

**kHome Protokollspezifikation**

Version 0.3

Peter Kappelt

letzte Änderung: 25.03.2017 16:49

# Inhalt

[1 Inhalt 3](#_Toc478223885)

[2 Aufbau eines kHome-Telegramms 5](#_Toc478223886)

[2.1 Medien 5](#_Toc478223887)

[2.1.1 kHome RF 5](#_Toc478223888)

[2.1.2 kHome Serial 5](#_Toc478223889)

[2.2 Standardtelegramm 5](#_Toc478223890)

[2.2.1 Protokolltyp 5](#_Toc478223891)

[2.2.2 Telegrammtyp 6](#_Toc478223892)

[2.2.2.1 REG\_W 6](#_Toc478223893)

[2.2.2.2 REG\_R 7](#_Toc478223894)

[2.2.2.3 REG\_B 8](#_Toc478223895)

[2.2.2.4 CNF\_W 8](#_Toc478223896)

[2.2.2.5 CNF\_R 9](#_Toc478223897)

[2.2.2.6 STS\_R 10](#_Toc478223898)

[2.2.2.7 ANS 11](#_Toc478223899)

[2.2.3 Adresse des Senders 12](#_Toc478223900)

[2.2.4 Adresse des Empfängers 12](#_Toc478223901)

[2.2.5 Nutzdaten 12](#_Toc478223902)

[2.2.6 CRC 12](#_Toc478223903)

[3 Device Files 14](#_Toc478223904)

[3.1 XML-Syntax vom Device Files 14](#_Toc478223905)

[3.2 Beispiel für eine Device File 15](#_Toc478223906)

[3.3 Software 16](#_Toc478223907)

[3.3.1 Erstellen und Bearbeiten von Device-Files 16](#_Toc478223908)

[3.3.2 Report-/ Code-Generatoren 18](#_Toc478223909)

[4 Versionsgeschichte 20](#_Toc478223910)

# Aufbau eines kHome-Telegramms

Ein kHome-Telegramm kann über verschiedene Protokolle und Hardwareschichten übertragen werden. Das bedeutet, dass der folgende Aufbau für den innersten Kern eines Telegramms steht. Je nach verwendetem Übertragungsmedium können neben aufgezeigtem Telegramm noch Präfixe und Suffixe folgen.

## Medien

Ein kHome-Telegramm kann auf verschiedenen Wegen übertragen werden. Zwischen den Medien kann ein Router/ Gateway vermitteln. Folgende Medien sind spezifiziert:

### kHome RF

kHome-Telegramme können im 868 MHz-Band über Funk übertragen werden. Dieser Übertragungsweg ist momentan mit den Texas Instruments CC1310-Chips in Evaluierung.

### kHome Serial

kHome-Telegramme können über eine übliche serielle Schnittstelle (UART, RS232) übertragen werden.

Folgender Aufbau wird für ein kHome-Serial-Telegramm verwendet:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 - n | n + 1 | n + 2 |
| Telegrammstart | kHome-Standardtelegramm | Carriage Return (Wagenrücklauf) | Line Feed (Zeilenvorschub) |
| 0xAA (0d170) | siehe 2.2 | \r (0x0D, 0d13) | \n (0x0A, 0d10) |

## Standardtelegramm

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |  | 5+(n–1) | 6+(n–1) |
| Protokolltyp | Telegrammtyp | Adresse des Senders | Adresse des Empfängers | [Länge der Nutzdaten](#_Nutzdaten) | [Nutzdaten  Byte 0](#_Nutzdaten) | … | [Nutzdaten Byte n – 1](#_Nutzdaten) | CRC |

### Protokolltyp

Das Protokolltyp-Byte informiert über die Spezifikation, der die nachfolgenden Daten folgen. Momentan muss an dieser Stelle immer 0x01 übertragen werden.

Es dient für künftige Versionen des Protokolls, welche größere Änderungen im Telegrammaufbau einführen. Ein kHome-Gerät kann somit korrekt (mit einem Fehler) antworten, wenn ein Telegramm einer neueren Protokollspezifikation gesendet wird. Es wird nicht versucht ein Telegramm neueren Aufbaus zu interpretieren.

