Homematic Wired RS485 Protokollbeschreibung

Inhalt

1. Grundsätzliches 2
1.1. Datenübertragung 3
1.1.1. Vermeidung von Kollisionen auf dem RS485 Bus 3
1.1.2. Addressen und Seriennummern 4
1.2. Protokollrahmen 4
1.2.1. Start- und Steuerzeichen 4
1.2.2. das Escape-Zeichen 5
1.2.3. Zieladresse 5
1.2.4. Kontrollzeichen 5
1.2.5. Absenderadresse 8
1.2.6. Framelänge 8
1.2.7. Framedaten 8
1.2.8. Checksumme 8
1.3. Befehlssatz 8
1.3.1. "!" (0x21) - Modulreset 11
1.3.2. "A" (0x41) - Announce??? 11
1.3.3. "C" (0x43) - Konfiguration neu lesen 11
1.3.4. "E" (0x45) - ??? 11
1.3.5. "K" (0x4B) - Key-Event (gegenüber HS485 anders) 12
1.3.6. "R" (0x52) - EEPROM lesen 12
1.3.7. "S" (0x53) - Aktor- / Sensorzustand abfragen 13
1.3.8. "W" (0x57) - EEPROM schreiben 13
1.3.9. "Z" (0x5A) - Zero-Communication Mode End 13
1.3.10. "c" (0x63) - Zieladresse löschen 13
1.3.11. "e" (0x65) - ??? 13
1.3.12. "g" (0x67) - ??? nur im Bootloader Mode 13
1.3.13. "h" (0x68) - Modultyp und Hardware-Version abfragen 14
1.3.14. "i" (0x69) - Information ??? 15
1.3.15. "l" (0x6C) - Lock (kleines L) 16
1.3.16. "n" (0x6E) - Seriennummer abfragen 16
1.3.17. "p" (0x70) - Packetgröße abfragen (nur im Bootloader-Mode) 16
1.3.18. "q" (0x71) - Zieladresse hinzufügen 16
1.3.19. "r" (0x72) - Firmwaredaten lesen (nur im Bootloader-Mode) 16
1.3.20. "s" (0x73) - Aktor setzen 17
1.3.21. "u" (0x75) - Update 17
1.3.22. "v" (0x76) - Firmware-Version 17
1.3.23. "w" (0x77) - Firmware-Daten schreiben (nur im Bootloader-Mode) 18
1.3.24. "x" (0x78) - LEVEL_SET 18
1.3.25. "z" (0x7A) - Zero-Communication Mode Start 18
1.3.26. "Ë" (0xCB) - Key-Sim - Event 19
1.4. Firmware Updates über den RS485 Bus 20
1.4.1. Der Update-Prozess 20
1.5. Eigene Beobachtungen. 24
1.6. Annahmen wegen Fehlender Informationen 24
1.7. Informationen aus dem Quellcode des HS484 Kernel Modul der LCU1 24
2. Module 25
2.1. Ideen für den Bau eigener Module 25
2.1.1. HMW-HB-IO - Homebrew IO Modul 25
3. Direktverknüpfungen 26
3.1. Bedingungen 26
4. History 29

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 Vermutlich werden CSMA/CA Techniken eingesetzt.

- Die Module "überwachen" permanent den Bus (Carrier Sense)
- Ist der Bus für eine bestimmte Zeit (DIFS*) frei (welche muss noch rausgefunden werden) wartet das Modul noch eine zufällige Zeit (Backoff) (ggf. auch abhängig der Geräteadresse).

  Ist der Bus weiterhin frei, kann das Modul senden.
- Nach vollständigem Empfang einer Nachricht wartet das Modul noch eine gewisse Zeit (SIFS**) und sendet dann die ACK Nachricht oder die Antwort.
- Bleibt die Antwort aus (missing ACK) wird nach einer Wartezeit (EIFS***) die Nachricht bis zu zwei mal wiederholt.

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.

* **DIFS -** Distributed Coordination Function Interframe Spacing:
Die Zeit, die vor dem Senden eines regulären Datenrahmens vergangen sein muss.

** **SIFS -** Short Interframe Spacing:
Die Zeit, die vor dem Senden eines Bestätigungsframes (ACKs)

*** **EIFS** - Extended Interframe Spacing:
Die Zeit, die vor dem Senden nach einer erkannten Kollision vergangen sein muss. (100ms)

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 0x000003ED zu 0x000002AC gesendet.

```
                                        Framelänge
                                        |     Nutzdaten
                                        |__   |___________
FD   00 00 03 ED   1C   00 00 02 DE   06   4B  01  00  0C   F9 8E
|    |            |    |                                   |
|    Zieladresse  |    Absenderadresse                     Checksumme
Startzeichen      Kontrollzeichen
```

1.2.1. Start- und Steuerzeichen

Das Startzeichen ist ein 1 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). Mit diesen Nachrichten bestätigt ein Modul Befehle während eines laufenden Firmwareupdates.
- 0xFC: Escape-Zeichen (siehe nächster Abschnitt)

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.

Die Homematic CCU hat selber die Adresse 0x00000001. Verarbeitet / antwortet jedoch auch auf Nachrichten mit der Adresse 0x00000001 bis 0x000000FF.

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 Sendefolgenummer an, so werden beide bestätigt, aber nur eine verarbeitet.

  Die Zentrale muss die Sendefolgenummer aller Module Überwachen und pro Modul tracken.
  Müssen alle Module auch die Sendefolgenummer aller angelernten Module Überwachen?

