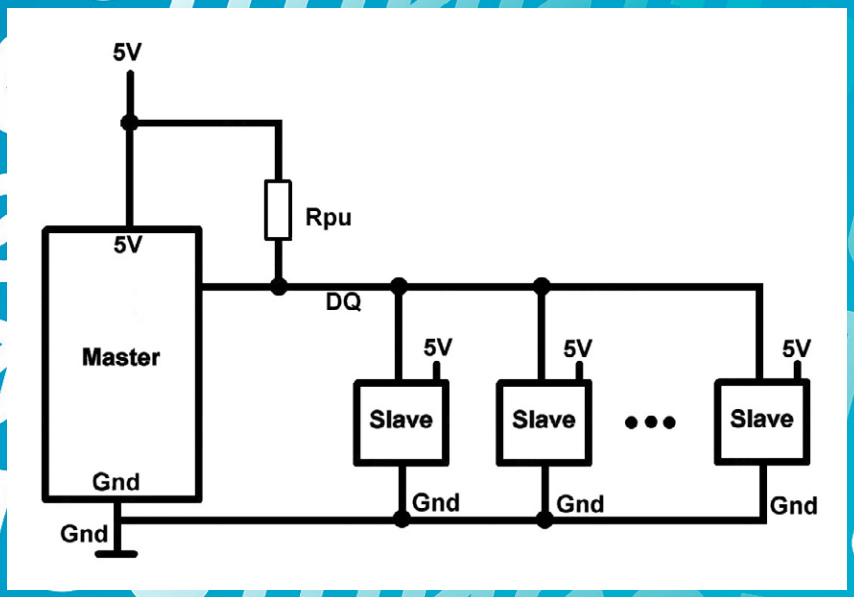
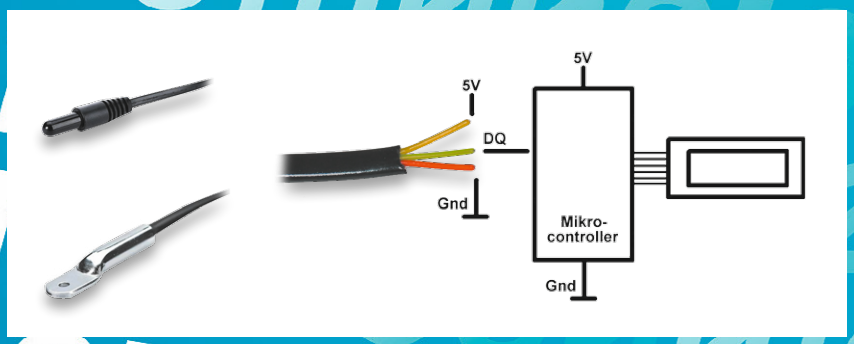
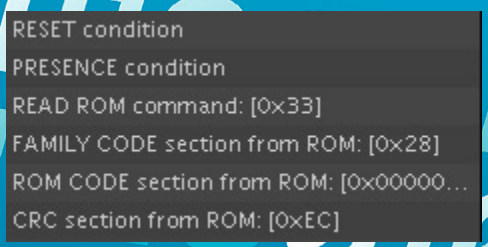
Digitale Hardwareschnittstellen

Teil 4: 1-Wire







Die 1-Wire-Schnittstelle wurde von der Firma Maxim Integrated Products (vormals Dallas Semiconductor) entwickelt und definiert sowohl den Busaufbau als auch das Protokoll. Es handelt sich um eine asynchrone serielle Verbindung zwischen einem Master und nahezu beliebig vielen Slaves. Der Name 1-Wire wurde gewählt, weil die Datenverbindung mit lediglich einer Datenleitung (zusätzlich zur Gnd-Verbindung) auskommt. Von einer seriellen Verbindung wird gesprochen, weil die Bits hintereinander übertragen werden. Die Übertragung heißt asynchron, weil es keine separate Taktleitung zwischen dem Master und dem Slave oder den Slaves gibt, sondern die Kommunikation zwischen Master und Slave(s) durch eine detaillierte Spezifikation der zeitlichen Abläufe geregelt wird. Die Verbindung erfolgt bidirektional, das heißt, dass Daten sowohl vom Master zum Slave als auch umgekehrt übertragen werden können.

Es gibt eine große Anzahl verschiedener Slaves für unterschiedliche Zwecke. Am weitesten verbreitet sind die Temperatursensoren DS1820 bzw. dessen Nachfolger DS18S20 und der aktuellere DS18B20 mit der Möglichkeit, die Auflösung zu konfigurieren und damit die Messzeit zu verkürzen. Der DS18S20 kann als direkter Ersatz des ursprünglichen DS1820 verwendet werden, während der modernere DS18B20 in neuen Entwicklungen bevorzugt werden sollte. Beide Temperatursensoren (DS1820/DS18S20 und DS18B20) müssen unterschiedlich ausgewertet werden, aber da sie per Software anhand unterschiedlicher Kennungen (Family-Codes – siehe unten) unterscheidbar sind, ist auch ein gemischter Betrieb an einem Bus leicht möglich. Unterschiede zwischen den Sensoren sind in [1] beschrieben.