### Telegrammtyp

Der Telegrammtyp definiert das Kommando, welches ausgeführt werden soll. Neben den hier spezifizierten Telegrammtypen kann jedes kHome-Gerät auch eigene Telegrammtypen spezifizieren.

|  |  |  |
| --- | --- | --- |
| Wertigkeit | Typ | Kurz-Klartext |
| 0x01 | Schreiben in ein Datenregister | REG\_W |
| 0x02 | Lesen eines Datenregisters | REG\_R |
| 0x03 | Broadcast eines Datenregisters | REG\_B |
| 0x04 | Schreiben eines Konfigurationsbytes | CNF\_W |
| 0x05 | Lesen eines Konfigurationsbytes | CNF\_R |
| 0x06 | Lesen eines Statusbytes | STS\_R |
| 0x07 – 0x09 | Reserviert |  |
| 0x10 – 0x5F | Gerätespezifisch (falls nicht spezifiziert: reserviert) |  |
| 0x60 – 0xFE | Reserviert |  |
| 0xFF | Antwort auf ein Telegramm | ANS |

#### REG\_W

Schreiben eines Wertes in ein Register. Die Standard-Register enthalten Daten, die für die alltägliche Ausführung des Gerätes zuständig sind (Solltemperaturen, Schaltstatus, …). Datenregister können 1, 2 oder 4 Byte breit sein, was in der Gerätedokumentation hinterlegt ist.

Ein Gerät kann bis zu 255 Register besitzen.

Je nach Geräteimplementierung können Register als Read-only definiert werden. Dies ist in der Gerätedokumentation hinterlegt.

Je nach Geräteimplementierung kann mit dem Schreiben eines Registers eine Funktion aufgerufen/ getriggert werden (z.B. Starten eines Sensor-Lesevorgangs, Schalten eines Ausgangs). Dies ist in der Gerätedokumentation hinterlegt.

Die Länge der Nutzdaten beträgt für einen REG\_W-Befehl mindestens zwei Byte. Sie werden folgendermaßen definiert:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 4 | 5 | 6 | 7 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | Nutzdaten Byte 1 | Nutzdaten Byte 2 | … |
|  | 0x02 | Adresse des Registers | Wert, der in das Register geschrieben werden soll. (höchstwertiges Byte) | Wert, der in das Register geschrieben werden soll (zweit-höchstwertiges Byte), register-abhängig |  |

Folgende Antworten sind möglich:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Weitere Daten in der Antwort |
| 0x00 | fehlerfrei | * #7: neuer Wert des Registers |
| 0xFB | die Länge der übermittelten Registerdaten stimmt nicht mit der Länge des Registers überein |  |
| 0xFC | der Wert, der in das Register geschrieben werden soll, ist ungültig | - |
| 0xFD | das CRC-Byte stimmte nicht mit dem Berechneten überein | - |
| 0xFE | Das Register ist als Read-only deklariert | - |
| 0xFF | Die Adresse des R ist unbekannt | - |

#### REG\_R

Auslesen eines Register-Wertes. Die Standard-Register enthalten Daten, die für die alltägliche Ausführung des Gerätes zuständig sind (Solltemperaturen, Schaltstatus, …). Datenregister können 1, 2 oder 4 Byte breit sein, was in der Gerätedokumentation hinterlegt ist.

Ein Gerät kann bis zu 255 Register besitzen.

Je nach Geräteimplementierung können Register als Read-only definiert werden. Dies ist in der Gerätedokumentation hinterlegt.

Je nach Geräteimplementierung kann mit dem Lesen eines Registers eine Funktion aufgerufen/ getriggert werden (z.B. Starten eines Sensor-Lesevorgangs, Schalten eines Ausgangs). Dies ist in der Gerätedokumentation hinterlegt.

Die Länge der Nutzdaten beträgt für einen REG\_R-Befehl immer ein Byte. Sie werden folgendermaßen definiert:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4 | 5 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | … |
|  | 0x01 | Adresse des Registers |  |

Folgende Antworten sind möglich:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Weitere Daten in der Antwort |
| 0x00 | fehlerfrei | * #7: Wert des Registers |
| 0xFD | das CRC-Byte stimmte nicht mit dem Berechneten überein | - |
| 0xFF | Die Adresse des Konfigurationsregisters ist unbekannt | - |

#### REG\_B

Globales Senden/ Broadcast eines Register-Wertes. Im Gegensatz zu anderen Befehlen werden mit diesem Befehl keine anderen Geräte angesprochen, sondern ein Register-Wert wird durch ein Gerät selbst auf das Netzwerk gesendet. Je nach Geräteimplementierung kann dies beispielsweise zeitbasiert oder durch Werteänderung erfolgen.