  Wenn eine Nachricht von der Gegenseite nicht bestätigt wurde, wird die Sendefolgenummer um eins verringert

- **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. Welche Nachrichten sind das? Discovery Nachrichten enthalten generell keine Absenderadresse

- **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 es 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.

  Das Synchronisationsbit setzen die Module nach einem Neustart oder Reset.
  Ein Z bzw. ein z-Befehl wird auch mit gesetztem Y-Bit gesendet

- **M - Adressmaske:**
  Wird eine Discovery-Nachricht versendet, so werden die M-Bits als Adressmaske benutzt. Sie gibt an, wie viele Bits (M+1) der Empfängeradresse mit der Zieladresse verglichen 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 wird unmittelbar nach der Empfangenen Nachricht gesendet. Der Absender wartet bis zu 100ms auf die ACK-Nachricht. Ist die Zeit verstrichen wird die Ursprüngliche Nachricht noch bis zu zwei mal wiederholt. Bleibt die Bestätigung weiterhin aus, so wird von einer Kommunikationsstörung ausgegangen. Grundsätzlich werden ALLE Befehle mindestens mit einer ACK-Nachricht beantwortet. Auch Unbekannte.

  Die CCU1 wartet 200ms bevor das Telegram wiederholt wird. Es werden insgesamt 3 sendeversuche unternommen.

- **Discovery-Nachricht:**
  Mit einer Discovery-Nachricht scannt die Zentrale alle am Bus angeschlossenen Module. Alle Module vergleichen mit Hilfe der Adressmaske die Zieladresse mit Ihrer eigenen Adresse. Die Adressmaske bestimmt wie viele Bits verglichen werden. Beginnend beim höchstwertigen Bit.

  Beispiel:

  ```
  Moduladresse: 000059ED -> 0000 0000 0000 0000 0101 1001 1110 1101
  Zieladresse:     00000000 -> 0000 0000 0000 0000 0000 0000 0000 0000

  Adressmaske (Konrollzeichen der Discovery Nachricht):
  5B -> 0101 1011 >> 3 = 0b1011 = 0x0B = 11
  Also werden die ersten 11+1 Bit verglichen

  0000 0000 0000 0000 0101 1001 1110 1101
  0000 0000 0000 0000 0000 0000 0000 0000

  Ergebniss: Wahr -> 0xF8
  ```

  Bei Übereinstimmung der Anzahl der Bits aus Adressmaske, mit der Adresse des Modul und der Zieladresse sendet das Modul ein 0xF8. Die Zentrale erkennt dies und stellt dadurch fest, dass sich mindestens ein Modul mit dieser Adressmaske am Bus befindet. Die Zentrale erwartet die Antwort vom Modul innerhalb von 15ms (8ms). Timing aus dem Dump eines Discovery-Scans der CCU1. Ansonsten geht die Zentrale davon aus, das es keine Übereinstimmung gab.

  Offene Frage: Wenn die Adressen mehrere Module auf die Adressmaske passen: welches Modul sendet das 0xF8 ? Laut Scan wird nur einmal F8 gesendet? Irgend ein Modul sendet, und die anderen Module bleiben Stumm, weil ein anderes schon F8 gesendet hat?

  Anschließend pass Zentrale Adressmaske und ggf. die Zieladresse nach folgender Regel an:

  Der Discovery-Scan beginnt immer bei Zieladresse 00000000.

  Adressmaske < 31
  Modul sendet 0xF8: Adressmaske wird um Eins erhöht.
  Modul ohne Antwort: In der vorherigen Zieladresse wird die Bitposition der Adressmaske + 1 auf 1 gesetzt

  Adressmaske = 31
  Adressmaske wird um Eins verringert und die vorherige Zieladresse wird um eins veringert.
  Anschliessend wird der weitere Scan rückwärts weiter geführt.

  Wenn die Adressmaske den Wert = 31 erreicht hat (alle 32 Bits von Moduladresse und Zieladresse stimmen überein) wurde die Adresse des Moduls gefunden.

  Wenn keine Übereinstimmung gefunden wurde. also kein 0xF8 vom Modul gesendet wurde, wiederholt die Zentrale das letzte Packet noch zwei mal. der Scan fortgesetzt.

  Dies wird so lange durchgeführt, bis alle Module am Bus gefunden wurden.
  Hinweis: Im Untersuchten Quellcode der LCU1 wird der Scan nach 256 gefundenen Modulen beendet.
  ist das bei Homematic auch so? Ggf. mal ausprobieren.

  ```
  Der Aubfau einer Discovery-Nachricht sieht wie folgt aus:

  FD  00000000  03   02  F9 8E
  |     |          |     |    Checksumme
  |     |          |     Länge
  |     |          Kontrollzeichen
  |     Zieladresse
  Startzeichen
  ```

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.

Absendeadressen von 00 00 00 01 bis 00 00 00 FF "gehören" scheinbar der Homematic CCU. Zumindest antwortet diese auf Nachrichten mit Empfängeradressen aus diesem Bereich

1.2.6. Framelänge

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

1.2.7. Framedaten

Pro Nachricht dürfen bis zu 253 Bytes Nutzdaten übertragen werden. In der Regel werden allerdings nur 32 Bytes an Daten pro Nachricht übertragen. Einzige bisher beobachtete Ausnahme:
Während der Übertragung von Firmwareupdates an die Module werden 128 Bytes pro Nachricht übertragen.

