kHome Protokollspezifikation

Version 0.4

Peter Kappelt

letzte Änderung: 08.04.2017 20:49

1 Inhalt

2 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.

2.1 Medien

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

2.1.1 kHome RF

kHome-Telegramme können im 868 MHz-Band über Funk, mit den Texas Instruments CC1310-Chips, übertragen werden. Ein kHome-Telegramm wird dabei in das CC1310-Datenpaket aus dem Proprietary-Mode eingebunden.

Diese Einstellungen wurden verwendet:

- 868,0 MHz Frequenz, 50 kbps Symbol Rate, 25 kHz Deviation
- 4 Byte Präambel
- 32 Bit Sync-Word, 0x930b51dde
- keine Sequenznummer, keine paketseitige Adresse

2.1.2 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)

2.2 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 Byte 0	...	Nutzdaten Byte n – 1	CRC

2.2.1 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.

2.2.2 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

2.2.2.1 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	...
	mindestens 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	• ab #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 Registers ist unbekannt	-

Beispiel:

```
01 01 FF 00 02 02 05 28
(Schreibe den Wert 5 in das Datenregister 0x02 des Gerätes mit der Adresse 0x00.
Wir senden von einem Gateway (-> Adresse 0xFF))

Antwort:
01 FF 00 FF 06 00 01 00 00 00 05 5C
(Fehlerfrei, neuer Wert 5 wurde übernommen)
```

2.2.2.2 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	• ab #7: Wert des Registers
0xFD	das CRC-Byte stimmte nicht mit dem Berechneten überein	-
0xFF	Die Adresse des Konfigurationsregisters ist unbekannt	-

Beispiel:

```
01 02 FF 00 01 02 27
(Lesen des Wertes des Register 0x02 von dem Gerät mit der Adresse 0x00)

Antwort:
01 FF 00 FF 06 00 02 00 00 00 05 FA
(Fehlerfrei, Wert des Registers ist 05, außerdem ist das Register 4 Bytes groß
(da 4 Datenbytes gesendet wurden))
```

2.2.2.3 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.

Beispiel:

```
01 03 06 FF 02 03 15 BC
(Das Gerät 0x06 sendet sein Register mit der Adresse 03. Der Wert ist 21 (0x15))
```

2.2.2.4 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	-

Beispiel:

```
01 04 FF 00 02 05 35 5E
(Schreibe von einem Gateway (Senderadresse 0xFF) den Wert 53 (0x35) in das
Konfigurationsregister 0x05 des Gerätes 0x00)

Antwort:
01 FF 00 FF 03 00 04 35 8B
(fehlerfrei, der neue Wert „53“ des Registers wurde übernommen)
```

2.2.2.5 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	...
	0x01	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	-

Beispiel:

```
01 05 FF 00 01 05 1B
(Lese von einem Gateway (Senderadresse 0xFF) aus das Konfigurationsregister 0x05
des Geräts 0x00)

Antwort:
01 FF 00 FF 03 00 05 35 9E
(fehlerfrei, das Konfigurationsregister hält den Wert 0x35)
```

2.2.2.6 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	...
	0x01	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	-

Beispiel:

```
01 06 FF 00 01 08 9E
(Lese von einem Gateway (Senderadresse 0xFF) aus das Statusregister 0x08 des
Geräts 0x00)

Antwort:
01 FF 00 FF 03 00 06 03 23
(fehlerfrei, das Statusregister hält den Wert 0x03)
```

2.2.2.7 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 0x02	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 Empfängeradresse ist 0xFF	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

2.2.3 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. Prinzipiell sind auch Geräte mit der Adresse 0 möglich, allerdings ist das die Standardadresse unkonfigurierter Geräte. Sie ist daher nicht für die produktive Benutzung freigegeben.

Die Senderadresse 255 ist für Gateways reserviert, z.B. für das kHome RF-zu-Serial Gateway. Mit dieser Adresse werden Schreib- oder Lesebefehle von diesem ausgeführt.

2.2.4 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

2.2.5 Nutzdaten

Im Telegramm folgen die Länge der Nutzdaten und die Nutzdaten selbst. Es können 0 bis 200 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)

2.2.6 CRC

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

Verwendet wird eine CRC-8-Wert mit dem Grundpolynom 0x07, der initiale Wert der CRC-Generierung ist 0x0.

