria.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.
Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.
The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2406 The **DS2406 (3)** allows control of other devices, like LEDs and relays. It superceeds the **DS2405** and **DS2407** Alternative switches include the **DS2408** or even **DS2450**

The **DS2407** is practically identical to the *DS2406* except for a strange *hidden* mode. It is supported just like the **DS2406**

Tai8570 The *TAI-8570* Pressure Sensor is based on a 1-wire composite device by *AAG* Electronica. The *TAI8570* uses 2 *DS2406* (3) chips, paired as a reader and writer to synthesize 3-wire communication. Only 1 of the *DS2406* (3) will allow *temperature* or *pressure* readings. It's mate's address can be shown as *sibling.*

The *TAI8570* uses the *Intersema* MS5534a pressure sensor, and stores calibration and temperature compensation values internally.

Design and code examples are available from AAG Electronica http://aag.com.mx/ , specific permission to use code in a GPL product was given by Mr. Aitor Arrieta of AAG and Dr. Simon Melhuish of OWW.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS2406.pdf
http://pdfserv.maxim-ic.com/en/ds/DS2407.pdf
http://www.embeddeddatasystems.com/page/EDS/PROD/IO/T8A
http://oww.sourceforge.net/hardware.html#bp

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.7. DS2408

Name DS2408 - 1-Wire 8 Channel Addressable Switch

Synopsis 8 port switch
29 [.]XXXXXXXXXXXX[XX][/[**latch.[0-7|ALL|BYTE]** | **LCD_M/[clear|home|screen|message]** | **LCD_H/[clear|home|yxscreen|screen|message|onoff]** | **PIO.[0-7|ALL|BYTE]** | **power** | **sensed.[0-7|ALL|BYTE]** | **strobe** | **por** | **set_alarm** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *29*

Special Properties

latch.0 ... latch.7 latch.ALL latch.BYTE *read-write,* binary
The 8 pins (PIO) latch a bit when their state changes, either externally, or through a write to the pin.
Reading the *latch* property indicates that the latch has been set.
Writing "true"(non-zero) to ANY *latch* will reset them all. (This is the hardware design).
ALL is all *latch* states, accessed simultaneously, comma separated.
BYTE references all channels simultaneously as a single byte. Channel 0 is bit 0.

Pio.0 ... Pio.7 Pio.all Pio.byte *read-write,* yes-no
State of the open-drain output (*PIO*) pin. 0 = non-conducting (off), 1 = conducting (on).
Writing zero will turn off the switch, non-zero will turn on the switch. Reading the *PIO* state will return the switch setting. To determine the actual logic level at the switch, refer to the *sensed.0* ... sensed.7 sensed.ALL sensed.BYTE property.
ALL references all channels simultaneously, comma separated.
BYTE references all channels simultaneously as a single byte. Channel 0 is bit 0.

power *read-only,* yes-no
Is the *DS2408* powered parasitically (0) or separately on the Vcc pin (1)?

sensed.0 ... sensed.7 sensed.ALL *read-only,* yes-no
Logic level at the *PIO* pin. 0 = ground. 1 = high (2.4V - 5V). Really makes sense only if the *PIO* state is set to zero (off), else will read zero.
ALL references all channels simultaneously, comma separated.
BYTE references all channels simultaneously as a single byte. Channel 0 is bit 0.

strobe *read-write,* yes-no
RSTZ Pin Mode Control. Configures RSTZ as either RST input or STRB output:

0 configured as RST input (default)

1 configured as STRB output

por *read-write,* yes-no
Specifies whether the device has performed power-on reset. This bit can only be cleared to 0 under software control. As long as this bit is 1 the device will allways respond to a conditional search.

set_alarm *read-write,* integer unsigned (0-333333333)
A number consisting of 9 digits XYYYYYYYY, where:

X select source and logical term

0 PIO OR

1 latch OR

2 PIO AND

3 latch AND

Y select channel and polarity

0 Unselected (LOW)

1 Unselected (HIGH)

2 Selected LOW

3 Selected HIGH

All digits will be truncated to the 0-3 range. Leading zeroes are optional. Low-order digit is channel 0.
Example:

100000033 Responds on Conditional Search when latch.1 or latch.0 are set to 1.

222000000 Responds on Conditional Search when sensed.7 and sensed.6 are set to 0.

000000000 (0) Never responds to Conditional Search.