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

cmd	Hex	Beschreibung / Frame-Typen	Dir	Antwort	IO-4-FM	LC-Sw2-DR	IO-12-Sw7-DR	LC-Dim1L-DR	LC-Bl1-DR	Sen-SC-12-DR	Sen-SC-12-FM	IO-12-FM	IO-12-Sw14-DR
					Devices supports frames								
!	0x21	Reset (Modul-Reset ohne Bootloader-Start)	To	ACK	A	A	A	A	A	A	A	A	A
A	0x41	Announce ? neues Modul "ankündigen"	From	I 0x69 (i) ???									
C	0x43	Modulkonfiguration neu lesen	To	ACK	A	A	A	A	A	A	A	A	A
E	0x45	???	To	I / ACK 0x65(e) ???									
K	0x4B	KEY_EVENT_LONG, KEY_EVENT_SHORT, KEY_SIM_SHORT, KEY_SIM_LONG	From To	I 0x69 (i)	D	D	D	D	D			D	
R	0x52	EEPROM lesen	To	I Eeprom Data	A	A	A	A	A	A	A	A	A
S	0x53	Aktorzustand abfragen LEVEL_GET	To	I 0x69 (i)	D	D	D	D	D	D	D	D	D
W	0x57	EEPROM schreiben	To	ACK	A	A	A	A	A	A	A	A	A
Z	0x5A	Zero-Communication Mode End Beendet den mit "z" gestarteten Mode	To	ACK	A	A	A	A	A	A	A	A	A
c	0x63	Zieladresse löschen bei HS485, wenn Device im Programmiermode ist. Wird das bei HM-Wired noch genutzt?	???	ACK									
e	0x65	??? Antwort auf "E" - Befehl	???	ACK									
g	0x67	**nur im Bootloader-Mode,** Wird nach dem Ende vom Firmwarevergleich an das Modul gesendet	To	ACK ???	B	B	B	B	B	B	B	B	B
h	0x68	Modultyp abfragen	To	I Modul Data	A	A	A	A	A	A	A	A	A
i	0x69	INFO_LEVEL INFO_FREQUENCY	From	ACK	D	D	D	D	D	D	D	D	D
l	0x6C	SET_LOCK	To	ACK	D	D	D	D	D			D	
n	0x6E	Seriennummer abfragen	**To**	I Serial	A	A	A	A	A	A	A	A	A
p	0x70	Packetgröße abfragen (**nur im Bootloader-Mode)**	To	I Packetsize	B	B	B	B	B	B	B	B	B
q	0x71	Zieladresse hinzufügen bei HS485, wenn Device im Programmiermode ist. Wird das bei HM-Wired noch genutzt?	???	ACK / ???									
r	0x72	Firmwaredaten lesen (**nur im Bootloader-Mode)**	To	ACK / I FW-Data	B	B	B	B	B	B	B	B	B
s	0x73	Aktor setzen	To	ACK									D
u	0x75	Update Bootloader Start, anschließender Reset	To	-	A	A	A	A	A	A	A	A	A
v	0x76	Firmware-Version des Gerätes abfragen	**To**	I FW-Version	AB	AB	AB	AB	AB	AB	AB	AB	AB
w	0x77	Firmwaredaten schreiben, **nur im Bootloader-Mode**	To	ACK ???	B	B	B	B	B	B	B	B	B
x	0x78	LEVEL_SET, TOGGLE_INSTALL_TEST, STOP	To	I 0x69 (i)	D	D	D	D	D			D	
z	0x7A	Zero-Communication Mode Start	To	ACK	A	A	A	A	A	A	A	A	A
Ë	0xCB	Key-Sim, KEY_SIM_LONG, KEY_SIM_SHORT ???	To	I 0x69 (i)									

Legende:
A Alle Module unterstützen diesen Nachrichtentyp (Frame)
B Dieser Nachrichtentyp (Frame) wird nur im Bootloader-Mode vom Modul unterstützt
D Gerätespezifischer Nachrichtentyp (Frame)

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.
Bestätigung mit einer ACK-Nachricht

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.
Bestätigung mit einer ACK-Nachricht

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 (alle 300ms) 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. Der K-Befehl an die Broadcast-Adresse wird z.B. gesendet wenn eine Taste gedrückt wird welche intern eines Gerätes zu einem Aktor zugewiesen wurde.

K-Befehle werden von einem "Empfänger" scheinbar nur ausgewertet wenn der "Sender" an den "Empfänger" angelernt wurde.

Bei langen Tastendruck wird alle 300ms 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) - Zero-Communication Mode End

Ein "Z" - Befehl wird zwei mal gesendet nachdem der die Zentrale alle Discovery Nachrichten verschickt hat bzw. nachdem die Zentrale den Firmware transfer eines Modules abgeschlossen 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. "g" (0x67) - ??? nur im Bootloader Mode

Der "g" - Befehl wird von der Zentrale an ein Modul gesendet nachdem die Firmwareaktualisierung für ein Modul abgeschlossen wurde. Weitere Details sind noch nicht bekannt

1.3.13. "h" (0x68) - Modultyp abfragen (Hardware Version???)

Als Antwort auf einen "h" - Befehl werden die Informationen zum Hardware-Typ gesendet. Der Hardware-Typ benötigt zwei Bytes. Das erste Byte beschreibt den Hardwaretype. Das zweite Byte soll wohl den Sub-Type beschreiben. Dieser ist bei den bisher untersuchten Modulen aber immer 0 gewesen.