Folgende Besonderheiten beim Telegrammaufbau sind zu beachten:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 | 5 | 6 | 7 |  |
| … | Adresse des Empfängers | Länge der Nutzdaten | Nutzdaten Byte 0 | Nutzdaten Byte 1 | Nutzdaten Byte 2 | … |
|  | 0xFF | 0x02 – 0x05 | Adresse des Registers | Wert des Registers (höchstwertiges Byte) | Wert des Registers (zweit-höchstwertiges Byte), registerspezifisch |  |

Eine Antwort auf dieses Telegramm ist nicht nötig.

#### CNF\_W

Schreiben eines Wertes in ein Konfigurationsregister. Konfigurationsregister enthalten Daten die das Verhalten eines Gerätes verändern (Timeouts, automatisches Senden, (de)aktivieren von Gerätefunktionen). Ein Gerät kann mit bis zu 255 Konfigurationswerten parametrisiert werden.

Mit der kHome-Protokollspezifikation werden folgende Registeradressen normiert:

|  |  |
| --- | --- |
| Konfigurations-registeradresse | Bedeutung |
| 0x00 | Geräteadresse. Standardmäßig auf 0x00 gesetzt. |

Sonstige Registeradressen und deren Bedeutungen sind in der jeweiligen Gerätedokumentation beschrieben.

Die Länge der Nutzdaten beträgt für einen CNF\_W-Befehl immer zwei Byte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4 | 5 | 6 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | Nutzdaten Byte 1 | … |
|  | 0x02 | Adresse des Konfigurations-registers | Wert, der in das Konfigurations-register geschrieben werden soll. |  |

Folgende Antworten sind möglich:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Weitere Daten in der Antwort |
| 0x00 | fehlerfrei | * #7: neuer Wert des Registers |
| 0xFC | der Wert, der in das Register geschrieben werden soll, ist ungültig | - |
| 0xFD | das CRC-Byte stimmte nicht mit dem Berechneten überein | - |
| 0xFE | Das Register ist als Read-only deklariert | - |
| 0xFF | Die Adresse des Konfigurationsregisters ist unbekannt | - |

#### CNF\_R

Lesen eines Wertes aus einem Konfigurationsregister. Konfigurationsregister enthalten Daten die das Verhalten eines Gerätes verändern (Timeouts, automatisches Senden, (de)aktivieren von Gerätefunktionen). Ein Gerät kann mit bis zu 255 Konfigurationswerten parametrisiert werden.

Hinweise zur Belegung der Registeradressen sind unter 2.1.2.3 zu finden.

Die Länge der Nutzdaten beträgt für einen CNF\_W-Befehl beträgt immer ein Byte:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4 | 5 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | … |
|  | 0x02 | Adresse des Konfigurations-registers |  |

Folgende Antworten sind möglich:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Weitere Daten in der Antwort |
| 0x00 | fehlerfrei | * #7: Wert des Registers |
| 0xFD | das CRC-Byte stimmte nicht mit dem Berechneten überein | - |
| 0xFF | Die Adresse des Konfigurationsregisters ist unbekannt | - |

#### STS\_R

Statusbytes enthalten Informationen zum Ablaufstatus der Software der Geräte. Da sie nur Status enthalten und vom Gerät selbst gesetzt werden sind sie Read-only-Register.

Mithilfe der kHome-Protokollspezifikation sind folgende Statusbytes normiert:

|  |  |
| --- | --- |
| Statusbyte-adresse | Bedeutung |
| 0x00 | Globaler Ausführungsstatus. Soll dem Anwender eine grundlegende binäre Information zum Ausführungsstatus des Gerätes geben. (Im Sinne von „alles OK“ oder „Problem“).  Ein Wert von 0x00 steht für eine grundlegen problemlose Funktion.  Die anderen Werte sind in der jeweiligen Gerätedokumentation beschrieben, jedoch sollten alle Werte ungleich 0x00 für einen Fehler stehen. |
| 0x01 | Gerätetyp |

Der Gerätetyp wird folgendermaßen vergeben:

|  |  |
| --- | --- |
| Gerätetyp | Beschreibung |
| 0x01 | temperatureSensor V1 |

Sonstige Statusbytes sind in der jeweiligen Gerätedokumentation beschrieben.