Außer den genannten Temperatursensoren gibt es für 1-Wire auch adressierbare elektronische Schalter (DS2413, DS2408), digitale Potenziometer (DS1805), Analog-Digital-Konverter (DS2450), 4k-RAM mit Zähler (DS2423), Batteriemonitor (DS2438), Echtzeituhren (DS2417), Speicher-ICs (DS24B33, DS1961), weitere Temperaturbausteine (DS28EA00, DS1921), eindeutige Silicon-Seriennummer (DS2401) und so weiter. Einige der 1-Wire-Slaves werden in einem Gehäuse angeboten, das einer Knopfzelle sehr ähnlich sieht – sogenannte iButtons –, bei denen Ober- und Unterseite je einen Kontakt zu dem eingebauten Baustein darstellt. Die meisten Bausteine sehen allerdings aus wie ein Transistor (meist im TO-92- oder SOT-223-Gehäuse) oder werden als Temperatursensoren inklusive angeschlossenem Kabel und in speziellen Gehäusen angeboten.

Busaufbau

Wie Bild 1 zeigt, besteht ein 1-Wire-Bus aus genau einem Master, der auch stets die Kommunikation anstößt, und einem oder vielen Slaves. Der Master und die Slaves müssen eine gemeinsame Gnd-Leitung haben. Die Datenübertragung erfolgt auf einer Leitung mit der Bezeichnung DQ, an die jeder Slave angeschlossen ist. Für den gesamten Bus muss es genau einen Pull-up-Widerstand R_{PU} von ca. 4,7 kΩ geben, der die Datenleitung nach Plus (V_{CC}) zieht. Jeder einzelne 1-Wire-Teilnehmer

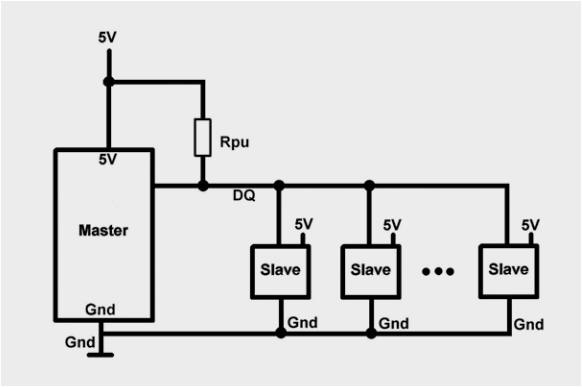


Bild 1: 1-Wire-Bus mit externer Spannungsversorgung

ist – wie in Bild 2 schematisch dargestellt – mit einer Open-Drain-Schaltung aufgebaut. Wenn ein 1-Wire-Baustein einen Spannungszustand auf die Datenleitung geben soll, zieht er den Spannungspegel entweder nach Gnd, indem der eingebaute MOSFET leitend geschaltet wird, oder er lässt die Datenleitung durch den Pull-up-Widerstand nach Plus (V_{CC}) ziehen. Zum Erkennen der an der Datenleitung anliegenden Spannung „lauscht" der 1-Wire-Baustein (mit Rx, Bild 2) auf den Bus. Der 1-Wire-Bus kann auf jeden Fall 20 bis 30 Meter lang sein. Manchmal liest man sogar von bis zu 200 Metern.

Die meisten 1-Wire-Bausteine ermöglichen eine Spannungsversorgung über eine dedizierte Leitung für die Betriebsspannung (typisch 5 Volt oder 3,3 Volt) oder aber alternativ eine sogenannte parasitäre Spannungsversorgung, bei der es keine explizite Spannungsversorgung gibt, sondern die Spannung aus der Datenleitung gewonnen wird, indem ein Kon-

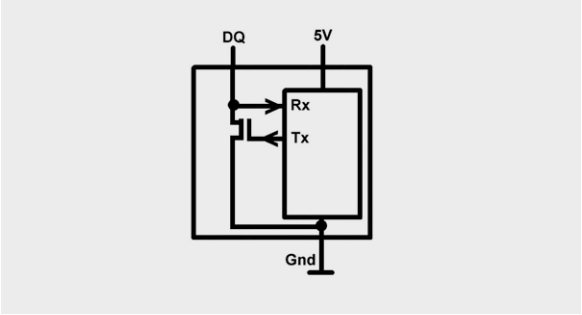


Bild 2: 1-Wire-Slave mit Open Drain

8-BIT CRC	48-BIT SERIAL NUMBER	8-BIT FAMILY CODE (28h)
MSB		LSB

Bild 3: 64-Bit ROM-Code

densator im Innern des Slaves sich während der Ruhezeiten (Spannung auf der Datenleitung durch den Pull-up-Widerstand auf V_{CC}) auflädt und die gespeicherte Spannung für den Betrieb wieder abgibt.

Bei externer Spannungsversorgung benötigt man für die Spannungsversorgung zwei Leitungen und für die Datenübertragung eine dritte Leitung. Bei einer parasitären Spannungsversorgung benötigt man außer einer Gnd-Leitung nur die Datenleitung, kommt also insgesamt mit nur zwei Leitungen aus – was auch die Bauform als iButtons ermöglicht. Bei der Bezeichnung 1-Wire wird die Gnd-Leitung nicht mitgezählt.