Bei der Entwicklung wurde ein Online-Tool verwendet, dass die CRC-8 Summe aus gegebenen Werten berechnet:
http://www.sunshine2k.de/coding/javascript/crc/crc_js.html.

In der aktuellen C-Software wurde folgende Routine zur Berechnung des CRC-Bytes über ein Byte-Array genutzt:

```
/**
 * @brief calculate the CRC-8 of a uint8_t array
 * @param[in] byteArray     pointer to a array
 * @param[in] length        length of the array
 *
 * @return CRC-8 of the value
 */
uint8_t khCalculateCRC8OfByteArray(uint8_t* byteArray, uint8_t length){
    //Implementation is from Texas Instruments DN502, but modified for CRC-8
    const uint8_t polynomial = 0x07;        //CRC polynomial
    uint8_t checksum = 0;                   //initial value

    uint8_t i, j, currentCRCByte;

    for(j = 0; j < length; j++){
        currentCRCByte = byteArray[j];
        for(i = 0; i < 8; i++){
            if((checksum & 0x80) ^ (currentCRCByte & 0x80)){
                checksum = (checksum << 1) ^ polynomial;
            }else{
                checksum = (checksum << 1);
            }
            currentCRCByte <<= 1;
        }
    }

    return checksum;
}
```

3 Device Files

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

Device Files besitzen die Dateiendung „*.khd“.

3.1 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 „ “ 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>			