Bisher verfügbare Module mit Hardware-Typ und EEPROM-Größe

HW-Typ	Gerät	EEPROM
0x10 (16)	HMW-IO-4-FM RS485 4fach-IO-Modul Unterputzmontage	1024 Byte
0x11 (17)	HMW-LC-Sw2-DR RS485-Schaltaktor, 2fach Hutschinenmontage	1024 Byte
0x12 (18)	HMW-IO-12-SW7-DR RS485-IO-Modul 12 Eingänge, 7 Schaltausgänge Hutschinenmontage	1024 Byte
0x14 (20)	HMW-LC-DIM1L-DR RS485-Dimmaktor 1fach, Phasenanschnitt Hutschinenmontage	1024 Byte
0x15 (21)	HMW-LC-BL1-DR RS485-Rollladenaktor, 1fach Hutschinenmontage	1024 Byte
0x16 (22)	HMW-IO-SR-FM Dieses Modul ist nicht verfügbar, obwohl es in der CCU eine Firmware gibt	1024 Byte
0x19 (25)	HMW-SEN-SC-12-DR RS485 Schließerkontakt, 12 Eingänge Hutschinenmontage	1024 Byte
0x1A (26)	HMW-SEN-SC-12-FM RS485 Schließerkontakt, 12 Eingänge Unterputzmontage	1024 Byte
0x1B (27)	HMW-IO-12-FM RS485 IO-Modul, 12-Fach Unterputzmontage	1024 Byte
0x1C (28)	HMW-IO-12-SW14-DR RS485 IO-Modul 12 Eingänge 14 Ausgänge Hutschinenmontage	1024 Byte

Das verwendete Namensschema für die Gerätebezeichnungen:

HMW: RS485 (HomeMatic-Wired)

IO: Eingang / Ausgang (Input / Output)
LC: Licht / Energie (Light Control)
SEN: Sensor
DIMxL: Dimmer mit x Kanälen Leistungsdimmer für ohmsche/induktive Lasten
SWx: Schaltaktor mit x Kanälen (Switch)
SC: Tür- / Fenster / Schliesserkontakt (Shutter Contact)
BLx: Jalousieaktor mit x Kanälen (Blind)

1: Einkanal
2: Zweikanal
4: Vierkanal
7: Siebenkanal
12: Zwölfkanal
14: Vierzehnkanal

FM: Unterputzmontage (Flush Mounted)
DR: Hutschienenmontage (DIN Rail)

1.3.14. "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.

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

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

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

1.3.16. "n" (0x6E) - Seriennummer abfragen

Als Antwort auf einen "n" - Befehl wird vom Modul die 10-Stellige Seriennummer gesendet.

1.3.17. "p" (0x70) - Packetgröße abfragen (nur im Bootloader-Mode)

Mit einem "p" - Befehl fragt die Zentrale vor einem Firmwareupdate-Prozess den Bootloader des betreffenden Modules an fest wie viel Bytes pro Nachricht das Modul akzeptiert. Die Antwort bzw. die Bestätigung auf den "p" - Befehl erfolgt mit einer "i" - Nachricht mit Startzeichen 0xFE. Die Länge der zurück gemeldeten Packetgröße beträgt ein Byte.

1.3.18. "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.

1.3.19. "r" (0x72) - Firmwaredaten lesen (nur im Bootloader-Mode)

Nach dem Schreiben der Firmwaredaten während eines Firmwareupdates eines Modules, wird der Programmspeicher eines Modules gelesen und überprüft. Mit dem "r" - Befehl wird das Lesen der Firmware aus dem Programmspeicher angefordert.
Dem Befehl folgt eine 2 Byte lange Startadresse und ein Byte für die Anzahl der zu lesenden Bytes.

1.3.20. "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 Zielaktors
3. Aktion / Wert
4. Aktion / Wert

Folgende Zustände für Byte 3 und 4 habe ich für folgende Homematic Aktoren gefunden:

- HMW-IO-12-Sw 7-DR:
  Byte 3: 0x00 -> Aus, 0x01 - 0xFE -> Ein (Homematic sendet 0xC8), 0xFF -> Toggle

- HMW-IO-12-Sw14-DR:
  Byte 3 und 4:
  - Schalt-Ausgang: 0x0000 -> Aus, 0x0001 - 0xFFFF -> Ein (Homematic sendet 0x03FF)
  - Analog-Digital Ausgang: Frequenz in Milliherz als unsigned long (0x1000 -> 1Hz)

Abweichend Funktion ELV-HS484
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

1.3.21. "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.
Siehe Abschitt 1.4.
Einem "u" - Befehl folgt keine Bestätigung. Der Bootloader wird soffort aktiviert.

1.3.22. "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?

1.3.23. "w" (0x77) - Firmware-Daten schreiben (nur im Bootloader-Mode)

Mit dem "w" - Befehl wird bei aktivem Bootloader die Firmware in den Programmspeicher des betreffenden Modules geschrieben. Dem Befehl folgt eine 2 Byte lange Startadresse und ein Byte für die Anzahl der Bytes die geschrieben werden sollen. Danach folgen die eigentlichen Daten.

Der "w" - Befehl wird mit einem Datenframe ohne Absenderadresse, also ohne gesetztes B-Bit vom Kontrollzeichen an das Modul übertragen. Siehe Abschnitt 1.4.

"w" - Befehle werden durch das Modul nur mit einem "ACK" bestätigt. Die Bestätigungsnachricht enthält dabei ein 0xFE als Startzeichen.