Lcd_h Lcd Screen Properites This mode uses the *DS2408* attached to a Hitachi HD44780 LCD controller in 4-bit mode. See *DATASHEET* for published details. Based on a commercial product from *HobbyBoards* by Erik Vickery.

LCD_H/clear *write-only,* yes-no
This will *clear* the screen and place the cursor at the start.

LCD_H/home *write-only,* yes-no
Positions the cursor in the *home* (upper left) position, but leaves the current text intact.

LCD_H/screen *write-only,* ascii text
Writes to the LCD *screen* at the current position.

LCD_H/screenyc *write-only,* ascii text
Writes to an LCD screen at a specified location. The controller doesn't know the true LCD dimensions, but typical selections are: 2x16 2x20 4x16 and 4x20.

Y (row) range 1 to 2 (or 4)

X (column) range 1 to 16 (or 20)

There are two formats allowed for the *screenyx* text, either ascii (readable text) or a binary form.

2 binary bytes The two first characters of the passed string have the line and row: e.g. "\x02\x04string"perl string writes ßtringät line 2 column 4.

ascii 2,12: Two numbers giving line and row: Separate with a comma and end with a colon e.g. "2,4:string"writes ßtringät line 2 column 4.

ascii 12: Single column number on the (default) first line: End with a colon e.g. "12:string"writes ßtringät line 1 column 12.

The positions are 1-based (i.e. the first position is 1,1).

LCD_H/onoff *write-only,* unsigned
Sets several screen display functions. The selected choices should be added together.

4 Display on

2 Cursor on

1 Cursor blinking

LCD_H/message *write-only,* ascii text
Writes a *message* to the LCD screen after clearing the screen first. This is the easiest way to display a message.

Lcd_m Lcd Screen Properites This mode uses the *DS2408* attached to a Hitachi HD44780 LCD controller in 8-bit mode. See *DATASHEET* for published details. Based on a design from Maxim and a commercial product from *AAG.*

LCD_M/clear *write-only,* yes-no
This will *clear* the screen and place the cursor at the start.

LCD_M/home *write-only,* yes-no
Positions the cursor in the *home* (upper left) position, but leaves the current text intact.

LCD_M/screen *write-only,* ascii text
Writes to the LCD *screen* at the current position.

LCD_M/screenyc *write-only,* ascii text
Writes to an LCD screen at a specified location. The controller doesn't know the true LCD dimensions, but typical selections are: 2x16 2x20 4x16 and 4x20.

Y (row) range 1 to 2 (or 4)

X (column) range 1 to 16 (or 20)

There are two formats allowed for the *screenyx* text, either ascii (readable text) or a binary form.

2 binary bytes The two first characters of the passed string have the line and row: e.g. "\x02\x04string"perl string writes ßtringät line 2 column 4.

ascii 2,12: Two numbers giving line and row: Separate with a comma and end with a colon e.g. "2,4:string"writes ßtringät line 2 column 4.

ascii 12: Single column number on the (default) first line: End with a colon e.g. "12:string"writes ßtringät line 1 column 12.

The positions are 1-based (i.e. the first position is 1,1).

LCD_M/onoff *write-only,* unsigned
Sets several screen display functions. The selected choices should be added together.

4 Display on

2 Cursor on

1 Cursor blinking

LCD_M/message *write-only,* ascii text
Writes a *message* to the LCD screen after clearing the screen first. This is the easiest way to display a message.

Standard Properties

address

r_address *read-only,* ascii
The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).
address starts with the *family* code
r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii
The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).
r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii
Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.
If no **Link Locator** is between the device and the master, the *locator* field will be all FF.
r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no
Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii
Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms Use the *set_alarm* property to set the alarm triggering criteria.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.
Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.
The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.
Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2408 The **DS2408 (3)** allows control of other devices, like LEDs and relays. It extends the **DS2406** to 8 channels and includes memory.

Alternative switches include the **DS2406, DS2407** and even **DS2450**

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS2408.pdf
http://www.hobby-boards.com/catalog/howto_lcd_driver.php
http://www.maxim-ic.com/appnotes.cfm/appnote_number/3286

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.8. DS2413

Name DS2413 - Dual Channel Addressable Switch

Synopsis Dual Switch
3A [.]XXXXXXXXXXXX[XX][/[**PIO.[A|B|ALL|BYTE]** | **sensed.[A|B|ALL|BYTE]** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *3A*

Special Properties

Pio.a Pio.b Pio.all Pio.byte *read-write,* yes-no

State of the open-drain output (*PIO*) pin. 0 = non-conducting (off), 1 = conducting (on).

