Homematic Wired RS485 Protokollbeschreibung

Inhalt

1. Grundsätzliches

Dieses Dokument ist ein Versuch das Homematic Wired Protokoll zu beschreiben. Das Protokoll bassiert grundsätzlich auf dem HS485 Protokoll von ELV. Diese Protokoll ist bereits gut Dokumentiert und wurde zur Entwicklung eigener Anwendungen offen gelegt.
Viele grundsätzliche Beschreibungen basieren hier auf dieser Dokumentation und wurden in dieses Dokument Übernommen und wo nötig angepasst.

Diese Dokumentation hier ist noch nicht vollständig und in einigen Punkten noch nicht ausreichend verifiziert. Die entsprechendenen Punkte sind mit gelb markiert um das hervorzuheben.

1.1. Datenübertragung

Die Datenübertragung erfolgt seriell über einen RS485 Bus mit folgenden Einstellungen

- 19200 Baud
- 8 Datenbit
- 1 Stoppbit
- Parität gerade

Der RS485 Bus übertragt die Signale mit einem Pegel von +5V. Für eine sichere Übertragung muss der Bus mit einem "Abschlusswiderstand" versehen werden. Dabei ist es unerheblich ob der " Abschlusswiderstand " sich am Ende des Busses befindet. Dieser kann sich auch innerhalb des Busses befinden und dient bei der relativ niedrigen Übertragungsgeschwindigkeit "nur" dazu den Bus auf einem definiertem Pegel zu halten.

Die Topologie des "Busses" ist bei Homematic-Wired unkritisch. Es funktioniert sowohl die Herkömmliche Busverkabelung. Auch eine sternförmige Verkabelung ist unkritisch.

Jede gesendete Nachricht (mit wenigen Ausnahmen) wird vom Empfänger quittiert. Falls keine Bestätigung erfolgt wird die Nachricht bis zu zwei mal wiederholt. Pro Nachrichten können 64 Byte Nutzdaten übertragen werden.

Ausnahmen:

- Nachrichten an die Broadcastadresse werden nicht bestätigt.

1.1.1. Vermeidung von Kollisionen auf dem RS485 Bus

Das Protokoll auf dem Bus muss Multimaster-Eigenschaften unterstützen. Es existiert mit der CCU zwar eine Zentrale die auch die Kommunikation steuert, einzelne Module müssen aber auch ohne Aufforderung durch diese Senden können. Z.B. bei anliegenden Events (Tastendrücke, geänderte Sensordaten usw.)

Die Informationen zur Kollisionsvermeidung sind derzeit Vermutungen von mir. Ggf. bedarf es noch einer weiteren Überprüfung.

- Die Module "überwachen" permanent den Bus. Wenn Module ohne Aufforderung der Zentrale senden wollen, so wird vor dem Senden geprüft ob der Bus "frei" ist. Der Bus ist frei wenn:
  - "aktuell" (innerhalb der letzten xxx ms?) keine Datenüberragung auf dem Bus stattgefunden hat.
  - nach einer zufälligen Wartezeit von xxx ms keine neue Datenübertragung erkannt wird.

  Jedes ankommende Byte löst im Modul (UART) ein Interrupt aus. Somit kann auf neu ankommende Daten ohne große Verzögerung reagiert werden.

- Bestätigungen werden ohne Wartezeit sofort versendet.

Mit den oben genannten Vorkehrungen sind Kollisionen dennoch nicht vollständig auszuschließen. Dadurch dass Nachrichten in der Regel bestätigt werden müssen und beim Ausbleiben dieser die Übertragung wiederholt wird scheint mir die Übertragungssicherheit hier dennoch sehr hoch. Alles weiter muss experimentell beim Bau von eigenen Modulen ermittelt werden. Ggf. im vergleich mit kommerziellen Modulen.

1.1.2. Addressen und Seriennummern

Jedes Gerät am Bus (mit Ausnahme vom Netzteil, Busabschluss und Filter) hat seine eigene weltweit einzigartige 32 Bit Adresse und eine 8 Byte lange Seriennummer. Die Zentrale (CCU) hat die Adresse 0x00000001. Die Adresse 0xFFFFFFFF ist die so genannte Broadcast-Adresse. Nachrichten an die Broadcast-Adresse werden von allen Modulen verarbeitet. Es existiert noch die Adresse 0x00000000. Die Funktion ist mir aber noch nicht klar.

Die Seriennummer besteht aus einer Zeichenkette mit 8 Alphanumerischen Zeichen.

Adressaufbau (Hexadezimale Schreibweise)

00 00 00 01	Zentrale (CCU1)
00 38 01 23	Eine Beispieladresse für ein Modul
FF FF FF FF	Broadcast-Adresse Wird von allen Modulen verarbeitet

Beispielseriennummer: heq25167

Diese Form der Seriennummer wird in den offiziellen Modulen verwendet. Grundsätzlich werden aber an allen Stellen der Seriennummer Alphanumerische Zeichen benutzt.