1.3.24. "x" (0x78) - LEVEL_SET

Ein "x" - Befehl wird wohl von der Zentrale zum Modul gesendet um per Script einen bestimmten Datenpunt zu verändern. Wird auch gesendet wenn man per WebUI der CCU einen "Ausgang" eines Modules ändert

Es werden folgende Daten gesendet:

1. Befehlsbyte x
2. Nummer des Zielaktors
3. Aktion / Wert

Folgende Werte für Byte habe ich für folgende Homematic Aktoren gefunden:

- HMW-IO-12-Sw 7-DR:
  0x00 -> Aus, 0x01 - 0xFE -> Ein (Homematic sendet 0xC8), 0xFF -> Toggle

- HMW-IO-12-Sw14-DR:
  Reagiert nicht auf den x-Befehl

1.3.25. "z" (0x7A) - Zero-Communication Mode Start

Ein "z" - Befehl wird zwei mal gesendet bevor z.B. die Zentrale Discovery Nachrichten verschickt bzw. bevor der Bootloader eines Gerätes aktiviert wird. Der "z" - Befehl enthält keine Daten und wird von der Zentrale 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 und eines Firmwareupdates eines Modules funktionieren z.B. keine Direktverknüpfungen. Auch keine Modul internen. Die Module reagieren auch nicht mehr auf externe Befehle. Ausgenommen den "Z"- und den "u" - 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.

1.3.26. "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 - kurzer Tastendruck (vom WebUI)
   1 - langer Tastendruck (vom WebUI)

   T - ist ein Counter und wird bei jedem Loslassen der Taste um eins erhöht. Bei einem Überlauf fängt der Counter wieder bei 0 an zu zählen. Die Funktionsweise des Counters muss noch geprüft werden

   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

5. Das 5. bis 8. Byte enthält noch einmal die Zieladresse des Aktors. Stimmt die Zieladresse hier nicht überein, wird der Aktor nicht geschaltet. Welcher tiefere Sinn steckt hier dahinter?

Key-Sim-Befehle werden von allen "Empfänger" mit der entsprechenden Adresse ausgewertet. Im Gegensatz zu K-Befehlen, wo die Auswertung in "Empfängern" scheinbar nur stattfindet, wenn "Sender" und "Empfänger" zuvor verknüpft (gepaired) wurden. Somit ist dieser Befehl für das Schalten der Aktoren von der Zentrale aus bestens geeignet.

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

Die Beschreibung hier stimmt nicht mit der Frame-Berschreibung in den XML-Dateien der Gerätebeschreibungen überein. Hier fehlt z.B. die Beschreibung des der Bytes 5 bis 8 Keine Ahnung wieso.

1.4. Firmware Updates über den RS485 Bus

Die Firmware der HomeMatic Wired Module können über den RS485 Bus mit neuen Firmware-Versionen aktualisiert werden. Pro Updatevorgang kann zur gleichen Zeit immer nur ein Modul aktualisiert werden. Während der Aktualisierung wird der Bus für jeglichen anderen Datenverkehr gesperrt.
Das Protokoll ist grundsätzlich dem der normalen Kommunikation vergleichbar bzw. identisch.

Beim Firmware-Update wird der Inhalt der in der CCU-Firmware mitgelieferten HEX-Files in den AVR Mikrocontroller übertragen. Damit die Zentrale weiß welches Firmwarefile zu welchem Modul gehört, gibt es ein Mapping-File. Das Mapping-File liegt im Ordner "/firmware/wfmap" der CCU. In diesem File wird beschrieben welches Hardware-Modul zu welcher Firmwaredatei passt. Die HEX-Adresse des Firmware-Files in welcher die aktuelle Firmware-Version abgelegt ist wird auch beschrieben.

Ein Beispiel-Eintrag im Mapping-File:

H16V0 hmw_io_4_fm_hw0.hex @0x77F0
| | |
| | Adresse der Versionsinformation im Hexfile
| Hex-File
Modultyp (Hier 0x16)

Die Versionsinformation im Hexfile wird in 2 Bytes ab der oben angegebenen Adresse gespeichert. Dabei ist allerdings zu beachten dass die Angegebene Adresse der decodierten Adresse (siehe Hexfile-Format) entspricht.
Im ersten Byte ist die Zahl nach dem Komma, und im zweiten Byte die Zahl vor dem Komma gespeichert.

Der Update-Prozess

Bevor der Bootloader des zu aktualisierenden Modules aktiviert wird, wird eine z-Nachricht an alle Teilnehmer (Broadcast) des Busses gesendet. Damit wird sicher gestellt, dass keine Kommunikation auf dem Bus mehr stattfindet. Der "z" - Befehl wird zwei mal hintereinander gesendet. Anschließend sendet die Zentrale einen "u" - Befehl an das Modul, was für das ein Firmwareupdate ausgewählt wurde. Der "u" - Befehl aktiviert in diesem Modul den Bootloader für eine kurze Zeit. Anschließend startet der eigentliche Aktualisierungsvorgang.