Writing zero will turn off the switch, non-zero will turn on the switch. Reading the *PIO* state will return the switch setting. To determine the actual logic level at the switch, refer to the *sensed* property.

ALL references both channels simultaneously, comma separated.

BYTE references both channels simultaneously as a single byte, with channel A in bit 0.

sensed.A sensed.B sensed.ALL sensed.BYTE *read-only,* yes-no

Logic level at the *PIO* pin. 0 = ground. 1 = high (2.4V - 5V). Really makes sense only if the *PIO* state is set to zero (off), else will read zero.

ALL references both channels simultaneously, comma separated.

BYTE references both channels simultaneously as a single byte, with channel A in bit 0.

Standard Properties

address

r_address *read-only,* ascii

The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).

address starts with the *family* code

r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii

The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii

The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii

The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).

r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii

Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.

If no **Link Locator** is between the device and the master, the *locator* field will be all FF.

r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no

Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii

Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms Use the *set_alarm* property to set the alarm triggering criteria.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2413 The **DS2413 (3)** allows control of other devices, like LEDs and relays. It differs from the *DS2405* with a cleaner interface and two channels The *DS2413* also has two channels like the *DS2406* and *DS2407* but has no memory, and no alarm. There is also varying types of switch and sensing in the *DS2408,* DS2409, LCD, DS276x, DS2450.

Unique among the switches, the *DS2413* can switch higher voltages, up to 28V.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://datasheets.maxim-ic.com/en/ds/DS2413.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.9. DS2423

Name DS2423 - 4kbit 1-Wire RAM with Counter

Synopsis RAM and counters.
1D [.]XXXXXXXXXXXX[XX][/[**counters.[A|B|ALL]** | **memory** | **pages/page.[0-15|ALL]** | **pages/count.[0-15|ALL]** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *1D*

Special Properties

counters.A counters.B counters.ALL *read-only,* unsigned integer
Debounced external counter. Associated with RAM *page.14* and *page.15* Note: counter increments only. It is reset when the chip loses power.
ALL returns the two values, separated by a comma. They are read sequentially.

memory *read-write,* binary
512 bytes of memory.

pages/page.0 ... pages/page.15 pages/page.ALL *read-write,* binary
Memory is split into 16 pages of 32 bytes each. Memory is RAM, contents are lost when power is lost. *ALL* is an aggregate of the pages. Each page is accessed sequentially.

pages/count.0 ... pages/count.15 pages/count.ALL *read-only,* unsigned integer

Write access to each page of memory. Actually only *page.12* and *page.13* have write counters.

page14 and *page.15* 's counters are associated with the external *counters.A* and *counters.B* triggers.

The value 0xFFFFFFFF is returned for *pages/count.0* through *pages/count.11*

ALL is an aggregate of the counters, comma separated. Each page is accessed sequentially.

Standard Properties

address

r_address *read-only,* ascii

The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).

address starts with the *family* code

r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii

The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii

The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii

The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).

r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii

Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.

If no **Link Locator** is between the device and the master, the *locator* field will be all FF.

r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no

Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii

Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms None.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2423 The **DS2423 (3)** is used for its counters. The internal counters (associated with pages 12 and 13) can detect memory tampering.

The external counters A and B page been used in circuit design, such as a wind anometer. *OWFS* system handles this automatically.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS2423.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.10. DS2433

Name DS2433 - EEPROM (4 kBit)

Synopsis Erasable programmable read-only memory (EEPROM)
23 [.]XXXXXXXXXXXX[XX][/[**memory** | **pages/page.[0-15|ALL]** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *23* DS2433

Special Properties

memory *read-write,* binary
512 bytes of memory. Initially all bits are set to 1. Writing zero permanently alters the memory.

pages/page.0 ... pages/page.15 pages/page.ALL *read-write,* yes-no
Memory is split into 16 pages of 32 bytes each. *ALL* is an aggregate of the pages. Each page is accessed sequentially.

Standard Properties

address

r_address *read-only,* ascii
The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).
address starts with the *family* code
r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii

The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).

r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii

Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.