Erlaubte Zeichen: 0-1, a-z, A-Z

1.2. Protokollrahmen

Jede Nachricht enthält ein Startzeichen, die Zieladresse, ein Kontrollzeichen die Absenderadresse, die Nutzdaten und deren Länge und am Schluss eine Checksumme.
Die Nuztdaten dürfen maximal 64 Byte lang sein. Daraus ergibt sich eine Nachrichtenlänge von 77 Bytes. Falls das Excape-Zeichen (siehe unten) benutzt werden muss, ergibt sich sogar eine maximale Nachrichtlenlänge von 153 Bytes.

Beispiel Protokollrahmen
In diesem Beispiel wird ein Key-Event (siehe unten) von der Adresse 0x3ED zu 0x2AC gesendet.

```
                                        Framelänge
                                        |     Nutzdaten
                                        |__   |___________
FD    00 00 03 ED   1C   00 00 02 DE    06  4B 01 00 0C    F9 8E
|     |             |    |               |   |  |  |  Event |
|     Zieladresse   |    Absenderadresse |   |  |  Aktor    Checksumme
Startzeichen        Kontrollzeichen      |   |  Taste
                                         Befehl (Key)
```

1.2.1. Start- und Steuerzeichen

Das Startzeichen ist ein ein Byte langes Steuerzeichen und markiert den Beginn einer neuen Nachricht. Keines der unten beschriebenen Steuerzeichen darf in der restlichen Nachricht noch einmal vorkommen. Falls notwendig muss das entsprechende Byte dann "Escaped" werden.

Es gibt drei verschiedene Steuerzeichen:

- 0xFD: Startzeichen für alle "normalen" (langen) Nachrichten (mit Adresse)
- 0xFE: Startzeichen für "spezielle" (kurze) Nachrichten (ohne Adresse). genauere Beschreibung folgt noch.
- 0xFC: Escape-Zeichen (siehe nächster Abschnitt)

Für "normale" Module sollte es Ausreichen wenn diese nur auf 0xFD als Startzeichen reagieren.

1.2.2. das Escape-Zeichen

Das Escape-Zeichen (0xFC) ist ein ein Byte langes Steuerzeichen welches immer dann zum Einsatz kommt, wenn die Bytes 0xFD, 0xFE oder 0xFC innerhalb einer Nachricht übertragen werden soll.

im folgenden Beispiel soll die Bytefolge 0x65 0xFD 0x5E übertragen werden.

Das erste Byte 0x65 wird "normal" übertragen. Das zweite Byte entspricht einem Steuerzeichen und muss Escaped werden. Das dritte Byte wird wieder "normal" übertragen.

Beim "Escapen" wird dem betroffenem Byte das Zeichen 0xFC vorangestellt und dem Byte selber das höchstwertige Bit gelöscht. Für die oben genannte Beispiel Bytefolge sieht das Ergebnis dann also so aus:

0x65 0xFC 0x7D 0x5E
| |
| Enspricht 0xFD nachdem das höchstwertige Bit belöscht wurde.
Escape Zeichnen

1.2.3. Zieladresse

Die Übertragung der 4 Byte langen Zieladresse erfolgt im Big-Endian-Format. Dabei wird das höchstwertige Byte als erstes Übertragen. Das niederwertigste Byte als letztes.

1.2.4. Kontrollzeichen

Im Kontrollzeichen ist der Typ der Nachricht kodiert. Zusätzlich werden im Kontrollzeichen noch Nachrichtenspezifische Bits übertragen.

Die drei unterschiedliche Nachrichtentypen werden durch die farbig markierten Bits 0 - 2 (siehe nebenstehende Tabelle) unterschieden.

Adressaufbau des Kontrollzeichens

Type \ Bit	7	6	5	4	3	2	1	0
I-Nachricht	Y	R	R	F	B	S	S	0
ACK-Nachricht	0	R	R	F	B	0	0	1
Discovery	M	M	M	M	M	0	1	1

Die Bits haben folgende Bedeutung:

- **S - Sendefolgenummer:**
  Die Sendefolgenummer kann die Zahlen 0 bis 3 darstellen und besteht aus zwei Bits. Bei jeder erfolgreich versendeten Nachricht wird die Sendefolgenummer um eins erhöht. Nach der Nummer 3 folgt dann wieder die 0. Muss die Nachricht wegen eines Fehlers wiederholt werden, so wird die Sendefolgenummer noch einmal verwendet. Kommt beim Empfänger zwei Nachrichten mit der gleichen Sendefolgeniummer an, so werden beide bestätigt, aber nur eine verarbeitet.