Adressierung

Jeder Slave am 1-Wire-Bus hat eine 64 Bit (8 Byte) lange ID – den sogenannten ROM-Code (Bild 3). 8 Bit des ROM-Codes sind der sogenannte Family-Code (Gerätefamilie), der den Typ des 1-Wire-Bausteins darstellt (zum Beispiel 28h für einen DS18B20 und 10h für einen DS18S20 oder DS1820) und per Programm abgefragt und ausgewertet werden kann. Dadurch kann man beispielsweise verschiedenartige Temperatursensoren an einem 1-Wire-Bus haben und im Programm die jeweils passende Umwandlungsroutine einsetzen. Die nächsten 48 Bit des ROM-Codes sind eine für den betreffenden Baustein weltweit eindeutige Identifikationsnummer. Mit diesen 48 Bit lassen sich 2 hoch 48 = 281.474.976.710.656 verschiedene 1-Wire-Bausteine eindeutig identifizieren!

Anders als zum Beispiel bei der I²C-Schnittstelle, wo teilweise nur 8 gleichartige Bausteine an einem Bus angeschlossen sein können, ist es dadurch bei 1-Wire problemlos möglich, zum Beispiel 30 gleichartige Temperatursensoren an einem 1-Wire-Bus angeschlossen zu haben und einzeln anzusprechen bzw. abzufragen. Außerdem enthält der ROM-Code einen 8 Bit langen Prüfcode (CRC = Cyclic Redundancy Check), mit dem die korrekte Übertragung der vorangegangenen 56 Bits des ROM-Codes geprüft werden kann.

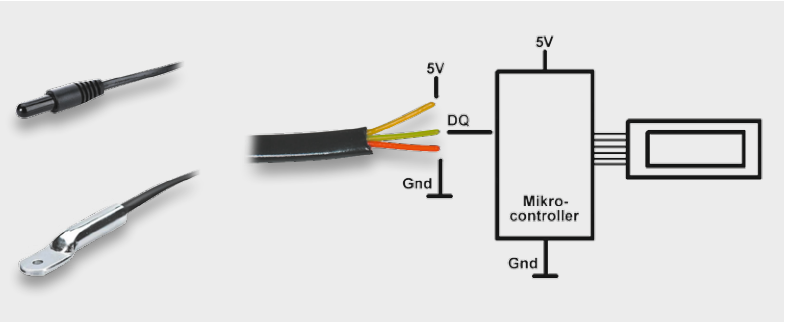


Bild 4: Gehäuseformen und Anschluss

DS18B20-Temperatursensor

Die bekannteste Anwendung des 1-Wire-Busses ist die Temperaturmessung mit DS18B20 (CN-10 93 37, CN-10 27 83) oder den älteren DS18S20. Deshalb wird die Arbeit mit 1-Wire im Folgenden anhand des DS18B20 erläutert. Mit einem DS18B20 lässt sich die Temperatur im Bereich von –55 °C bis +125 °C messen, wobei die Temperaturmessung zwischen –10 °C und +85 °C mit einer Genauigkeit von 0,5 °C erfolgt. (Die Genauigkeit gibt an, wie genau die vom Sensor dargestellte Temperatur der wahren Temperatur entspricht.) Die Auflösung lässt sich zwischen 9 und 12 Bit einstellen. (Die Auflösung gibt an, wie fein abgestuft eine analoge Größe digital dargestellt wird.) Eine Temperaturumwandlung mit der maximalen 12-Bit-Auflösung erfolgt in 750 ms. Die Spannungsversorgung kann zwischen 3 Volt und 5,5 Volt sein, wobei wie oben beschrieben eine parasitäre Spannungsversorgung zum Einsatz kommen kann und dann außer der Gnd-Leitung nur eine einzige (Daten-) Leitung benötigt wird. Mit einem zweiadrigen Kabel kann man also verschiedene Sensoren an verschiedenen Orten an einem Bus einsetzen. Bild 4 zeigt links zwei verschiedene von ELV angebotene Gehäuseformen des DS18B20 – für Flüssigkeiten geeignet bzw. mit metallischer Befestigungslasche. Rechts im Bild sieht man den Anschluss an einen Mikrocontroller. Bild 4 zeigt die nicht parasitäre Spannungsversorgung.

Der Blick auf Bild 5 zeigt die verschiedenen Blöcke im Innern des DS18B20. Der linke Block erkennt, ob der parasitäre oder der normale Spannungsversorgungsmodus verwendet wird. In diesem Block ist auch der Kondensator angedeutet, der bei parasitärer Spannungsversorgung für die Speicherung der Spannung zum Einsatz kommt. Der zweite Block zeigt den Speicherbereich für den 64-Bit-Code, der wie oben beschrieben den Family-Code beinhaltet und eine eindeutige Adressierung des Bausteines ermöglicht. Der Inhalt dieses Speicherbereiches lässt sich nicht durch den Benutzer verändern (ROM = Read Only Memory). In dem Speicherbereich mit dem Namen Scratchpad sind die Temperaturdaten und ein Konfigurationsregister gespeichert (vgl. auch Bild 11).