Die Länge der Nutzdaten für einen STS\_R-Befehl beträgt immer genau ein Byte:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4 | 5 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | … |
|  | 0x02 | Adresse des Statusregisters |  |

Folgende Antworten sind möglich:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Weitere Daten in der Antwort |
| 0x00 | fehlerfrei | * #7: Wert des Statusregisters |
| 0xFD | das CRC-Byte stimmte nicht mit dem Berechneten überein | - |
| 0xFF | Die Adresse des Statusregisters ist unbekannt | - |

#### ANS

Eine Antwort auf ein vorheriges Telegramm. Es muss vom ursprünglichen Telegrammsender auf ein Antworttelegramm gewartet werden, bevor ein neues gesendet wird.

Mit dem Antworttelegramm wird ein Status überliefert, der Auskunft über die Ausführung des im Telegramm spezifizierten Befehles gibt.

Die Länge der Nutzdaten beträgt mindestens zwei Byte:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 4 | 5 | 6 | 7 |  |
| … | Länge der Nutzdaten | Nutzdaten Byte 0 | Nutzdaten Byte 1 | Nutzdaten Byte 2 | … |
|  | mindestens 0x01 | Antwortcode | Typ des Telegramms, auf das geantwortet wird | spezifisch |  |

Folgende Antwortcodes sind spezifiziert:

|  |  |  |
| --- | --- | --- |
| Antwortcode | Beschreibung | Länge der Nutzdaten |
| 0x00 | OK. Befehl wurde problemlos ausgeführt | mindestens 2 |
| 0x01 – 0x09 | Reserviert |  |
| 0x10 – 0x5F | Gerätespezifisch |  |
| 0x60 - | Reserviert |  |
| 0xFB | Fehler: Die Länge der übermittelten Registerdaten stimmt nicht mit der Länge des Registers überein | 2 |
| 0xFC | Fehler: Der Wert, der in das Register geschrieben wurde, ist ungültig. (siehe Gerätedokumentation) | 2 |
| 0xFD | Fehler: Das CRC-Byte stimmte nicht mit dem berechneten überein Telegrammbyte #6 (Typ des Telegramms, auf das geantwortet wird, ist in diesem Fall ebenfalls 0xFD) | 2 |
| 0xFE | Fehler: Das Register ist als Read-only deklariert, es darf nur gelesen werden | 2 |
| 0xFF | Fehler: Die Adresse des (Konfigurations-, Status-) Registers ist unbekannt | 2 |

### Adresse des Senders

Der Sender kann Adressen von 1 bis 254 besitzen (inklusive 1 und 254) besitzen. Somit sind mit der aktuellen Protokollrevision bis zu 254 kHome-Geräte in einem Netzwerk möglich.

### Adresse des Empfängers

Zusätzlich zu den bei 1.1.3 beschriebenen Adressen kann die Empfängeradresse 255 betragen. Dabei handelt es sich um ein Broadcast-Paket, siehe 2.1.2.3

### Nutzdaten

Im Telegramm folgen die Länge der Nutzdaten und die Nutzdaten selbst. Es können 0 bis 255 Byte Nutzdaten übertragen werden. Jeweilige Anforderungen an die Nutzdaten und deren Bedeutung ist vom Telegrammtyp abhängig.

Zuerst wird das nullte Byte übertragen, als letzte das n-te Byte (wobei n = Länge der Nutzdaten – 1)

### CRC

Zur Verifizierung der korrekten Übertragung wird eine CRC-Wert über die Daten, vom Protokolltypbyte bis zum letzten Nutzdatenbyte.

Verwendet wird eine CRC-8-Wert mit dem Grundpolynom 0x07.

Informationen zur Berechnung, eine C#-Beispielimplementierung sowie ein Online-CRC-Rechner sind auf folgenden Webseiten zu finden:

<http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html#ch44>

http://www.sunshine2k.de/coding/javascript/crc/crc\_js.html

# Device Files

Device Files definieren alle Register (und weitere Daten), die für ein kHome-Gerät relevant sind.