- **B - Absenderaddresse:**
  Enthält das Kontrollbyte das B-Bit, so wird nach dem Kontrollbyte die 4 Byte Absenderadresse erwartet. Die Absenderadresse ist immer erforderlich, da zur Bestätigung von Nachrichten die Absenderadresse bekannt sein muss. Es gibt jedoch Ausnahmen in denen die Absenderadresse nicht notwendig ist. ??? Was sind die Ausnahmen?

- **F - letztes Paket:**
  Ist ein Datensatz zu groß für eine Nachricht (> 64 Byte), so wird die Nachricht aufgeteilt. Die letzte Nachricht des Datensatzes wird dabei mit einem gesetzten F -Bit gesendet. Nachrichten < 64 Byte haben daher immer das F-Bit gesetzt. Bei allen von Homematic Wired bisher beobachteten Nachrichten war das F-Bit bisher immer gesetzt. Kann sein, das das Bei Homematic nicht verwendet wird.

- **R - Empfangsfolgenummer:**
  Die Empfangsfolgenummer wird zur Bestätigung von Nachrichten verwendet. Erhält ein Modul eine Nachricht, so wird diese bestätigt. In der Bestätigungsnachricht entspricht die Empfangsfolgenummer der Sendefolgenummer der erhaltenen Nachricht. Der Sender erkennt daran, dass die Nachricht erfolgreich an den Empfänger übertragen wurde.

- **Y - Synchronisationsbit:**
  Wird beim Senden das Synchronisationsbit gesetzt, so wird im Empfänger die Sendefolgenummer zurückgesetzt und die Empfangsfolgenummer wird auf den Wert der Sendefolgenummer gesetzt und bestätigt. Dabei ist zu beachten, dass jede Nachricht mit gesetztem Y-Bit verarbeitet werden muss. Also auch wiederholte Nachrichten.

- **M - Adressmaske:**
  Wird eine Discovery-Nachricht versendet, so entsprechen die 5 M-Bit der Adressmaske. Sie gibt an, wie viele Bits (M+1) der Empfängeradresse gewertet werden sollen.

Nachrichtentypen

- **I-Nachricht:**
  Soll ein Datenaustausch zwischen den Modulen erfolgen, so wird eine I-Nachricht verwendet. Enthält die gesendete Nachricht eine Abfrage an das Modul, so wird mit einer I-Nachricht geantwortet. Diese Antwort enthält bereits die Bestätigung der vorherigen Nachricht.

- **ACK-Nachricht:**
  Erhält ein Modul eine Nachricht, auf die es keine Antwort senden muss so bestätigt es diese Nachricht mit einer ACK-Nachricht. Die ACK Nachricht muss innerhalb von 100ms gesendet sein. ??? (laut HS485 Quellcode trifft das auch auf HM zu?)
  Grundsätzlich werden ALLE Befehle mindestens mit einer ACK-Nachricht beantwortet. Auch Unbekannte.

- **Discovery-Nachricht:**
  Für einen Scan am Bus, um neue Module zu suchen, sendet die Zentrale Discovery-Nachrichten. Alle Module die noch nicht mit der Zentrale gekoppelt sind vergleichen mit Hilfe der Adressmaske die Zieladresse mit ihrer eigenen Adresse. Die Adressmaske bestimmt, wie viele Bits gültig sind, beginnend beim höchstwertigen Bit. Bei Übereinstimmung sendet das Modul ein 0xF8. Die Zentrale erkennt dies und stellt dadurch fest, dass sich noch weitere Module mit dieser Adressmaske am Bus befinden, und pass die Adressmaske und die Zieladresse an. Dies wird so lange durchgeführt, bis alle Module am Bus gefunden wurden.

  Um unterscheiden zu können, welche Informationen für die Zentrale und welche für die Module am Bus sind, wird das Startzeichen überprüft. 0xFD => Nachricht für die Module 0xFE => Nachricht für die Zentrale

  Die Bedeutung der einzelnen Bytes:
  - **Startzeichen:**
    Bei Nachrichten an die Zentrale ist das Startzeichen immer 0xFE.
  - Länge:
    Die Länge enthält die Anzahl der Datenbytes zuzüglich der Anzahl der Bytes für die Checksumme.
  - Befehl:
    Der Befehl für den Discovery-Mode ist 0x01.
  - Checksumme: Die CRC16-Checksumme ist 2 Byte lang. Sie wird wieder mit dem Polynom 0x1002 berechnet.

  Der Aufruf zum Starten des Discovery-Mode sieht wie folgt aus:

  ```
  FE   04  01  F9 8E
  |    |   |   |
  |    |   |   Checksumme
  |    |   Befehl
  |    Länge
  Startzeichen
  ```

  Nun fängt die Zentrale an, den Bus nach angeschlossenen Modulen zu durchsuchen.