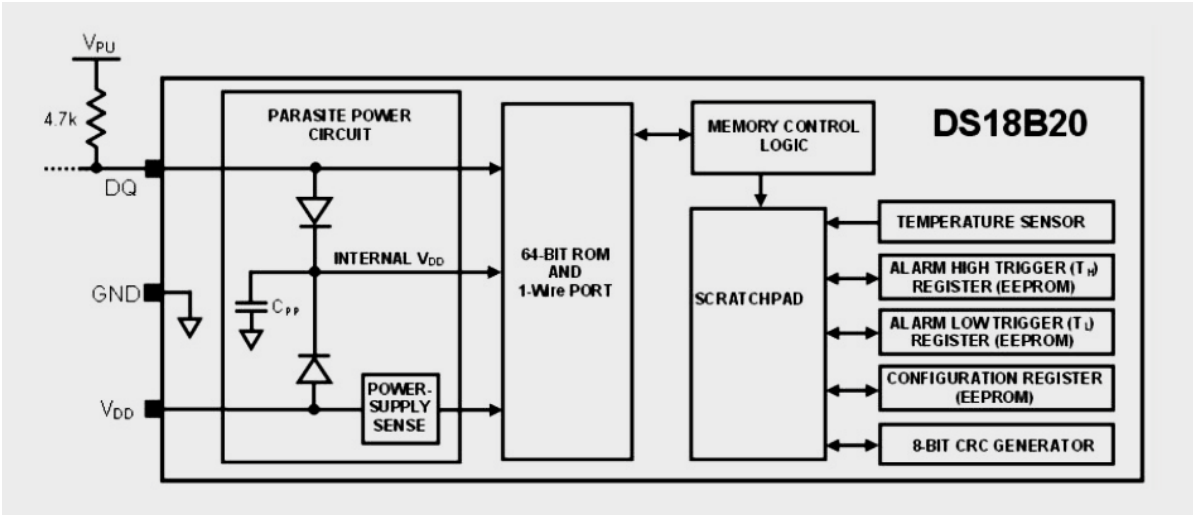


Bild 5: Aufbau DS18B20 Temperatursensor (Datenblattauszug)

1. ROM Commands (Adressierung)			2. Function Commands (Befehl)		
33	Read ROM	ROM Code eines Slaves auslesen	44	Convert T	Temperaturmessung anstoßen
F0	Search ROM	ROM Codes mehrerer Slaves auslesen	BE	Read Scratchpad	Scratchpad lesen
CC	Skip ROM	Alle Slaves ansprechen	4E	Write Scratchpad	Scratchpad beschreiben (TH, TL und Konfigurationsregister)
55	Match ROM	Nur einen bestimmten Slave ansprechen	B4	Read Power Supply	Art der Spannungsversorgung auslesen
EC	Alarm Search	Slaves mit gesetztem Alarmflag ansprechen	48	Copy Scratchpad	TH, TL und Konfigurationsregister in EEPROM schreiben
			B8	Recall	TH, TL, Konfigurationsregister aus EEPROM in Scratchpad

Tabelle 1: Kommandos

Im Scratchpad gibt es auch je ein Register für Temperaturmaximum und Temperaturminimum, mit deren Hilfe der DS18B20 als Thermostat eingesetzt werden kann, indem er bei Über- bzw. Unterschreiten eines definierten Temperaturwertes einen Alarm signalisiert, der vom Benutzerprogramm ausgewertet werden kann. Rechts im Blockbild sieht man EEPROM-Speicherbereiche, deren Inhalte auch ohne Spannung am Sensor gespeichert bleiben. Dadurch kann man die Konfiguration und den Maximum-/Minimumwert für den Thermostat dauerhaft speichern. (Tipp: Wenn man die Thermostatfunktion nicht benötigt, kann man die beiden letzteren Register (TH und TL) auch zur dauerhaften Speicherung von zwei beliebigen (Byte-großen) Werten benutzen!)

Protokoll

Das Ansprechen eines 1-Wire-Slaves erfolgt immer nach dem gleichen Schema: Zunächst wird ein ROM Command (Adressierung) vom Master auf den Bus gesendet, welches angibt, welche Slaves angesprochen werden, und dann wird ein Function Command (Befehl) vom Master gesendet, welcher angibt, was auf Slave-Seite getan werden soll. Tabelle 1 zeigt die möglichen Kommandos des DS18B20. Ein Standardablauf ist beispielsweise, dass zunächst der hexadezimale Wert CC vom Master auf den Bus gesendet wird, was angibt, dass alle Slaves gleichzeitig angesprochen werden sollen, und danach vom Master eine hexadezimale 44 auf den Bus gesendet wird, wodurch alle Slaves dazu veranlasst werden, eine Temperaturmessung vorzunehmen. Eine weitere Möglichkeit wäre, dass der Master eine hexadezimale 33 auf den Bus gibt, was den (einen) angeschlossenen Slave dazu veranlassen würde, seinen ROM-Code auf den Bus zu geben. Diese und weitere Kombinationen von ROM Command und Function Command werden im Folgenden detailliert analysiert.

ROM-Code auslesen

Beim Aufbau einer Anlage ist es sinnvoll, zunächst zu prüfen, wie die ROM-Codes der teilnehmenden Temperatursensoren sind. Dazu schließt man *einen* DS18B20 am Bus an und der Master gibt eine hexadezimale 33 (Read ROM Kommando) auf den Bus. Eine genaue Analyse mit einem Logikanalysator (Bild 6 und der Blick in das Datenblatt [2]) zeigt, dass am Anfang jeder 1-Wire-Kommunikation eine sogenannte Reset Condition vom Master auf den Bus gegeben wird, die von Slave-Seite mit einem „Presence" beantwortet wird. Der Signalverlauf aus Bild 6 wird nun anhand Bild 7 im Einzelnen besprochen: Im Ruhezustand liegt der Bus durch den Pull-up-Widerstand auf logisch High. Um eine Kommunikation zu beginnen, zieht der Master den Spannungspegel auf der Datenleitung für mindestens 480 µs auf Low (Reset Condition/Reset Pulse) und gibt den Bus dann wieder frei (High durch Pull-up-Widerstand). Der Slave wartet 15 bis 60 µs (damit saubere Signale vorliegen) und zieht dann den Bus seinerseits für 60 µs bis 240 µs auf Low und signalisiert damit seine Bereitschaft zur Kommunikation (vgl. Abbildung 13 im Datenblatt [2]).