Hier der bisher beobachte Ablauf:

```
                          (KZ: Y RF B S )     S  R Y F B
- Zentrale -> Broadcast: "z"    (KZ: 1001 1010)   I[1](0,Y,F,B)
- Zentrale -> Broadcast: "z"    (KZ: 1001 1100)   I[2](0,Y,F,B)
- Zentrale -> Modul: "u"        (KZ: 0011 1110)   I[3](1,  F,B)
- Modul -> Zentrale: ACK        (KZ: 0111 1001)   ACK (3,  F,B)

Die Folgende Kommunikation erfolgt ohne Absenderadresse:
                          (KZ: Y RF B S )     S  R Y F B
- Zentrale -> Modul: "u"        (KZ: 0011 0000)   I[0](1,  F  )
- Modul -> Zentrale: ACK (FE)   (KZ: 0001 0001)   ACK (0,  F  )
- Zentrale -> Modul: "p"        (KZ: 0011 0010)   I[1](1,  F  )
- Modul -> Zentrale: "I" (FE)   (KZ: 0011 0000)   I[0](1,  F  ) (Antwort auf den vorherigen "p" - Befehl.
Hier wird vermutlich festgelegt wie groß die Datenpakete sind mit der die Firmwaredaten vom Modul erwartet werden.
Im beobachteten Fall standen hier 2 Bytes: 0x00 und 0x80.

                          (CC: Y RF B S )     S  R Y F B
- Modul -> Zentrale: ACK (FE)   (CC: 0001 0001)   I[1](1   F  )
```

Anschließend beginnt das Senden der Programmdaten. Nach jedem Block sendet das Modul eine Bestätigung mit einem 0xFE Startzeichen.

In diesem Beispiel erfolgt die Übertragung der Firmware-Daten in Blöcken zu je 128 (0x80) Bytes (Antwort auf den "p" - Befehl, siehe oben). durch einen "w" - Befehl. Dabei wird keine Absender-Adresse übertragen.

Gesendet wird abwechselnd 128 Bytes. Darauf erfolgt vom Modul eine Bestätigung. Die Bestätigung wird mit dem Starzeichen 0xFE übertragen.

Wo ist eigentlich definiert wie groß der Programmspeicher des betreffenden Modules ist. Oder wird der komplette Inhalt des Hex-Files übertragen? Das vermute ich mal.

Hier exemplarisch vier Programmdatenpackete mit anschließender Bestätigung durch das Modul:

- Zentrale -> Modul FD 00 00 82 01 14 86 77 00 00 80 0C 94 EF 1D 18 95 18 95 18 95 18 95 18 95 0C 94 5A 01
18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 0C 94 1D 1D 18 95 18 95 18 95 18 95
18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 44 65 62 75
67 2D 4D 6F 64 65 20 6F 6E 20 00 00 00 3D 05 44 05 4C 05 53 05 5A 05 60 05 66 05 6C 05 72 05 77 05 7D 05 82
5B 40
- Modul -> Zentrale: FE 00 51 02 11 5C

```
(CC: Y RF B S )         S  R Y F B
 CC: 0101 0001        ACK (2   1  )
```

- Zentrale -> Modul FD 00 00 82 01 16 86 77 00 80 80 05 87 05 8B 05 90 05 94 05 98 05 9C 05 A0 05 A4 05 A7 05 AB
05 AE 05 B1 05 B4 05 B7 05 B9 05 BC 05 BF 05 C1 05 C3 05 C6 05 C8 05 CA 05 CC 05 CE 05 D0 05 D2 05 D4 05 D5
05 D7 05 D9 05 DA 05 DC 05 DE 05 DF 05 E1 05 E2 05 E4 05 E5 05 E6 05 E8 05 E9 05 EB 05 EC 05 ED 05 EF 05 F0
05 F1 05 F3 05 F4 05 F6 05 F7 05 F8 05 FA 05 FB 05 FC 7C 05 FC 7E 05 FF 05 01 06 02 06 04 06 05 06 07 06 08
06 09 32 E6
- Modul -> Zentrale: FE 00 71 02 75 18

```
(CC: Y RF B S )         S  R Y F B
 CC: 0111 0001        ACK (3   1  )
```

- Zentrale -> Modul FD 00 00 82 01 10 86 77 01 00 80 06 0B 06 0C 06 0E 06 0F 06 11 06 13 06 14 06 16 06 17 06 19
06 1A 06 1C 06 1E 06 1F 06 21 06 22 06 24 06 26 06 27 06 29 06 2B 06 2C 06 2E 06 2F 06 31 06 33 06 34 06 36
06 37 06 39 06 3B 06 3C 06 3E 06 3F 06 41 06 42 06 44 06 45 06 47 06 48 06 4A 06 4B 06 4D 06 4E 06 50 06 51
06 52 06 54 06 55 06 56 06 58 06 59 06 5A 06 5B 06 5D 06 5E 06 5F 06 60 06 61 06 62 06 64 06 65 06 66 06 67
6A E8
- Modul -> Zentrale: FE 00 11 02 D9 D4

```
(CC: Y RF B S )         S  R Y F B
 CC: 0001 0001        ACK (0   1  )
```

- Zentrale -> Modul FD 00 00 82 01 12 86 77 01 80 80 06 68 06 69 06 6A 06 6B 06 6C 06 6D 06 6E 06 6F 06 70 06 71
06 72 06 73 06 75 06 76 06 77 06 78 06 79 06 7A 06 7B 06 7D 06 7E 06 7F 06 81 06 82 06 83 06 85 06 87 06 88
06 8A 06 8C 06 8E 06 90 06 92 06 94 06 96 06 99 06 9C 06 9E 06 A1 06 A4 06 A7 06 AB 06 AE 06 B2 06 B6 06 BA
06 BE 06 C3 06 C8 06 CD 06 D2 06 D7 06 DD 06 E3 06 EA 06 F1 06 F8 06 FF 06 07 07 FF 07 FF 10 E0 19 C0 20 E0
FC 7E 70
- Modul -> Zentrale: FE 00 31 02 BD 90

```
(CC: Y RF B S )         S  R Y F B
 CC: 0011 0001        ACK (1   1  )
```

