

# **FS20Nut & FHTNut**

868Mhz Funkinterface für das Ethernut 1.3

Steuerung von ELV  
FS20 Komponenten  
und  
FHT80b Heizungsreglern  
unter Nut/OS

**[www.fhtnut.de](http://www.fhtnut.de)**

Thomas Stollberg

12.01.2010

# Inhaltsverzeichnis

<b>1 Grundlagen zur Übertragung</b>	<b>1</b>
1.1 Physikalische Übertragung . . . . .	1
1.2 Datenübertragung . . . . .	1
<b>2 Das FHT80b Protokoll</b>	<b>3</b>
2.1 Ansteuerung der Stellantriebe . . . . .	3
2.2 Kommunikation mit dem FHT80b Regler . . . . .	3
2.3 Datenpakete . . . . .	3
2.3.1 Befehlsbyte . . . . .	4
2.3.2 Adressbyte . . . . .	4
2.3.3 Befehlerweiterungsbyte . . . . .	4
2.4 Bekannte Werteregister . . . . .	5
2.4.1 Wochenprofil . . . . .	5
2.4.2 Sonstige Werteregister . . . . .	6
2.5 Codierung . . . . .	6
2.5.1 Uhrzeiten Wochenprofil . . . . .	6
2.5.2 Ist-Temperatur . . . . .	7
2.5.3 Soll-Temperatur . . . . .	7
<b>3 FS20 Steuerbefehle</b>	<b>8</b>
3.1 FS20 Funktionen . . . . .	8
3.2 Steuerbefehle . . . . .	8
<b>4 KeyMatic®</b>	<b>9</b>
4.1 Ansteuerung des KeyMatic® Türschlossantrieb . . . . .	9
<b>5 Senden und Empfangen mit dem Atmega128</b>	<b>10</b>
5.1 Senden . . . . .	10
5.1.1 Schaltung zum Senden . . . . .	10
5.1.2 Software zum Senden . . . . .	11
5.2 Empfangen . . . . .	12
5.2.1 Schaltung zum Empfangen . . . . .	12
5.2.2 Software zum Empfangen . . . . .	13
<b>6 Steuerung über den embedded Webserver</b>	<b>15</b>
6.1 Integration in das Beispielprojekt HTTP Daemon (httpd) . . . . .	15
6.2 Steuerung der FHT80b Regler . . . . .	15
6.2.1 Status der Regler abfragen . . . . .	15
6.2.2 Werte eines Reglers verändern . . . . .	18
6.3 FS20 Steuerbefehle senden . . . . .	18
6.4 FS20 Steuerbefehle empfangen . . . . .	19
<b>7 Protokollbeispiele</b>	<b>20</b>
7.1 Anmeldung der Zentrale an einem FHT80b Regler . . . . .	20
7.2 FHT80b Regler übermittelt kompletten Status . . . . .	23

# 1 Grundlagen zur Übertragung

## 1.1 Physikalische Übertragung

Das FS20- und das FHT80b-Protokoll arbeiten auf 868 Mhz. Für dieses Projekt nutze ich ein 868 Mhz ELV-Sendemodul und ein 868 Mhz ELV-Superhet-Empfangsmodul.

Um eine logische 1 zu Übertragen wird die Datenleitung für  $600\mu\text{s}$  auf High und anschließend für  $600\mu\text{s}$  auf Low gesetzt. Für eine logische 0 wird die Datenleitung für  $400\mu\text{s}$  auf High und  $400\mu\text{s}$  auf Low gesetzt. FHT80b Befehle sind genauso aufgebaut.

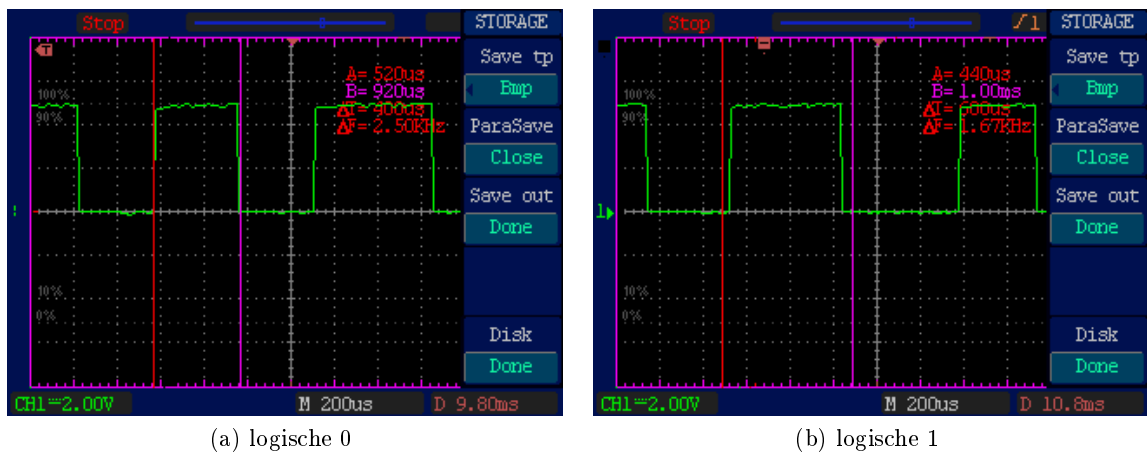


Abb. 1: Sendeport des Atmega128 auf einem Oszilloskop

## 1.2 Datenübertragung

Die Datenübertragung beginnt immer mit einer Synchronisationssequenz von 12 logischen nullen, gefolgt von einer logischen eins.