Danach kann der Master nun das Read-ROM-Kommando (eine hexadezimale 33) auf den Bus geben.

Bild 6: Signalverlauf ROM-Code lesen (Gesamtdarstellung)

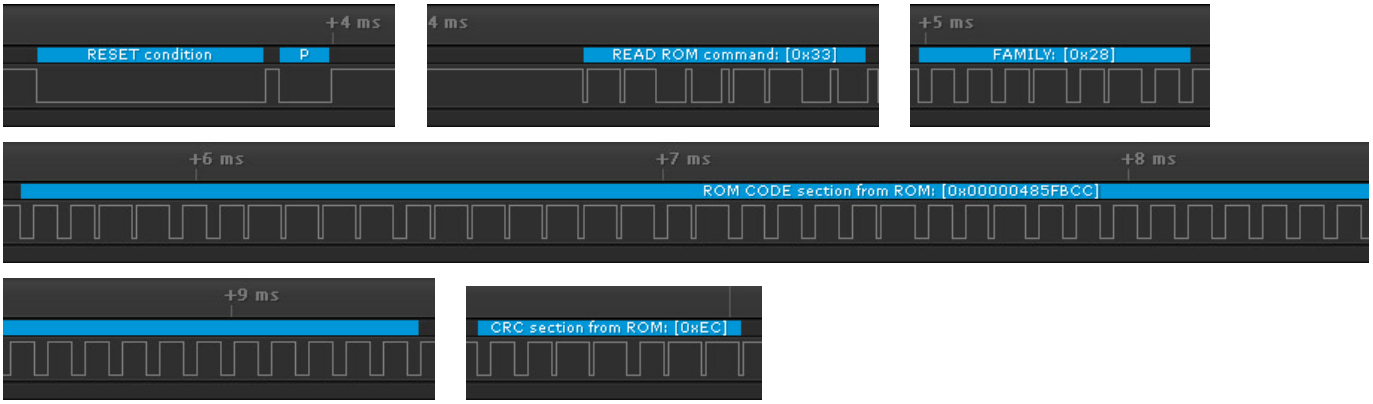


Bild 7: Signalverlauf ROM-Code lesen (in Teile zerlegt)

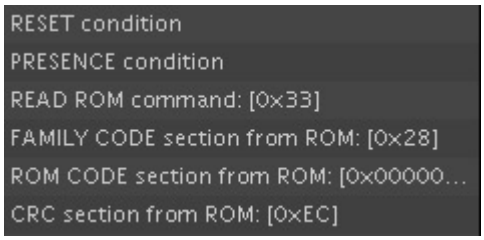


Bild 8: Dekodierte Zeichen: ROM-Code lesen

Das macht der Master, indem er den Bus für eine 1 für maximal 15 µs und für eine 0 für mindestens 60 µs auf Low zieht. In Bild 7 kann man im zweiten Teilbild die seriell vom Master gesendete Bitfolge 1 1 0 0 1 1 0 0 erkennen. Wenn man nun (aus dem Datenblatt) weiß, dass stets das niedrigstwertige Bit zuerst übertragen wird, dann kann man diese Bitfolge manuell als 33 dekodieren (weil 0011 0011 binär = 33 hexadezimal ist). Der Slave wertet im Zeitraum zwischen 15 µs und 60 µs nach einer fallenden Flanke aus, ob ein High-Pegel (1) oder ein Low-Pegel (0) am Bus anliegt (Abbildung 14 oben im Datenblatt [2]).

Nachdem der Slave nun diese 33h empfangen hat, gibt der Slave den Family-Code auf den Bus (drittes Teilbild in Bild 7). Dafür zieht der Master für jedes Bit den Bus für mindestens 1 µs auf Low und gibt ihn dann wieder frei, woraufhin der Slave für eine logische 1 den Bus sofort auf High gehen lässt und für eine logische 0 den Bus für ca. 14 µs auf Low gezogen hält, was der Master jeweils erkennen kann (Abbildung 14 im Datenblatt [2]). Im Beispiel wird als Family-Code seriell 0 0 0 1 0 1 0 0 übertragen. Da wiederum das niedrigstwertige Bit zuerst übertragen wurde, wurde also 0010 1000, also 28h übertragen. Nach dem Family-Code 28h wird die einzigartige 48 Bit lange Seriennummer in gleichartiger Technik vom Slave an den Master signalisiert. Abschließend wird, wie in Bild 7 im letzten Teilbild zu sehen, die Prüfsumme (CRC) über die vorangegangenen Bits übertragen.