  Bei jedem gefundenen Modul sendet die Zentrale eine Nachricht. Die Antwortnachricht besteht im Prinzip aus einem I-Block, allerdings mit verkürzten Adressen. Sie bestehen nur aus einem Byte. Bis auf Ziel- und Absenderadresse entsprechen alle Bezeichnungen den bereits beschriebenen. Der Befehl ist hierbei mit 0x80,

Ziel- und Absenderadresse mit 0x00 und das Kontrollzeichen mit 0x98 festgelegt. Ist das Durchsuchen des Busses nach neuen Modulen abgeschlossen, wird eine Nachricht mit der Adresse 0xFFFFFFFF gesendet.

1.2.5. Absenderadresse

Die Übertragung der 4 Byte langen Absenderadresse erfolgt wie die der Zieladresse im Big-Endian-Format. Dabei wird das höchstwertige Byte als erstes Übertragen. Das niederwertigste Byte als letztes.

1.2.6. Framelänge

Die Framelänge enthält die Anzahl der Datenbytes und zuzüglich die länge der Checksumme

1.2.7. Framedaten

Pro Nachricht dürfen 64 Byte Framedaten (Nutzdaten) übertragen werden.
Bei Homematic Wired können bis zu 253? Bytes Nutzdaten übertragen werden Die von mir getesteten Module Empfangen und Senden zumindest brav diese längen. Bei der Kommunikation mit der Zentrale werden allerdings, wie bisher beobachtet nur 32 Bytes benutzt.

1.2.8. Checksumme

Die zwei Byte lange CRC16-Checksumme wird mit dem Polynom 0x1002 berechnet. Auch hier gilt Escape-Pflicht, falls ein Byte einem der Steuerzeichen entsprechen sollte. Siehe oben.

1.3. Befehlssatz

Jedes Modul besitzt einen Mikrocontroller mit integriertem EEPROM-Speicher (zumeist ein ATmega32). In diesem Speicher wird die Konfiguration der Module abgelegt. Jeder Eingang und Ausgang besitzt innerhalb des Moduls eine eindeutige Nummer. Wird ein Taster an einem Modul betätigt, so wird das EEPROM nach möglichen Zielaktoren durchsucht. Sind ein oder mehrere Aktoren gefunden, so wird eine Nachricht an die Aktoren der jeweiligen Module gesendet.
Die Steuerung von Modulen erfolgt mit nur wenigen einfachen Befehlen. Das Byte mit dem Befehl steht immer an der ersten Stelle der Framedaten. Die Liste der Befehle ist möglicherweise nicht komplett weil es ggf. noch weitere Modulspezifische Befehle gibt.

Folgende Module wurden bisher untersucht:

1. HMW-IO-12-Sw14-DR
2. HMW-IO-4-FM

Befehlsübersicht

Befehl	Hex-Code	Beschreibung
! ->	0x21	führt einen Reset des Moduls durch
A ?->	0x41	Announce ? neues Modul "ankündigen" **Bestätigung: mit "i" Nachricht**
C ->	0x43	Modulskonfiguration neu lesen
E ->	0x45	???
K <->	0x4B	Key-Event KEY_EVENT_LONG, KEY_EVENT_SHORT **Bestätigung: mit "i" Nachricht**
R ->	0x52	EEPROM lesen **Bestätigung: EEPROM-Data**
S ?->	0x53	Aktorzustand abfragen LEVEL_GET **Bestätigung: mit "i" Nachricht**
W ->	0x57	EEPROM schreiben
Z ->	0x5A	End Discovery Mode
c <->	0x63	Zieladresse löschen Bei HM Wired noch genutzt?
e ?	0x65	??? Antwort auf "E" - Befehl

Befehlsübersicht

Befehl	Hex-Code	Beschreibung
h ->	0x68	Modultyp und Hardware-Version abfragen **Bestätigung: entsprechende Modul-Daten**
i <-	0x69	INFO_LEVEL INFO_FREQUENCY
l ->	0x6C	Lock SET_LOCK
n <-	0x6E	??? HMW-IO-12-FM antwortet darauf **Bestätigung: mit "h" Nachricht**
q <->	0x71	Zieladresse hinzufügen Bei HM Wired noch genutzt?
s ?->	0x73	Aktor setzen
u ->	0x75	Update (Der Bootloader im Modul wird aktiviert)
v ->	0x76	Firmware-Version des Gerätes anfragen **Bestätigung: Firmware-Daten**
x ?->	0x78	LEVEL_SET TOGGLE_INSTALL_TEST STOP **Bestätigung: mit "i" Nachricht**
z ->	0x7A	Start Discovery Mode
Ë ?->	0xCB	Key-Sim ??? KEY_SIM_LONG KEY_SIM_SHORT **Bestätigung: mit "i" Nachricht**

Legende:

-> Befehle in der Regel von der Zentrale zum Modul
<- Befehle in der Regel vom Modul zur Zentrale
<-> Befehle in der Regel vom Modul zur Zentrale bzw. zu einem anderen Modul

1.3.1. "!" (0x21) - Modulreset

Hiermit kann man einen Neustart eines Moduls erzwingen. Damit nicht versehentlich ein Modul neu gestartet wird, muss das zweite Byte des Nachrichtenframes ebenfalls ein "!" enthalten.