Nach der Übertragung der Programmdaten erfolgt eine Überprüfung der geschriebenen Daten:

```
                        (CC: Y RF B S )     S  R Y F B
- Zentrale -> Modul: "p"  (KZ: 0001 0100)   I[2](0,  F  )
- Modul -> Zentrale: "I" (FE)  (KZ: 0101 0010)   I[1](2,  F  ) (Antwort auf den vorherigen "p" - Befehl.
```

Hier wird vermutlich wieder festgelegt wie groß die Datenpakete sind mit der die Firmwaredaten vom Modul erwartet werden. Im beobachteten Fall standen hier 2 Bytes: 0x00 und 0x80.

```
                        (CC: Y RF B S )     S  R Y F B
- Modul -> Zentrale: ACK (FE)  (CC: 0011 0001)   ACK (1   F  )
```

Das Lesen beginnt dann hier: (hier 2 Beispiele)

- Zentrale -> Modul FD 00 00 82 01 36 06 72 00 00 80 9F 30
- Modul -> Zentrale FE 00 74 82 0C 94 EF 1D 18 95 18 95 18 95 18 95 18 95 18 95 0C 94 5A 01 18 95 18 95 18 95 18
95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 0C 94 1D 1D 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18
95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 44 65 62 75 67 2D 4D 6F 64 65 20
6F 6E 20 00 00 00 3D 05 44 05 4C 05 53 05 5A 05 60 05 66 05 6C 05 72 05 77 05 7D 05 82 1E 26
- Modul -> Zentrale FE 00 51 02 11 5C (ACK) ???

- Zentrale -> Modul FD 00 00 82 01 50 06 72 00 80 80 64 BE
- Modul -> Zentrale FE 00 16 82 05 87 05 8B 05 90 05 94 05 98 05 9C 05 A0 05 A4 05 A7 05 AB 05 AE 05 B1 05 B4 05 B7 05 B9 05 BC 05 BF 05
C1 05 C3 05 C6 05 C8 05 CA 05 CC 05 CE 05 D0 05 D2 05 D4 05 D5 05 D7 05 D9 05 DA 05 DC 05 DE 05 DF 05 E1 05 E2 05 E4 05 E5 05 E6 05
E8 05 E9 05 EB 05 EC 05 ED 05 EF 05 F0 05 F1 05 F3 05 F4 05 F6 05 F7 05 F8 05 FA 05 FB 05 FC 7C 05 FC 7E 05 FF 05 01 06 02 06 04 06 05
06 07 06 08 06 09 AA FA
- Modul -> Zentrale FE 00 71 02 75 18 (ACK) ???

```
                          (CC: Y RF B S )     S  R Y F B
- Zentrale -> Modul: "g"  (KZ: 0011 0110)   I[3](1,  F  )
- Zentrale -> Modul: "g"  (KZ: 0011 0110)   I[3](1,  F  )
- Zentrale -> Modul: "g"  (KZ: 0011 0110)   I[3](1,  F  )
```

Warum Wird das 3 mal gesendet. Und for allem was ist der "g" Befehl? Steht "g" Vielleicht für "good" also alles ok?

```
                          (CC: Y RF B S )     S  R Y F B
- Modul -> Zentrale: ACK  (KZ: 0111 1001)   ACK (3,  F,B)
Diese ACK kommt von der Adresse 0x00000000 ?
```

Die folgenden Befehle werden wieder mit langen Befehlen (0xFD) und mit Absenderadresse gesendet

```
                          (KZ: Y RF B S )     S  R Y F B
- Zentrale -> Broadcast: "Z"  (KZ: 1001 1110)   I[3](0,Y,F,B)
- Zentrale -> Broadcast: "Z"  (KZ: 1001 1000)   I[0](0,Y,F,B)
```

Nach dem Freischalten der Buskommunikation durch den "Z" - Befehl werden nun noch die Hardware-Version ("h" - Befehl) und die neue Firmware-Version ("v" - Befehl) abgefragt.

Damit ist die Firmwareaktualisierung des Modules abgeschlossen.

Offene Fragen:
Was passiert wenn die Firmwareaktualisierung durch einen Übertragungsfehler fehl schlägt?
Ist das Modul dann "Unbrauchbar" oder kann man den Aktualisierungsvorgang wiederholen? Theoretisch sollte das funktionieren. Da durch die Aktualisierung nicht den Bootloader-Bereich verändert.

1.5. 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.
- Die CCU arbeiten alle Anforderungen zu sendenden Nachrichten der Reihe nach ab. Z.B. aus abgearbeiteten Scripten. Es wird immer erst eine Message bestätigt bevor die Nächste gesendet wird. Wenn es zu einer Nachricht keine Bestätigung gibt, wird erst auf alle Sendeversuche gewartet bevor die neue Nachricht aus der Queue abgearbeitet wird

1.6. Annahmen wegen Fehlender Informationen

1.7. 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 bis zu 8ms auf Antwort gewartet
- während einer Discovery-Aktion können max. 255 Geräte erkannt werden. Sind mehr als 255 Geräte am Bus, bricht Discovery ab.