Device Files besitzen die Dateiendung „\*.khd“.

## XML-Syntax vom Device Files

Device-Files enthalten eine XML-Syntax, die hier exemplarisch dargestellt ist:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <khd> | | | | |
|  | <version> | | | |
|  | |  | Device File Version. Momentan immer „1.0“ |  |
|  | </version> | | | |
|  | <meta> | | | |
|  | | <author> | Autor der Device File | </author> |
|  | | <comment> | Kommentar zum Gerät | </comment> |
|  | | <deviceVersion> | Version des kHome-Gerätes (Software, Hardware) | </deviceVersion> |
|  | | <deviceID> | Gerätetyp (dezimal), Statusregister Adresse 0x01 (siehe auch 2.1.2.6) | </deviceID> |
|  | </meta> | | | |
|  | <dataRegister> | | | |
|  | | <address> | Adresse des Registers als hexadezimaler Text. Falls nicht angegeben: 0. | </address> |
|  | | <lengthByte> | Länge des Registers in Byte. Kann 1, 2 oder 4 sein. Falls nicht angegeben: 1 | </lengthByte> |
|  | | <readOnly> | Markiert das Register als Read-only, wenn „true“. Sonst „false“. Falls nicht angegeben: false. | </readOnly> |
|  | | <initialValue> | Initialer Wert (nach dem Einschalten/ Resetten) als vorzeichenbehaftete Dezimalzahl.  Falls nicht angegeben: 0 | </initialValue> |
|  | | <name> | Name des Registers im Klartext. Keine Leerzeichen, Sonderzeichen, Zeilenumbrüche | </name> |
|  | | <description> | Beschreibung des Registers. Kann Sonderzeichen und Leerzeichen enthalten. Zeilenumbrüche werden durch ein „<br/>“ dargestellt. | </description> |
|  | </dataRegister> | | | |
|  | <dataRegister> | | | |
|  | |  | weitere(s) Datenregister (optional) |  |
|  | </dataRegister> | | | |
|  | <configRegister> | | | |
|  | | <lengthByte> | immer „1“ (muss nicht angegeben werden) | </lengthByte> |
|  | | <…> | Tags wie beim dataRegister (außer „lengthByte“) |  |
|  | </configRegister> | | | |
|  | <configRegister> | | | |
|  | |  | weitere(s) Konfigurationsregister (optional) |  |
|  | </configRegister> | | | |
|  | <statusRegister> | | | |
|  | | <lengthByte> | immer „1“ (muss nicht angegeben werden) | </lengthByte> |
|  | | <readOnly> | immer „true“ (muss nicht angegeben werden) | </readOnly> |
|  | | <…> | Tags wie beim dataRegister (außer „lengthByte“ und „readOnly“ |  |
|  | </statusRegister> | | | |
|  | <statusRegister> | | | |
|  | |  | weitere(s) Statusregister (optional) |  |
|  | </statusRegister> | | | |
| </khd> | | | | |

## Beispiel für eine Device File

Ein Beispiel für eine Device-File kann folgendermaßen aussehen (exemplarisch, nicht semantisch sinnvoll):

<?xml version=**"1.0"** encoding=**"UTF-8"** standalone=**"no"**?>

<khd>

<dataRegister>

<address>**49**</address>

<lengthByte>**2**</lengthByte>

<readOnly>**true**</readOnly>

<initialValue>**15**</initialValue>

<name>**dataRegOne**</name>

<description>**Read only,**&lt;**br/**&gt;**initial 15,**&lt;**br/**&gt;**dataRegOne,**&lt;**br/**&gt;**Length 2,**&lt;**br/**&gt;**Address 1**</description>

</dataRegister>

<dataRegister>

<address>**2**</address>

<lengthByte>**4**</lengthByte>

<readOnly>**false**</readOnly>

<initialValue>**2**</initialValue>

<name>**dataRegTwo**</name>

<description>**initial 2,**&lt;**br/**&gt;**dataRegTwo,**&lt;**br/**&gt;**Length 4,**&lt;**br/**&gt;**Address 2**</description>