Noch einmal zur Übung: In diesem Fall sieht man am Signalverlauf, dass die Bits 0 0 1 1 0 1 1 1 übertragen werden. Da wieder „rückwärts", also das niedrigstwertige Bit zuerst übertragen wird, handelt es sich um 1110 1100 = ECh. Damit ist eine komplette Kommunikation zwischen Master und Slave beendet. Der Spannungspegel auf dem Bus wird durch den Pull-up-Widerstand auf High gezogen gehalten. Der Master kennt nun den ROM-Code des angeschlossenen Slaves. Die komplette Kommunikation aus Bild 6 sieht man in Bild 8 noch einmal in Form von (vom Logikanalysator) dekodierten Zeichen: RESET vom Master – PRESENCE vom Slave – Read ROM Kommando vom Master – Family-Code vom Slave – Seriennummer vom Slave – CRC vom Slave.

Temperatur messen und auslesen – ein Sensor

Um die Temperatur mit einem angeschlossenen DS18B20 messen zu lassen und auszulesen, wird zunächst vom Master mit CCh 44h jeder angeschlossene DS18B20 veranlasst, die Temperatur zu messen (vgl. Tabelle 1). Danach muss man mindestens 750 ms (bei höchster Auflösung) warten, bis die Temperatur intern im DS18B20 bereitsteht. Nach mindestens 750 ms (bei 12-Bit-Auflösung) kann der Master dann CCh BEh (vgl. Tabelle 1) auf den Bus geben, um alle (in diesem Fall den einzigen) DS18B20 zum Übertragen der Temperatur auf den Bus aufzufordern. Die Temperatur ist in einem Speicherbereich mit dem Namen Scratchpad im DS18B20 gespeichert. Bild 9 zeigt den Signalverlauf der zwei Schritte. Im oberen Teilbild ist zu sehen, dass die Kommunikation wie oben beschrieben wieder mit Reset vom Master und Presence vom Slave beginnt. Danach gibt der Master CCh 44h auf den Bus, wodurch alle angeschlossenen DS18B20 angewiesen werden, die Temperatur zu messen. Nach Ablauf von mindestens 750 ms (bei 12-Bit Auflösung) erfolgt eine zweite Kommunikation (Bild 9 unten), die wiederum mit Reset/Presence beginnt. Der Master sendet dann CCh BEh woraufhin der Slave die Bits des Scratchpads nacheinander auf den Bus gibt. Bild 10 zeigt die übertragenen dekodierten Daten auf dem 1-Wire-Bus. Das Scratchpad besteht aus 9 Bytes (Bild 11).

Der Wert der Temperatur befindet sich in Byte 0 und Byte 1. Bild 12 zeigt den Aufbau der Temperaturregister. Die Bits 15 bis 11 stellen das Vorzeichen der Temperatur dar. Die restlichen Bits stellen den Wert der Temperatur dar. Bild 13 zeigt beispielhafte Temperaturwerte. Um die Fehlerfreiheit der übertragenen Bits zu überprüfen, kann man die Prüfsumme (CRC) aus Byte 8 benutzen.

Read Power Mode

Wie oben beschrieben kann man einen DS18B20 (wie auch andere 1-Wire Slaves) „normal" (mit einer positiven Spannung und Gnd) oder im parasitären Spannungsversorgungsmodus (Spannung wird aus Datenleitung gewonnen) betreiben. Der Master kann per Software abfragen, in welcher der beiden Betriebsarten sich ein angeschlossener Slave befindet.

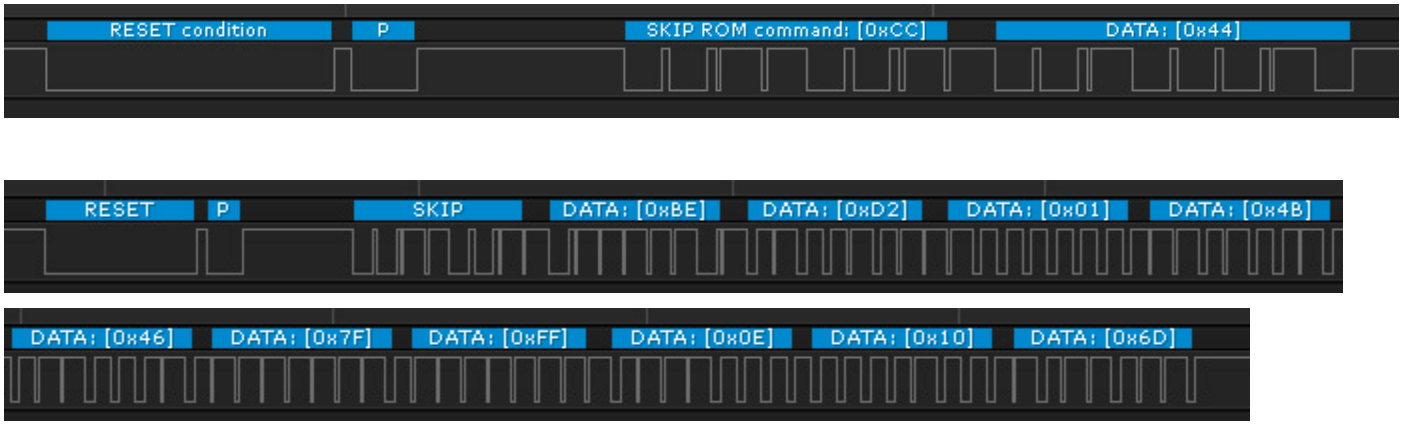


Bild 9: Temperaturmessung anstoßen (oben) und Temperatur auslesen (unten) (Signalverlauf)