If no **Link Locator** is between the device and the master, the *locator* field will be all FF.

r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no

Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii

Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms None.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data

caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2433 The **DS2433 (3)** is used for storing memory which should be available even after a reset or power off. It's main advantage is for audit trails (i.e. a digital purse). *OWFS* system handles this automatically.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://pdfserv.maxim-ic.com/en/ds/DS2433.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Christian Magnusson (E-Mail: mag@mag.cx)

A.1.6.11. DS2450

Name DS2450 - Quad A/D Converter

Synopsis

Voltage * 4 and Memory. **20** [.]XXXXXXXXXXXX[XX][/[**PIO.[A-D|ALL]** | **volt.[A-D|ALL]** | **volt2.[A-D|ALL]**]]
20 [.]XXXXXXXXXXXX[XX][/[**8bit/volt.[A-D|ALL]** | **8bit/volt2.[A-D|ALL]**]]
20 [.]XXXXXXXXXXXX[XX][/[**memory** | **pages/page.[0-3|ALL]** | **power**]
20 [.]XXXXXXXXXXXX[XX][/[**alarm/high.[A-D|ALL]** | **alarm/low.[A-D|ALL]** | **set_alarm/high.[A-D|ALL]** | **set_alarm/low.[A-D|ALL]** | **set_alarm/unset** | **set_alarm/volthigh.[A-D|ALL]** | **set_alarm/volt2high.[A-D|ALL]** | **set_alarm/voltlow.[A-D|ALL]** | **set_alarm/volt2low.[A-D|ALL]**]
20 [.]XXXXXXXXXXXX[XX][/[**address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

CO2 sensor **20** [.]XXXXXXXXXXXX[XX][/[**CO2/ppm** | **CO2/power** | **CO2/status**]

Family Code *20*

Special Properties

alarm/high.A ... alarm/high.D alarm.high.ALL

alarm/high.A ... alarm/high.D alarm.high.ALL *read-write,* binary
The alarm state of the voltage channel. The alarm state is set one of two ways:

voltage conversion Whenever the *DS2450* measures a voltage on a channel, that voltage is compared to the high and low limits *set_alarm/volthigh* and/or *set_alarm/voltlow* and if the alarm is enabled *set_alarm/high* and/or *set_alarm/low* the corresponding flag is set in *alarm/high* and/or *alarm/low*

manual set The flag can be set by a direct write to *alarm/high* or *alarm/low*

memory *read-write,* binary
32 bytes of data. Much has special implications. See the datasheet.

pages/page.0 ... pages/page.3 pages/page.ALL *read-write,* binary
Memory is split into 4 pages of 8 bytes each. Mostly for reading and setting device properties. See the datasheet for details.
ALL is an aggregate of the pages. Each page is accessed sequentially.

Pio.a ... Pio.d Pio.all *read-write,* yes-no
Pins used for digital control. 1 turns the switch on (conducting). 0 turns the switch off (non-conducting).
Control is specifically enabled. Reading *volt* will turn off this control.
ALL is an aggregate of the voltages. Readings are made separately.

power *read-write,* yes-no
Is the *DS2450* externally powered? (As opposed to parasitically powered from the data line).
The analog A/D will be kept on continuously. And the bus will be released during a conversion allowing other devices to communicate.
Set `true` (`1`) only if Vcc powered (not parasitically powered). Unfortunately, the *DS2450* cannot sense it's powered state. This flag must be explicitly written, and thus is a potential source of error if incorrectly set.
It is always safe to leave *power* set to the default 0 (off) state.

set_alarm/high.A ... set_alarm/high.D set_alarm/high.ALL

set_alarm/low.A ... set_alarm/low.D set_alarm/low.ALL *read-write,* yes-no
Enabled status of the voltage threshold. 1 is on. 0 is off.

set_alarm/volthigh.A ... set_alarm/volthigh.D set_alarm/volthigh.ALL

set_alarm/volt2high.A ... set_alarm/volt2high.D set_alarm/volt2high.ALL

set_alarm/voltlow.A ... set_alarm/voltlow.D set_alarm/voltlow.ALL

set_alarm/volt2low.A ... set_alarm/volt2low.D set_alarm/volt2low.ALL *read-write,* floating point

The upper or lower limit for the voltage measured before triggering an alarm.

Note that the alarm must be enabled *alarm/high* or *alarm.low* and an actual reading must be requested *volt* for the alarm state to actually be set. The alarm state can be sensed at *alarm/high* and *alarm/low*

set_alarm/unset *read-write,* yes-no

Status of the power-on-reset (POR) flag.