1.3.2. "A" (0x41) - Announce???

Wenn ein Modul noch nicht an der Zentrale angelernt ist? schicken die Module nach einem Key-Event einen "Announce"-Befehl (Bsp.: A<5><18><0><3><6>heq28734<161><6>).

Dieser wird NUR einmal versendet, auch wenn kein ACK erfolgt. A-Befehle werden nicht bestätigt.

Nachrichtenaufbau:

1. Befehlsbyte A
2. Sensornummer
3. Modul-Type
4. Hardware Version (Da wird aber 0 gesendet?)
5. Firmware-Version (Stelle vor dem Punkt)
6. Firmware-Version (Stelle nach dem Punkt)
7. 10-Stellige Seriennummer

1.3.3. "C" (0x43) - Konfiguration neu lesen

Die Konfigurationsparameter aller Module werden im EEPROM gespeichert. Da sich nicht alle Änderungen im EEPROM direkt auf die Funktion auswirken ist in einigen Fällen ein erneutes auslesen der Konfigurationsparameter erforderlich. Der "C" - Befehl wird z.B. nach jedem Ändern von Modulparameter und Direktverknüpfung durch das Webinterface der Zentrale aufgerufen.

1.3.4. "E" (0x45) - ???

1.3.5. "K" (0x4B) - Key-Event (gegenüber HS485 anders)

Das Key-Event wird bei jedem Drücken und Loslassen eines an einem Modul angeschlossenen Tasters gesendet. Wird ein Taster länger betätigt, so wird in festen Zeit abständen erneut ein Key-Event übertragen. Nachrichten an die Broadcast-Adresse werden mit dem Zielaktor 0 versendet.

Es werden folgende Daten gesendet:

1. Befehlsbyte K
2. Nummer des Sensoreingangs
3. Nummer des Zielaktors
4. Event

E - entspricht dem Tasten -Event.
0 0 - wird nur bei Key-Sim-Befehlen benutzt?
0 1 - wird nur bei Key-Sim-Befehlen benutzt?
1 0 - Taste losgelassen (kurzer Tastendruck)
1 1 - Taste losgelassen (langer Tastendruck)

T - wird bei jedem Loslassen der Taste um eins erhöht

Die Bits im Event geben Informationen über das Tasten -Ereignis an:

Bit	7	6	5	4	3	2	1	0
	T	T	T	T	T	T	E	E

Bei "K" - Befehlen an die Broadcast Adresse bleibt Bit 3 (Nummer des Zielaktors) = 0

Direkt im Anschluss and einen "K" - Befehl wird ein "A" - Befehl gesendet. Scheinbar aber nur, wenn der "K" - Befehl an die Broadcast-Adresse (0xFFFFFFFF) gesendet wird

Bei langen Tastendruck wird alle x ms eine neue Nachricht versendet. Das Event-Bit bleibt bei jeder dieser Nachrichten gleich

Werden an einem Tastsensor mehrere Tasten gleichzeitig gedrückt, so wird eine Nachricht pro Taste gesendet.

1.3.6. "R" (0x52) - EEPROM lesen

Auslesen der Konfigurationsparameter aus dem EEPROM der Module. Maximal sind 252 Byte an EEPROM-Daten mit einem Befehl auslesbar. Dem "R" - Befehl folgt die 2 Byte lange Startadresse (Wieder im Big-Endian-Format) und ein Byte für die Anzahl der zu lesenden Datenbytes. Als Antwort wird dann der EEPROM-Inhalt gesendet.
Die Homematic Zentrale liest aber scheinbar immer nur 32 Byte aus.

Bei einem Test mit dem Modul "HMW-IO-12-FM" konnte ich aber bis zu 253 Bytes auf einmal auslesen. Beim Versuch 254 oder 255 Bytes auszulesen sind auch nur 253 Bytes gesendet wurden. Da 2 Bytes der Datenlänge noch für die Checksumme benötigt werden.
Eigentlich dürften aber nur 251 nutzbar sein, Da die Checksumme, sofern da Escaped werden muss bis zu 4 Bytes lang sein kann.

1. Befehlsbyte R
2. Höherwertiges Byte der Startadresse
3. niederwertiges Byte der Startadresse
4. Anzahl der zu lesenden Bytes

Wenn Byte 3 und 4 nicht gesendet werden scheinbar 242 mal 0xFF gesendet. Wiso. oder ist das ein Teil des EEPROMS?

1.3.7. "S" (0x53) - Aktor- / Sensorzustand abfragen

Der Befehl S gefolgt von der Aktor- / Sensornummer sendet den jeweiligen Zustand. Als Antwort wird zunächst die Aktornummer im Datenbyte 0 wiederholt. Im Datenbyte 1 steht der Aktor- / Sensorzustand.