</dataRegister>

<configRegister>

<address>**5**</address>

<lengthByte>**1**</lengthByte>

<readOnly>**false**</readOnly>

<initialValue>**2**</initialValue>

<name>**configOne**</name>

<description>**configOne,**&lt;**br/**&gt;**address 5,**&lt;**br/**&gt;**value 2**</description>

</configRegister>

<configRegister>

<address>**6**</address>

<lengthByte>**1**</lengthByte>

<readOnly>**true**</readOnly>

<initialValue>**7**</initialValue>

<name>**configTwo**</name>

<description>**configTwo:**&lt;**br/**&gt;**address 6**&lt;**br/**&gt;**value 7**&lt;**br/**&gt;**read only**</description>

</configRegister>

<statusRegister>

<address>**8**</address>

<lengthByte>**1**</lengthByte>

<readOnly>**true**</readOnly>

<initialValue>**3**</initialValue>

<name>**statusOne**</name>

<description>**statusOne:**&lt;**br/**&gt;**Value 3**&lt;**br/**&gt;**Address 8**</description>

</statusRegister>

<statusRegister>

<address>**9**</address>

<lengthByte>**1**</lengthByte>

<readOnly>**true**</readOnly>

<initialValue>**200**</initialValue>

<name>**statusTwo**</name>

<description>**statusTwo:**&lt;**br/**&gt;**address 9**&lt;**br/**&gt;**value 200**</description>

</statusRegister>

<meta>

<author>**Peter Kappelt**</author>

<comment>**This is an empty device. Just for testing!**&lt;**br/**&gt;**An a second line of description.**</comment>

<deviceVersion>**V1.0**</deviceVersion>

</meta>

<version>**1.0**</version>

</khd>

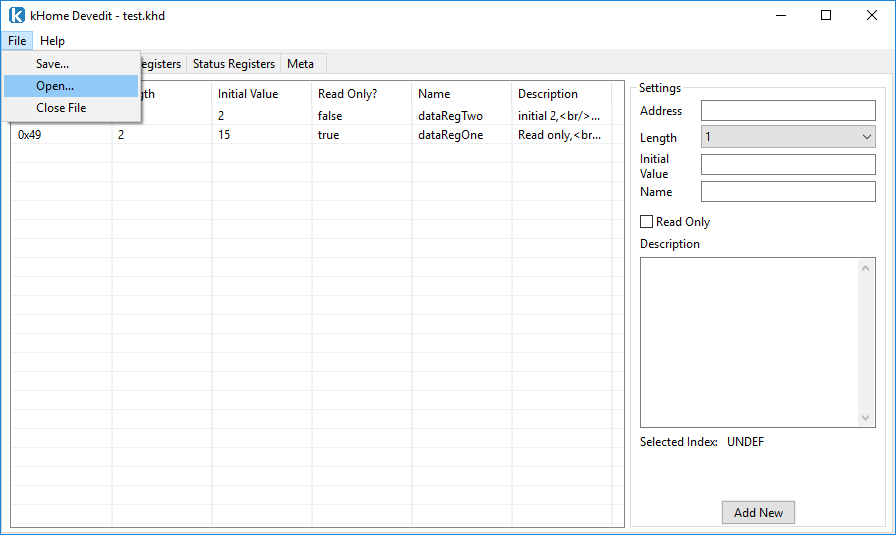
## Software

Zur Bearbeitung und Weiterverwendung der kHome-Device-Files stehen die Java-basierten kHome Devedit-Tools bereit.