The POR is set when the *DS2450* is first powered up, and will match the alarm state until explicitly cleared. (By writing 0 to it).

The purpose of the POR is to alert the user that the chip is not yet fully configured, especially alarm thresholds and enabling.

volt.A ... volt.D volt.ALL

8bit/volt.A ... 8bit/volt.D 8bit/volt.ALL *read-only,* floating point

Voltage read, 16 bit resolution (or 8 bit for the *8bit* directory). Range 0 - 5.10V.

Output (*PIO*) is specifically disabled.

ALL is an aggregate of the voltages. Readings are made separately.

volt2.A ... volt2.D volt2.ALL

8bit/volt2.A ... 8bit/volt2.D 8bit/volt2.ALL *read-only,* floating point

Voltage read, 16 bit resolution (or 8 bit for the *8bit* directory). Range 0 - 2.55V.

Output (*PIO*) is specifically disabled.

ALL is an aggregate of the voltages. Readings are made separately.

CO2 (Carbon Dioxide) SENSOR PROPERTIES The CO2 sensor is a device constructed from a SenseAir K30 and a *DS2450*

CO2/power *read-only,* floating point

Supply voltage to the CO2 sensor (should be around 5V)

CO2/ppm *read-only,* unsigned

CO2 level in ppm (parts per million). Range 0-5000.

CO2/status *read-only,* yes-no

Is the internal voltage correct (around 3.2V)?

Standard Properties

address

r_address *read-only,* ascii
The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).
address starts with the *family* code
r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii
The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).
r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii
Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.
If no **Link Locator** is between the device and the master, the *locator* field will be all FF.
r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no
Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii
Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms None.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds2450 The **DS2450 (3)** is a (supposedly) high resolution A/D converter with 4 channels. Actual resolutin is reporterd to be 8 bits. The channels can also function as switches. Voltage sensing (with temperature and current, but sometimes restricted voltrage ranges) can also be obtained with the **DS2436** , **DS2438** and **DS276x**

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet

DS2450 http://pdfserv.maxim-ic.com/en/ds/DS2450.pdf

CO2 sensor http://www.senseair.se/Datablad/k30%20.pdf

CO2 device https://www.m.nu/co2meter-version-2-p-259.html?language=en

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Paul Alfille (E-Mail: paul.alfille@gmail.com)

A.1.6.12. DS28EC20

Name DS28EC20 - EEPROM (20 kBit)

Synopsis Erasable programmable read-only memory (EEPROM)
43 [.]XXXXXXXXXXXX[XX][/[**memory** | **pages/page.[0-79|ALL]** | **address** | **crc8** | **id** | **locator** | **r_address** | **r_id** | **r_locator** | **type**]]

Family Code *23* DS28EC20

Special Properties

memory *read-write,* binary
512 bytes of memory. Initially all bits are set to 1. Writing zero permanently alters the memory.

pages/page.0 ... pages/page.79 pages/page.ALL *read-write,* yes-no
Memory is split into 80 pages of 32 bytes each. *ALL* is an aggregate of the pages. Each page is accessed sequentially.

Standard Properties

address

r_address *read-only,* ascii
The entire 64-bit unique ID. Given as upper case hexidecimal digits (0-9A-F).
address starts with the *family* code
r address is the *address* in reverse order, which is often used in other applications and labeling.

crc8 *read-only,* ascii
The 8-bit error correction portion. Uses cyclic redundancy check. Computed from the preceding 56 bits of the unique ID number. Given as upper case hexidecimal digits (0-9A-F).

family *read-only,* ascii
The 8-bit family code. Unique to each *type* of device. Given as upper case hexidecimal digits (0-9A-F).

id

r_id *read-only,* ascii
The 48-bit middle portion of the unique ID number. Does not include the family code or CRC. Given as upper case hexidecimal digits (0-9A-F).
r id is the *id* in reverse order, which is often used in other applications and labeling.

locator

r_locator *read-only,* ascii

Uses an extension of the 1-wire design from iButtonLink company that associated 1-wire physical connections with a unique 1-wire code. If the connection is behind a **Link Locator** the *locator* will show a unique 8-byte number (16 character hexidecimal) starting with family code FE.

If no **Link Locator** is between the device and the master, the *locator* field will be all FF.

r locator is the *locator* in reverse order.

present (DEPRECATED) *read-only,* yes-no

Is the device currently *present* on the 1-wire bus?

type *read-only,* ascii

Part name assigned by Dallas Semi. E.g. *DS2401* Alternative packaging (iButton vs chip) will not be distiguished.