1. Befehlsbyte S
2. Die Nummer des abzufragenden Aktors / Sensors

Eine Antwort auf einen S-Befehle erfolgt mit einer I-Nachricht mit dem "i" - Befehl.

1.3.8. "W" (0x57) - EEPROM schreiben

Mit dem "W" - Befehl werden Konfigurationsparameter direkt in das EEPROM eines Moduls geschrieben. Dem Befehl folgt eine 2 Byte lange Startadresse und ein Byte für die Anzahl der Datenbytes. Danach folgen die eigentlichen Daten. Da das Schreiben in das EEPROM einige Zeit dauert, sollte die maximale Anzahl an EEPROM-Daten pro Nachricht 32 Byte nicht überschreiten.
Jedes Modul kann auf die Werkseinstellung zurückgesetzt werden, indem das gesamte EEPROM mit 0xFF gefüllt wird. Danach ist ein Modulreset erforderlich.

Der Aufbau des EEPROMs aller Module kann aus den XML-Dateien in der Zentrale unter " /firmware/hs485types" entnommen werden.

"W" - Befehle werden durch das Modul nur mit einem "ACK" bestätigt

1.3.9. "Z" (0x5A) - End Discovery Mode ???

Ein "Z" - Befehl wird zwei mal gesendet nachdem der die Zentrale alle Discovery Nachrichten verschickt hat. Der "Z" - Befehl enthält keine Daten und wird von der Zenrale immer an die Broadcast Adresse geschickt.
(0x00000001 -> 0xFFFFFFFF)

Der "Z" - Befehl gibt die Ursprünglichen Modulfunktionen wieder frei, die durch den "z" - Befehl vorübergehen abgeschaltet wurden. Erst nach dem "Z" - Befehl können während des Discovery gefundene Module Ihre Meldungen an die Zentrale absetzen.

1.3.10. "c" (0x63) - Zieladresse löschen

Wie der "q" - Befehl. Nur wird diesmal dann die Verküpfung gelöscht.
Wird das noch bei Momematic Wired genutzt?

1.3.11. "e" (0x65) - ???

Antwort eines Moduls auf einen "E" - Befehl

1.3.12. "h" (0x68) - Modultyp und Hardware-Version abfragen

Als Antwort auf einen "h" - Befehl werden die Informationen zu Hardware-Typ und die Hardware-Version gesendet. Hardware-Typ und -Version benötigen jeweils ein Byte.

Bisher verfügbare Module mit Hardware-Typ und EEPROM-Größe

HW-Typ	Gerät	EEPROM
0x10 (16)	HMW-IO-4-FM RS485 I/O module 4-channel (flush-mount)	1024 Byte
0x11 (17)	HMW-LC-Sw2-DR RS485 switch actuator 2-channel (DIN rails)	1024 Byte
0x12 (18)	HMW-IO-12-Sw7-DR RS485 I/O module 12-channel in and switch actuator 7-channel (DIN rails)	1024 Byte
0x14 (20)	HMW-LC-Dim1L-DR RS485 dimming actuator 1-channel leading edge (DIN rails)	1024 Byte
0x15 (21)	HMW-LC-Bl1-DR RS485 blind actuator 1-channel (DIN rails)	1024 Byte
0x16 (22)	HMW-IO-SR-FM RS485 I/O SR	1024 Byte
0x19 (25)	HMW-Sen-SC-12-DR RS485 shutter contact 12-channel (DIN rails)	1024 Byte
0x1A (26)	HMW-Sen-SC-12-FM RS485 shutter contact 12-channel (flush-mount)	1024 Byte
0x1C (28)	HMW-IO-12-Sw14-DR RS485 I/O module 12-channel in and switch actuator 14-channel (DIN rails)	1024 Byte
0x1B (27)	HMW-IO-12-FM RS485 I/O module 12-channel (flush-mount)	1024 Byte

1.3.13. "i" (0x69) - Information ???

Anfragen an die Module z.B. durch den "S" - Befehl werden durch den "i" - Befehl beantwortet. Durch diese Antwort entfällt die Bestätigung durch eine ACK-Nachricht da der "i" - Befehl gleichzeitig die Antwort darstellt.

Einige Module können Logging-Nachrichten veschicken. Das sind auch "i" - Events. Logging Nachrichten gehen nur an die Zentrale?

Nachrichtenaufbau:
1. Befehlsbyte i
2. Nummer des Aktor / Sensor
3. Ab dem dritten Byte können Aktor- / Sensor-Werte übertragen werden.
   Z.b. bei HMW-IO-12-Sw14-DR: hier meldet der Analog Eingang in Bit 3 und 4 den Wert zwischen 0 und 1023 als unsigned long.

1.3.14. "l" (0x6C) - Lock (kleines L)

Mit diesem Befehl kann ein Aktor auf gesperrt (gelockt poder Inhibit) werden.