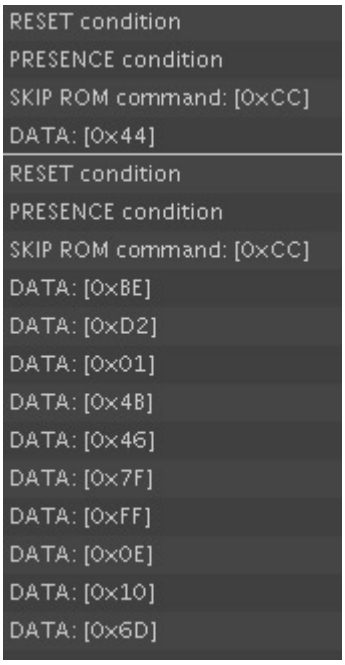


Bild 10: Temperaturmessung anstoßen und auslesen (dekodierter Text)

SCRATCHPAD

Byte 0	Temperature LSB
Byte 1	Temperature MSB
Byte 2	T_H Register or User Byte 1
Byte 3	T_L Register or User Byte 2
Byte 4	Configuration Register
Byte 5	Reserved (FFh)
Byte 6	Reserved
Byte 7	Reserved (10h)
Byte 8	CRC

Bild 11: DS18B20 Memory Map (Scratchpad) (Datenblattauszug)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Bild 12: Temperaturregister DS18B20 (Datenblattauszug)

Er sendet dafür (bei einem angeschlossenen Slave) CCh B4h (vgl. Tabelle 1) an den Slave, woraufhin der Slave seinen Spannungsversorgungsmodus zurückgibt. FFh steht für nichtparasitäre/konventionelle Spannungsversorgung und 00h steht für parasitäre Spannungsversorgung. In Bild 14 ist der Signalverlauf der Kommunikation zu sehen.

Die Kommunikation beginnt wie immer mit einem RESET vom Master und einer Bestätigung (PRESENCE Condition) vom Slave. Danach folgt CCh (Skip ROM) und Read Power Supply B4h vom Master und die Antwort vom Slave (FFh bzw. 00h). Bild 15 zeigt die dekodierte Kommunikation.

TEMPERATURE (°C)	DIGITAL OUTPUT
+125	0000 0111 1101 0000
+85	0000 0101 0101 0000
+25.0625	0000 0001 1001 0001
+10.125	0000 0000 1010 0010
+0.5	0000 0000 0000 1000
0	0000 0000 0000 0000
-0.5	1111 1111 1111 1000
-10.125	1111 1111 0101 1110
-25.0625	1111 1110 0110 1111
-55	1111 1100 1001 0000

Bild 13: Temperaturwerte (Beispiele) (Datenblattauszug)

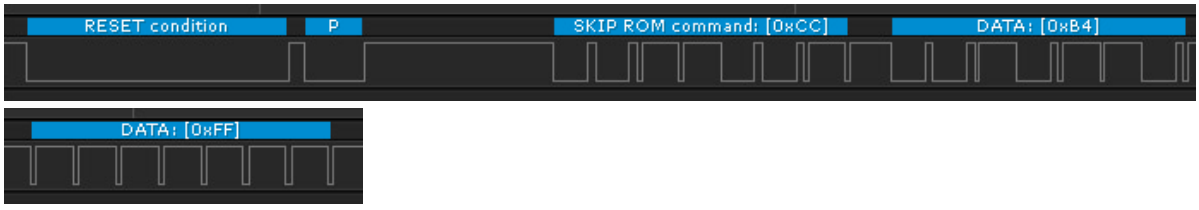


Bild 14: Signalverlauf Power Mode (geteilt)

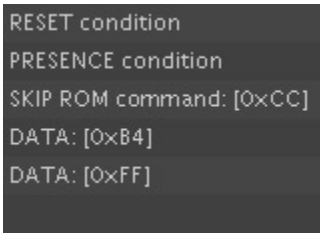


Bild 15: Power Mode (dekodiertes Signal)

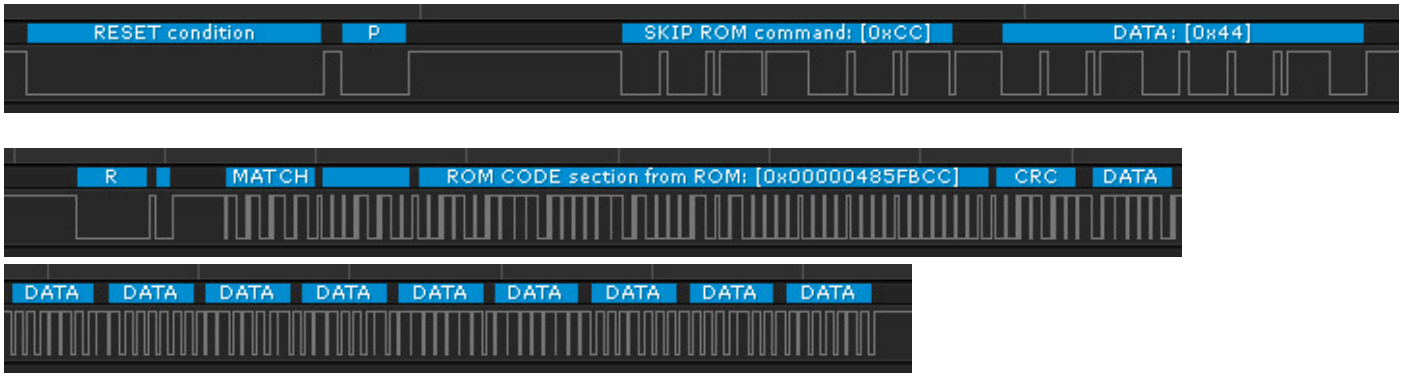


Bild 16: Signalverlauf Match ROM (unten geteilt)