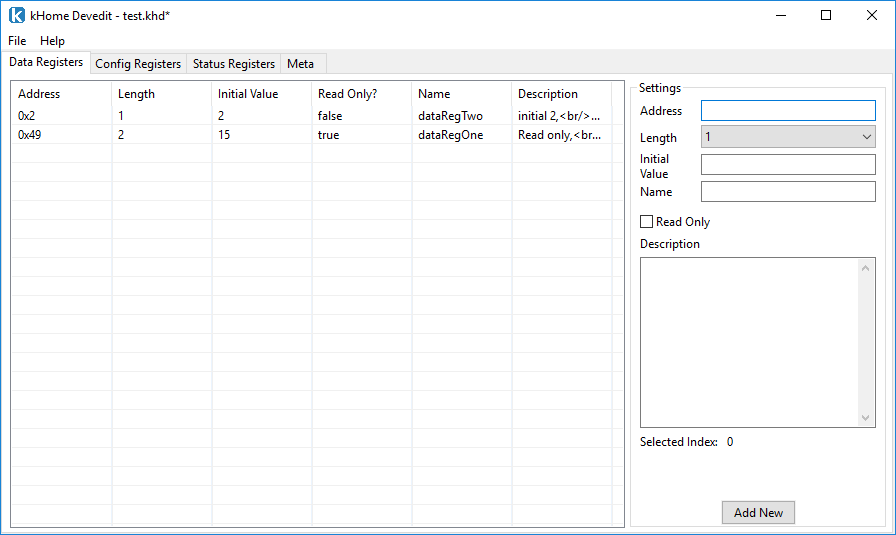
Zum Starten des Programmes muss auf dem Computer eine Java Runtime Environment, in der Version 1.8, installiert sein.

### Erstellen und Bearbeiten von Device-Files

Eine kHome-Device-File kann über das Menü „File“ geöffnet und gespeichert werden. Nachdem eine Device-File geöffnet wurde, steht dessen Dateiname in der Titelzeite.



Wenn eine Änderung innerhalb des Programms vorgenommen wurde, diese aber noch nicht gespeichert wurde, wird die Titelzeile mit einem Stern („\*“) ergänzt:

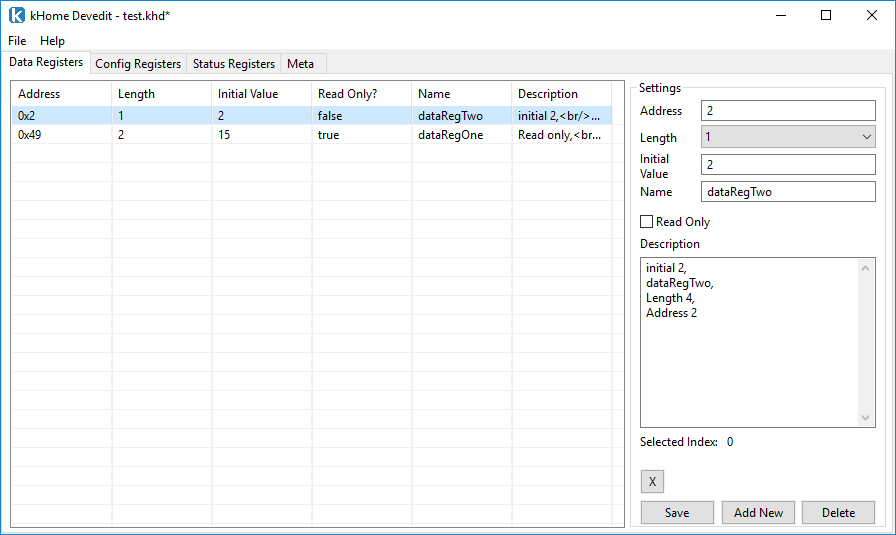


Das Programm ist in mehrere Tabs unterteilt.

Im Meta-Tab können die Meta-Informationen der Device-File eingesehen und bearbeitet werden. Nachdem Meta-Informationen bearbeitet wurden, ist der Klick auf „Save Meta“ nicht zu vergessen.

Der Aufbau der Tabs „Data Registers“, „Config Registers“ und „Status Registers“ ist grundlegend identisch. In ihnen werden die jeweiligen Register hinzugefügt, bearbeitet oder entfernt.

Zum Bearbeiten oder Löschen eines bereits definierten Registers muss dieses in der Tabelle ausgewählt werden:



In der „Settings“-Leiste werden die entsprechenden Werte übernommen, weiterhin erscheinen die Buttons „X“, „Save“ und „Delete“.

Mit dem Button „X“ kann die aktuelle Zeile deselektiert werden. Dies ist zum Erstellen eines neuen Registers nötig, da dort kein bestehendes Register angewählt sein darf.

Mit dem Button „Delete“ wird das gewählte Register gelöscht.

Änderungen in den Registereinstellungen können in den Eingabefeldern vorgenommen werden. Ein Klick auf „Save“ übernimmt diese Änderungen.