Nachrichtenaufbau:
1. Befehlsbyte I
2. 0 ??? möglicherweise niederwertige Byte der Aktor/Kanalnummer
3. Aktor / Kanalnummer ???
4. 1 - für gesperrt, 0 - für nicht gesperrt

1.3.15. "q" (0x71) - Zieladresse hinzufügen

Jedes Modul besitzt eine unterschiedliche Anzahl an Eingängen und Ausgängen. Um diese Ein- / Ausgänge mit den Ein- / Ausgängen anderer Module zu verknüpfen, müssen Zieladresse und Zielaktor im Modul gespeichert werden. Dies kann entweder direkt mit EEPROM-Schreibzugriffen durchgeführt werden oder mit dem "q" - Befehl. Dazu wird mit dem "q" - Befehl auch die Nummer des Eingangs und des Aktors, der programmiert werden soll, mitgesendet.

Bei Homematic werden die Direktverknüpfungen in der Regel über das Webinterface vorgenommen. Dabei wird die Verknüpfung direkt mit Schreibzugriffen in EEPROM it dem "W" - Befehl gesetzt. Der "q" - Befehl scheint hier nicht mehr verwendet zu werden.

1.3.16. "s" (0x73) - Aktor setzen

Setzt den Zustand eines Modul-Ausgangs. Der "s" Befehl wird in der Regel von der Zentrale aus gesendet.

Es gilt der folgende Nachrichtenaufbau:
1. Befehlsbyte "s"
2. Nummer des Sensoreingangs
3. Nummer des Zielaktors
4. Aktion

Je nach Aktor werden unterschieldliche Zustände im Aktion übertragen:
5. Schaltaktor:
   Aus - 0x00, An - 0x01, Toggle - 0xFF

6. Jalousie:
   Runter - 0x20, Hoch - 0x10, Aus - 0xFE, Auf Schlitz fahren - 0xFF

7. Dimmer:
   direktes Setzen - 0 - 16, herunterdimmen - 0x11, heraufdimmen - 0x12,
   herrauf, herrunter dimmen - 0x13, Toggle - 0x14, alter Wert - 0x15

Abweichender Nachrichtenaufbau für Homematic nach verschiedenen Tests:

1. Befehlsbyte "s"
2. Nummer des Zielaktors
3. Aktion

Folgende Zustände für das Aktor-Byte hab ich für die Homematic Aktoren gefunden:

- HMW-IO-12-Sw14-DR:
  hier wird beim Analog-Digital Ausgang die Frequens in Byte 3 und Byte 4 als unsigned long Übertragen

1.3.17. "u" (0x75) - Update

Startet den Update-Mode eines Moduls. Mit anderen Worten: Der Bootloader wird aktiviert.
Anschliessend kann eine neue Firmware in das Modul geflashed werden. Details dazu müssen noch ermittelt werden.

1.3.18. "v" (0x76) - Firmware-Version

Die Firmware-Version besteht aus zwei Bytes:
1. Vorkommastelle
2. Nachkommastelle.

Die Antwort auf das "v"-Event erfolgt ohne Event-Byte. es werden lediglich die 2 Versionsbytes übertragen.
Die Antwort wird 3 mal gesendet?

1.3.19. "x" (0x78) - ???

Ein "X" - Befehl wird wohl von der Zentrale zum Modul gesendet um per Script einen bestimmten Datenpunt zu verändern.

1.3.20. "z" (0x7A) - Start Discovery Mode

Ein "z" - Befehl wird zwei mal gesendet bevor die Zentrale Discovery Nachrichten verschickt. Der "z" - Befehl enthält keine Daten und wird von der Zenrale immer an die Broadcast Adresse geschickt. (0x00000001 -> 0xFFFFFFFF)

Durch das Senden des "z" - Befehles wird an allen am Bus angeschlossenen Modulen das Senden vorübergehend abgeschaltet. Während des Discovery-Modes funktionieren z.B. keine Direktverknüpfungen. Auch keine Modul internen. Die Module reagieren auch nicht mehr auf externe Befehle. Ausgenommen den "Z" - Befehl.

Der "Z" - Befehl gibt die Ursprünglichen Modulfunktionen wieder frei, die durch den "z" - Befehl vorübergehen abgeschaltet wurden. Erst nach dem "Z" - Befehl können während des Discovery gefundene Module Ihre Meldungen an die Zentrale absetzen.

1.3.21. "Ë" (0xCB) - Key-Sim - Event

Entspricht einem Key-Event mit gesetztem achtem Bit im Befehlsbyte. Der Key-Sim Event wird von der Zentrale aus benutzt und wird bei jedem Klick auf eine Tastenschaltfläche aus dem WebUI der Zentrale gesendet. Der Key-Sim-Event ist fast identisch mit dem Key-Event. Beim Befehlsbyte ist das Bit 7 gesetzt.