Alarms None.

Description

1-Wire *1-wire* is a wiring protocol and series of devices designed and manufactured by Dallas Semiconductor, Inc. The bus is a low-power low-speed low-connector scheme where the data line can also provide power.

Each device is uniquely and unalterably numbered during manufacture. There are a wide variety of devices, including memory, sensors (humidity, temperature, voltage, contact, current), switches, timers and data loggers. More complex devices (like thermocouple sensors) can be built with these basic devices. There are also 1-wire devices that have encryption included.

The 1-wire scheme uses a single *bus* master and multiple *slaves* on the same wire. The bus master initiates all communication. The slaves can be individually discovered and addressed using their unique ID.

Bus masters come in a variety of configurations including serial, parallel, i2c, network or USB adapters.

OWFS design *OWFS* is a suite of programs that designed to make the 1-wire bus and its devices easily accessible. The underlying priciple is to create a virtual filesystem, with the unique ID being the directory, and the individual properties of the device are represented as simple files that can be read and written.

Details of the individual slave or master design are hidden behind a consistent interface. The goal is to provide an easy set of tools for a software designer to create monitoring or control applications. There are some performance enhancements in the implementation, including data caching, parallel access to bus masters, and aggregation of device communication. Still the fundemental goal has been ease of use, flexibility and correctness rather than speed.

Ds28ec20 The **DS28EC20 (3)** is used for storing memory which should be available even after a reset or power off. It's main advantage is for audit trails (i.e. a digital purse). *OWFS* system handles this automatically.

Addressing All 1-wire devices are factory assigned a unique 64-bit address. This address is of the form:

Family Code 8 bits

Address 48 bits

CRC 8 bits

Addressing under OWFS is in hexidecimal, of form:

01.123456789ABC

where **01** is an example 8-bit family code, and **12345678ABC** is an example 48 bit address. The dot is optional, and the CRC code can included. If included, it must be correct.

Datasheet http://datasheets.maxim-ic.com/en/ds/DS28EC20.pdf

See Also

Programs **owfs (1)** owhttpd (1) owftpd (1) owserver (1) **owdir (1)** owread (1) owwrite (1) owpresent (1) **owtap (1)**

Configuration and testing **owfs (5)** owtap (1) owmon (1)

Language bindings **owtcl (3)** owperl (3) owcapi (3)

Clocks **DS1427 (3)** DS1904(3) DS1994 (3) DS2404 (3) DS2404S (3) DS2415 (3) DS2417 (3)

ID **DS2401 (3)** DS2411 (3) DS1990A (3)

Memory **DS1982 (3)** DS1985 (3) DS1986 (3) DS1991 (3) DS1992 (3) DS1993 (3) DS1995 (3) DS1996 (3) DS2430A (3) DS2431 (3) DS2433 (3) DS2502 (3) DS2506 (3) DS28E04 (3) DS28EC20 (3)

Switches **DS2405 (3)** DS2406 (3) DS2408 (3) DS2409 (3) DS2413 (3) DS28EA00 (3)

Temperature **DS1822 (3)** DS1825 (3) DS1820 (3) DS18B20 (3) DS18S20 (3) DS1920 (3) DS1921 (3) DS1821 (3) DS28EA00 (3) DS28E04 (3) EDS0064 (3) EDS0065 (3) EDS0066 (3) EDS0067 (3) EDS0068 (3) EDS0071 (3) EDS0072 (3)

Humidity **DS1922 (3)** DS2438 (3) EDS0065 (3) EDS0068 (3)

Voltage **DS2450 (3)**

Resistance **DS2890 (3)**

Multifunction (current, voltage, temperature) **DS2436 (3)** DS2437 (3) DS2438 (3) DS2751 (3) DS2755 (3) DS2756 (3) DS2760 (3) DS2770 (3) DS2780 (3) DS2781 (3) DS2788 (3) DS2784 (3)

Counter **DS2423 (3)**

LCD Screen **LCD (3)** DS2408 (3)

Crypto **DS1977 (3)**

Pressure **DS2406 (3)** – TAI8570 EDS0066 (3) EDS0068 (3)

Availability http://www.owfs.org/

Author Christian Magnusson (E-Mail: mag@mag.cx)

Abbildungsverzeichnis

Tabellenverzeichnis

Index