Match ROM

In den bisherigen Beispielen war jeweils nur ein DS18B20 Sensor am Bus angeschlossen. In den meisten Fällen hat man mehrere 1-Wire-Slaves am Bus angeschlossen, die man direkt über die jeweilige eindeutige Seriennummer aus dem ROM-Code anspricht. Selbstverständlich können die Slaves nicht nur mehrere DS18B20-Temperatursensoren sein, sondern es kann auch eine Mischung aus DS18B20 und DS18S20 oder auch völlig anderen 1-Wire-Slaves sein. Über die Seriennummer und den Family-Code (beides im ROM-Code) ist jeder Slave eindeutig identifizierbar.

Bild 16 zeigt die Kommunikation für das Anstoßen der Temperaturmessung aller am Bus befindlichen Sensoren (CCh 44h) und das anschließende Auslesen der Temperatur eines bestimmten Sensors. Dieser Ablauf ist in der Praxis sehr nützlich, weil man im Programm oft einen Ein-Sekunden-Interrupt hat, an dessen Ende man einfach pauschal ALLE Sensoren die Temperatur messen lässt und an dessen Anfang man gezielt die verschiedenen bekannten Sensoren (Außentemperatur, Innentemperatur 1, Innentemperatur 2 o. Ä.) auslesen und auswerten kann. In Bild 17 sieht man die dekodierte Kommunikation.

Die hier am DS18B20 gezeigten Beispiele von 1-Wire-Abläufen lassen sich auf andere 1-Wire-Bausteine übertragen. Der Ablauf ist immer gleich und die Details (Befehle, Register usw.) müssen dem jeweils verwendeten Slave-Datenblatt entnommen werden. Die meisten Programmiersprachen bieten 1-Wire-Libraries bzw. 1-Wire-Befehle für die einfache Einbindung von 1-Wire-Bausteinen an. So kann man elegant die Anzahl aller angeschlossenen 1-Wire-Slaves zählen lassen, die ROM-Codes aller angeschlossenen Slaves anzeigen lassen, Daten zu den Slaves senden und Daten von den Slaves empfangen. Da das 1-Wire-Timing exakt definiert ist, kann man sogar spezialisierte eigene Slaves programmieren, die sich an einen 1-Wire-Bus hängen lassen.

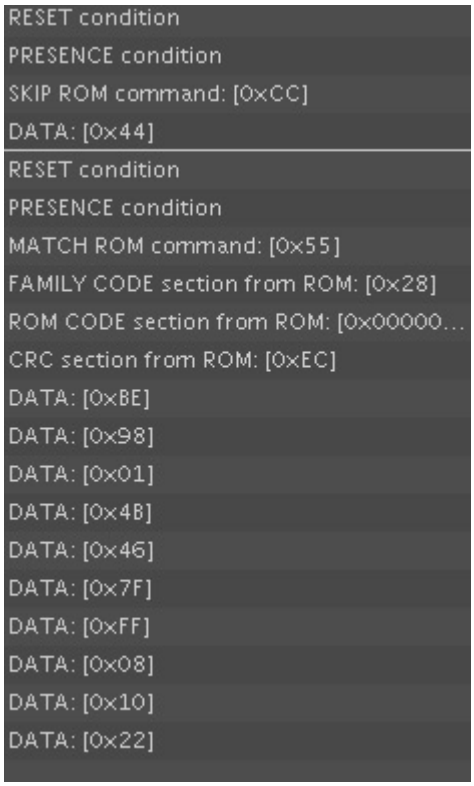


Bild 17: Dekodiertes Signal Match ROM

Fazit

Die 1-Wire-Schnittstelle ist eine gut verständliche asynchrone serielle Schnittstelle, die es ermöglicht, sehr viele – auch unterschiedliche – Slaves an einem Bus aus 3 bzw. sogar 2 Leitungen an unterschiedlichen Stellen im Labor oder im Haus über weltweit einzigartige Seriennummern anzusprechen. Im nächsten Teil dieser Artikelserie geht es um die Grundlagen der USB-Schnittstelle, die bekanntermaßen eine enorme Verbreitung hat. ELV

Weitere Infos:

[1] DS18B20 vs. DS18S20/DS1820: http://www.maximintegrated.com/en/app-notes/index.mvp/id/4377
[2] Datenblatt DS18B20: http://www.elv.de: Webcode #10096

Preisstellung April 2017 – aktuelle Preise im ELV Shop

Empfohlene Produkte	Best.-Nr.	Preis
H-Tronic TS 1 Temperatursensor DS18B20	CN-10 93 37	€ 12,95
H-Tronic TS 2 Temperatursensor DS18B20	CN-10 27 83	€ 14,95
Kupplung für RJ45-Stecker	CN-10 65 34	€ 1,–
ISDN-Kabel mit RJ45-Steckern	CN-10 65 35	€ 6,50
Delock Adapter Terminalblock mit Drucktaste > RJ45 Buchse	CN-11 52 88	€ 5,95
Modular-Einbaubuchse 2x RJ45	CN-11 21 47	€ 0,88