Es werden folgende Daten gesendet:

1. Befehlsbyte Ë
2. Nummer des Sensoreingangs
3. Nummer des Zielaktors
4. Event

   E - entspricht dem Tasten -Event.
   0 0 - kurzer Tastendruck (vom WebUI)
   0 1 - langer Tastendruck (vom WebUI)
   1 0 - wird nur bei Key-Befehlen benutzt?
   1 1 - wird nur bei Key-Befehlen benutzt?

   T - wird bei jedem Loslassen der Taste um eins erhöht

   Die Bits im Event geben Informationen über das Tasten -Ereignis an:

Bit	7	6	5	4	3	2	1	0
	T	T	T	T	T	T	E	E

5. Das 5. bis 8. Byte enthält noch einmal die Zieladresse des Aktors

Auf den Key-Sim - Befehl antwortet der Aktor mit einem i-Befehl

1.4. Eigene Beobachtungen.

- Falls keine Bestätigung kommt, erfolgen bis zu zwei Wiederholungen
- Die Wiederholung erfolgen nach ca. 190 ms.
- Vermutung: die Sendefolgenummer muss mit der jeweiligen Empfängeradresse, die Empfangsfolgenummer mit der jeweiligen Adresseradresse gespeichert werden.
- Befehle an Geräteinterne Verknüpfungen werden nicht bestätigt. Oder zumindestens nicht über den Bus. Das Key-Event von z.B. einem Taster wird zusätzlich an die Broadcast-Addresse (0xFFFFFFFF) geschickt.
  Im Anschluss erfolgt noch ein "A" - Befehl mit der Seriennummer auch an die Broadcast-Addresse.
- Beim Zurücksetzen auf Werkseinstellungen werden nur 16 Bytes mit 0xFF pro Nachricht verschickt. Jede einzelne Nachricht wird mit ACK bestätigt.
- Durch das setzen der Werkseinstellungen wird das entsprechende Modul auch gleichzeitig abgelernt.
- Zwischen 2 direkt hintereinander gesendeten Telegrammen vergehen ca. 7,5 ms.
- Beim "normalen" Ablernen eines Moduls (ohne Werkseinstellung) scheint es keinen weiteren Datenverkehr zu geben. Das Modul wird dabei scheinbar nur in der Zentrale gelöscht.
  Das ist möglicherweise ein Bug der CCU. Das gerät ist nämlich noch in der Liste der Weboberfläche vorhanden. Ein erneuter Löschversuch schlägt fehl mit dem Hinweis das das Gerät nicht an der CCU angemeldet sei. Nach einem Neuladen der Weboberfläche existiert das Gerät dann auch nicht mehr.
- Nachrichten an die Broadcastadresse werden von den Modulen nicht bestätigt.

1.5. Annahmen wegen Fehlender Informationen

1.6. Informationen aus dem Quellcode des HS484 Kernel Modul der LCU1

- Bei "besetztem" Bus erfolgt eine Zufällige Wartezeit von 5 bis 20 ms
- Nach jedem Discovery Frame wird 15ms auf Antwort gewartet

2. Module

- **Anzahl der Verknüpfungen zwischen Modulen = 100**
  Alle Module können interne Verknüpfungen speichern. Die Anzahl der internen Verknüpfungen ist auf 100 beschränkt. Das bedeutet dass z.B. ein einzelner Tastereingang eine Beziehung zu max. 100 einzelnen Relaisausgängen speichern kann.

2.1. Ideen für den Bau eigener Module

- Mehrfach Lichtschalter mit Led-Rückmeldung
  z.B. für diese Multitaster:
  http://www.mikrocontroller.net/topic/189119
  http://www.haus-bus.de/index.php?show=products

- 2-Fach-Motorsteuerung für meine Jalousien die bisher noch per Eigenbau FS20-Modul gesteuert werden

- LED-RGB-Dimmer (Ggf. auch per Funk)

- Infrarot-Modul (Sender) (Ggf. auch per Funk)
  z.B. zum Steuern von TV, Radio usw.

- Infrarot-Modul (Empfänger) (Ggf. auch per Funk)
  Koppelbar mit IR-Sender (siehe oben)

- Generelles Datenmodul. ggf. um LCD usw. einzubuinden

- HM-RF <--> HM Wired Buskoppler
  Um z.B. Direktverknüpfungen zwischen Funk- und Wired Modulen zu ermöglichen
  z.B. zum Steuern von Wirde-Dimmer mit Funk Taster (langer Tastendruck)

2.1.1. HMW-HB-IO - Homebrew IO Modul

Integrierter Befehlssatz

- Empfangsbefehle
  ! (0x21), C (0x43), K (0x4B), R (0x52), S (0x53), W (0x57), Z (0x5A), h (0x68), l (0x6C), s (0x73), v (0x76), x (0x78), z (0x7A), Ë (0xCB)

- Sendebefehle
  A (0x41), K (0x4B), i (0x69),

- noch unklar
  E (0x45), e (0x65),