3.2 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>
```

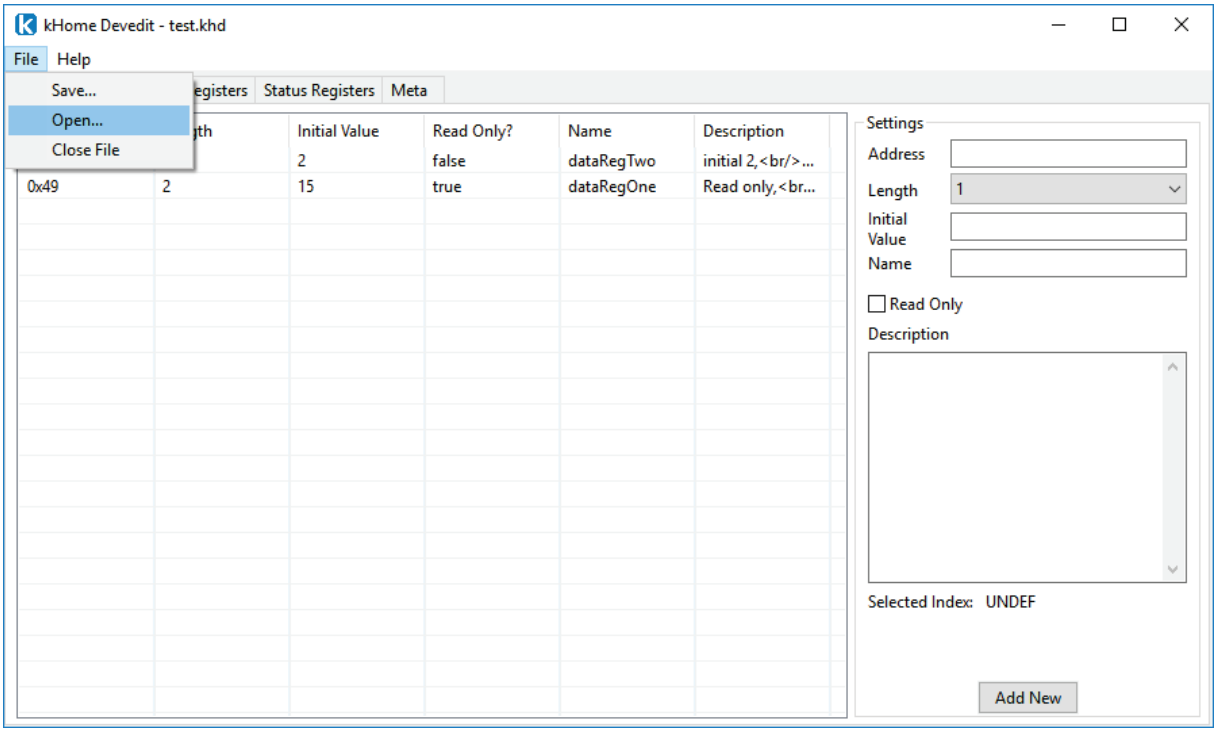
3.3 Software

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

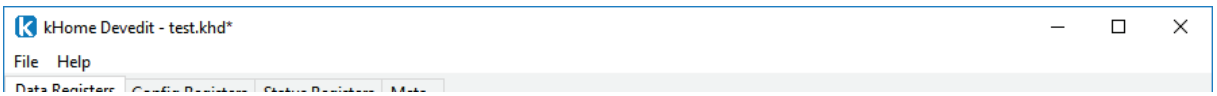
Zum Starten des Programmes muss auf dem Computer eine Java Runtime Environment, in der Version 1.8, installiert sein.

3.3.1 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 Titelzeile.



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

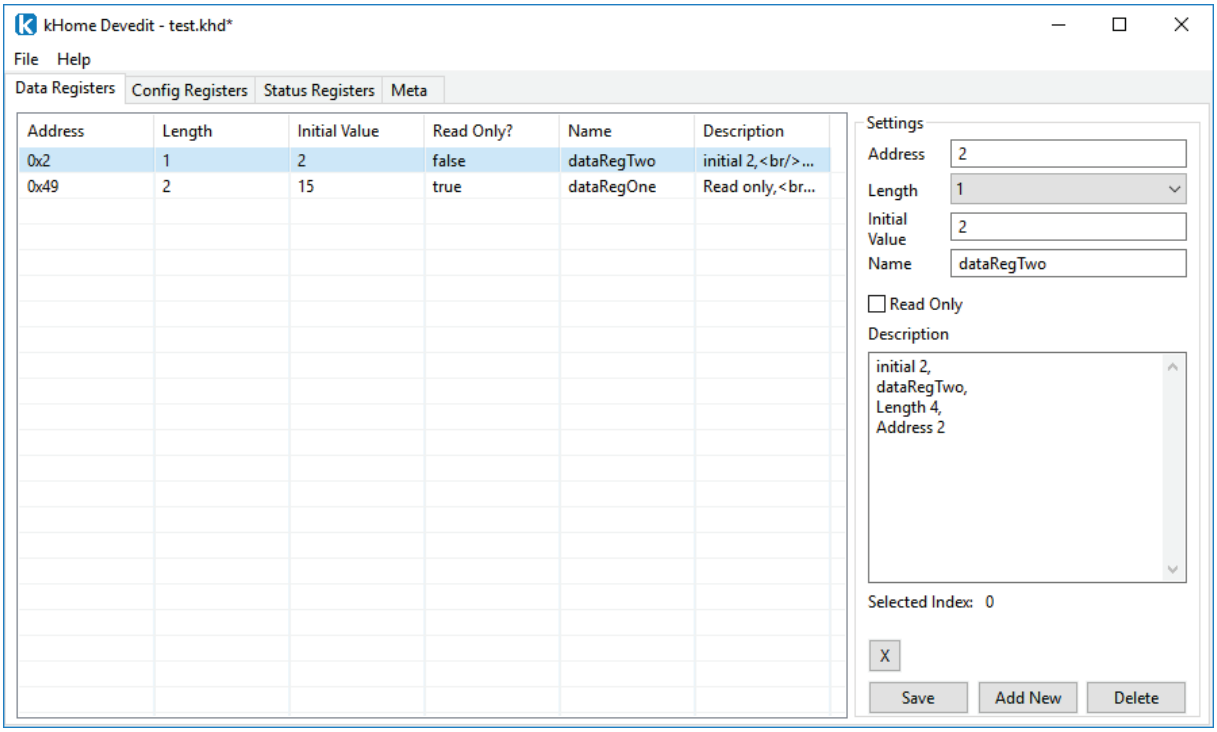


Das Programm ist in mehrere Tabs unterteilt.

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.

3.3.2 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 „ “ dargestellt.

4 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
2. April 2017	V0.31	• Funktion zur CRC-Berechnung • maximale Länge der Nutzdaten auf 200 Byte beschränkt (ursprünglich 255) • kHome RF: kurze Beschreibung
2. April 2017	V0.32	• Behebung kleiner inhaltlicher Fehler
8. April 2017	V0.4	• Beispiele zu den Telegrammtypen