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),

3. Direktverknüpfungen

Über Direktverknüpfungen können direkte Verbindungen zwischen einem Sender (Taster, Sensor o.ä) und einem Aktor (Dimmer, Relais ö.ä.) erstellt werden, die auch dann funktionieren, wenn die HomeMatic-Zentrale nicht funktioniert oder auch gar nicht am Bus angeschlossen ist. Je nach Modul können "einfache" in der Zentrale vordefinierte Profile (z.B. eine Treppenhauslichtfunktion) erstellen oder auch komplexe Einstellungen (der so genannte Experten-Mode) vorgenommen werden. Wobei die vordefinierten Aktionen am Ende auch nur die "Experten Einstellungen" entsprechend verändern.

Je nach verknüpften Aktor stehen unterschiedliche Parameter zur Verfügung. So existieren die Parameter RampOn / RampOff z.B. nur bei Dimmern.

Für kurze (SHORT) und lange (LONG) Tastenaktionen existiert je Parametersatz

Im Prinzipiell kann man mit den "Experten-Parameter" ein Art Ablaufsteuerung erstellen, die dann durch einen kurzen oder langen Tastendruck des entsprechenden verknüpften Senders ausgelöst wird. Über Sprungbefehle bzw. Vergleichsoperationen lassen sich auf diese Weise recht komplexe Abläufe (z.B. eine Blinkfunktion) programmieren, die mit einem Programm so nicht oder nur sehr umständlich (oder mit Hilfe von Skripten) möglich wären.

3.1. Bedingungen

Je nach Empfängertyp gibt es verschiedene Parameter die für die Ablaufsteuerung herangezogen werden können. Zusammen mit der Vergleichsoperation führen diese dann beim Erreichen der Bedingung ein Aktion oder optionales Sprungziel aus. Folgende Bedingungen (CT = Condition Threshold) stehen zur Verfügung:

Bedingungen (CT = Condition Threshold)	
CT_RAMPOFF	Ausschalt-Rampe
CT_RAMPON	Einschalt-Rampe
CT_OFFDELAY	Ausschalt-Verzögerung
CT_ONDELAY	Einschalt-Verzögerung
CT_OFF	Ausschalt-Verweildauer
CT_ON	Einschalt-Verweildauer

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

Operation	Vergleichslogik	**Werteeinfluss**	Erläuterung
X GE **COND_VALUE_LO**	X >= **COND_VALUE_LO**	LO	X größer oder gleich (**g**reater/**e**qual) Vergleichswert LO

- UI_HINT

  SENSOR

- CHANNEL

- SHORT_ON_TIME_MODE
  Minimal, Absolute

- SHORT_OFF_TIME_MODE
  Minimal, Absolute

- SHORT_TOGGLE_USE
  DONT_USE, DIRECT, INVERTED

- SHORT_ACTION_TYPE
  INACTIVE, ACTIVE

- SHORT_ONDELAY_TIME

- SHORT_ON_TIME

- SHORT_OFFDELAY_TIME

- SHORT_OFF_TIME

- SHORT_JT_ONDELAY

- SHORT_JT_ON

- SHORT_JT_OFFDELAY

- SHORT_JT_OFF

- LONG_ON_TIME_MODE

- LONG_OFF_TIME_MODE

- LONG_TOGGLE_USE

- LONG_MULTIEXECUTE

- LONG_ACTION_TYPE

- LONG_ONDELAY_TIME

- LONG_ON_TIME

- LONG_OFFDELAY_TIME

- LONG_OFF_TIME

- LONG_JT_ONDELAY

- LONG_JT_ON

- LONG_JT_OFFDELAY

- LONG_JT_OFF

4. History

Version	Datum	Änderungen
87	07.04.2013	Liste der Änderungen: - 1.2.1 Start- und Steuerzeichen aktualisiert - 1.2.4 Weitere Erläuterungen zu Discovery - 1.2.7 Framedaten aktualisiert - 1.3 Befehlssatz aktualisiert - 1.3.9 "Z" (0x5A) - Zero-Communication Mode End - 1.3.13 HM Namensschema hinzugefügt - 1.3.16 "n" (0x6E) - Seriennummer abfragen - 1.3.21 Update - 1.3.25 "z" (0x7A) - Zero-Communication Mode Start - 1.3.26 Key-Sim - Event - 1.5 Eigene Beobachtungen Neue Abschnitte: - 1.3.12 "g" (0x67) - ??? nur im Bootloader Mode - 1.3.17 "p" (0x70) - Packetgröße abfragen (nur im Bootloader-Mode) - 1.3.19 "r" (0x72) - Firmwaredaten lesen (nur im Bootloader-Mode) - 1.3.23 "w" (0x77) - Firmware-Daten schreiben (nur im Bootloader-Mode) - 1.4 Firmware Updates über den RS485 Bus
86	08.01.2013	Liste der Ergänzungen: - 1.1.1 Vermeidung von Kollisionen auf dem RS485 Bus - 1.2.3 Zieladresse - 1.2.4 Kontrollzeichen Sendefolgenummer, Synchronisationsbit, Adressmaske, ACK-Nachricht, Discovery-Nachricht Ergänzt - 1.2.5 Absenderadresse - 1.3.5 "K" (0x4B) - Key-Event (gegenüber HS485 anders) - 1.3.18 "s" (0x73) - Aktor setzen - 1.3.22 "x" (0x78) - LEVEL_SET - 1.3.24 "Ë" (0xCB) - Key-Sim - Event - 3 Direktverknüpfungen Inhalt stammt teilweise von homematic-inside.de und ist noch nicht vollständig. - 4 History
0	-	Initiale Version