Um ein neues Register zu definieren, muss sichergestellt werden, dass keines angewählt ist. Falls der Button mit der Aufschrift „X“ sichtbar ist, muss dieser gedrückt werden um die aktuelle Auswahl aufzuheben.

Nun können in der „Settings“-Leiste die Parameter für das neue Register eingegeben werden. Durch einen Klick auf „Add New“ wird dieses übernommen.

### Report-/ Code-Generatoren

Die Devedit-Tools stellen Funktionen bereit, die die Register und Metadaten in externe Dateien einbetten.

Sie basiert auf einer Dateivorlage, in dem verschiedene Tags durch die entsprechenden Daten ersetzt werden. Die entsprechende Funktion ist vorbereitet und muss nur mit einer Vorlage aufgerufen werden.

Beispielhaft wurden folgende Funktionen für die Generatoren vorbereitet:

* Ein HTML-Bericht der Informationen und Register eines Gerätes
* Ein C-API, das als Registerverwaltung, besonders für kHome-Software auf Embedded-Geräten dient. Verschiedene Funktionen zum Lesen/ Schreiben der Register sind bereitgestellt und in der Datei selbst kommentiert. Die Kommentare sind Doxygen-kompatibel, womit direkt aus dem generierten Quelltextdateien eine HTML-Dokumentation der Funktionen erstellt werden kann.

Folgende Tags sind definiert. Ein Tag in einer Vorlage wird durch die passenden Daten ersetzt.

|  |  |
| --- | --- |
| Tag | Beschreibung |
| {$GEN\_TIME} | Zeitstempel zum Zeitpunkt der Generierung, Format „yyyy-mm-dd hh:mm:ss“ |
| {$META\_AUTHOR} | Meta-Information: Autor |
| {$META\_COMMENT} | Meta-Information: Kommentar |
| {$META\_DEVICE\_ID\_DEC} | Meta-Information: Gerätetyp, in dezimaler Darstellung |
| {$META\_DEVICE\_ID\_HEX} | Meta-Information: Gerätetyp, in hexadezimaler Darstellung |
| {$META\_DEVICE\_VERSION} | Meta-Information: Geräteversion (Hardware, Software) |
| {$FILE\_NAME} | Dateiname der geöffneten Device-File |
| {$BLOCK\_DATAREGISTER\_START}  …  {$BLOCK\_DATAREGISTER\_STOP} | Der Inhalt des Blockes wird so oft kopiert, wie Datenregister existieren. Innerhalb des Blockes werden die Tags für das jeweilige Register ersetzt |
| {$BLOCK\_CONFIGREGISTER\_START}  …  {$BLOCK\_CONFIGEGISTER\_STOP} | Der Inhalt des Blockes wird so oft kopiert, wie Konfigurationsregister existieren. Innerhalb des Blockes werden die Tags für das jeweilige Register ersetzt |
| {$BLOCK\_STATUSREGISTER\_START}  …  {$BLOCK\_STATUSREGISTER\_STOP} | Der Inhalt des Blockes wird so oft kopiert, wie Statusregister existieren. Innerhalb des Blockes werden die Tags für das jeweilige Register ersetzt |
| Folgende Element können nur innerhalb eines Blockes vorkommen: |  |
| {$ADDRESS\_DEC} | Adresse des Registers in dezimaler Darstellung |
| {$ADDRESS\_HEX} | Adresse des Registers in hexadezimaler Darstellung |
| {$LENGTH\_BYTE} | Länge des Registers in Byte (1, 2 oder 4) |
| {$INITIAL\_VALUE} | Wert nach dem Einschalten/ Resetten |
| {$READ\_ONLY} | True, wenn das Register als Read-only markiert ist. Sonst false |
| {$NAME} | Name des Registers |
| {$DESCRIPTION} | Beschreibung des Registers. Ein Zeilenumbruch wird durch den HTML-Tag „<br/>“ dargestellt. |

# Versionsgeschichte

|  |  |  |
| --- | --- | --- |
| Datum | Version | Änderungen |
| 28. Februar 2017 | V0.1 | * Initiale Version |
| 11. März 2017 | V0.2 | * Device Files & Software * Datenregister: Hinweis auf verschiedene Längen |
| 25. März 2017 | V0.3 | * Devedit: neuer Generatortag für Zeitstempel * Devedit: Generierung von C-Register-APIs |