0 0 0 0 0 0 0 0 0 0 0 0 1



Abb. 2: FS20 Synchronisation auf einem Logic Analyzer

Danach erfolgt die Übertragung der Daten. Das gesamte Datenpaket besteht aus 5 oder 6 Bytes je nachdem ob noch ein zusätzliches Befehlsweiterungsbyte benötigt und mit übertragen wird.

Es werden folgende Bytes der Reihe nach Übertragen:

Hauscode1	(hc1)
Hauscode2	(hc2)
Adresse	(adr)
Befehl	(cmd)
(Befehlsweiterung)	(ext)
Prüfsumme	(chk)

Nach jedem Byte wird noch ein Paritätskontrollbit zur Paritätsprüfung übertragen.

Die Prüfsumme ergibt sich aus der Quersumme aller (5 oder 6) Bytes. Handelt es sich bei den übertragenen Daten um einen FS20 Steuerbefehl so wird zu der Quersumme noch die Konstante 0x06 addiert. Bei einem Steuerbefehl an einen FHT80b Regler wird die Konstante 0x0C addiert.

Ob ein Befehlsweiterungsbyte übertragen wird, wird im Befehlsbyte festgelegt. In diesem werden nämlich nur die fünf niederwertigen Bits (Bit0 - Bit4) für den eigentlich Befehl benötigt.

Die drei höherwertigen Bits (Bit5 - Bit7) werden genutzt um das Protokoll selbst zu steuern. Ist Bit5 gesetzt, also  $\text{Bit5} = 1$ , zeigt es an, dass im Anschluss an das Befehlsbytes ein Befehlsweiterungsbyte übertragen wird. Bit6 und Bit7 werden zur Kommunikation mit einem FHT80b Heizungsregler benötigt.

## 2 Das FHT80b Protokoll

### 2.1 Ansteuerung der Stellantriebe

Alle 116s sendet der FHT80b Regler seine Steuerinformationen an die Stellantriebe. Diese Nachricht werden 2x mit einem Abstand von 10ms gesendet.



Abb. 3: FHT80b Nachricht an die Stellantriebe

Es kann pro Sendezyklus immer nur ein Stellantrieb oder gleich alle Stellantriebe angesteuert werden. Nachrichten an die Ventile erkennt man am Befehlsbyte. Das Befehlserweiterungsbyte enthält in diesem Fall die Ventilstellung (0 = ZU, 255 = 100% AUF). Das Adressbyte enthält die Adresse des Stellantriebs (1-8). Ist das Adressbyte gleich 0 werden damit alle Stellantriebe angesprochen.

### 2.2 Kommunikation mit dem FHT80b Regler

Nach dem Erhalt der Steuerinformationen an die Stellantriebe hat die Zentrale nach etwa 80ms die Möglichkeit die Kommunikation zu beginnen. Geschieht dies nicht, kann der FHT80b Regler nach etwa 200ms die Kommunikation starten um geänderte Werte zu übermitteln. Nachrichten zwischen FHT80b Regler und Zentrale werden nur einmal gesendet! Nur derjenige der die Kommunikation begonnen hat, hat die Möglichkeit Werte zu ändern, der andere bestätigt nur den Erhalt der Datenpakete und hat sonst keinen Einfluss auf den Kommunikationsablauf. Gesendete Informationen werden vom FHT80b Regler nur bestätigt, wenn diese auch wirklich neu sind!

### 2.3 Datenpakete

Der Aufbau der Datenpakete ist dem des FS20 Protokolls identisch nur das Befehlsweiterungsbyte wird hier immer mitgesendet! Es enthält die eigentliche Information.

Es gibt zwei Arten von Datenpaketen:

- Protokollpakete
- Informationspakete

Protokollpakete steuern die Kommunikation, mit Informationspaketen werden neue Werte übermittelt. Informationspakete können nur von dem Kommunikationsinitiator gesendet werden.

Ob es sich um ein Protokoll- oder Informationspaket handelt erkennt man am Befehlsbyte. Empfangene Datenpakete werden bestätigt, indem sie zurückgesendet werden. Nur der Absender im Befehlsbyte wird geändert. Die Zentrale sendet die Bestätigung 90ms nach dem Erhalt des Datenpakets, der FHT80b Regler nach 60ms.

### 2.3.1 Befehlsbyte

Am Befehlsbyte lässt sich erkennen ob es sich um ein Protokoll- oder um ein Informationspaket handelt, es kann nur folgende vier Werte annehmen:

0x67 = 0b01100111 Protokollpaket gesendet vom FHT80b Regler  
0x69 = 0b01101001 Informationspaket gesendet vom FHT80b Regler  
0x77 = 0b01110111 Protokollpaket gesendet von der Zentrale  
0x79 = 0b01111001 Informationspaket gesendet von der Zentrale

Das Befehlerweiterungsbit ist also immer gesetzt! Entspricht das Befehlsbyte keinem der genannten Werte handelt es sich um ein Datenpaket vom FHT80b Regler an die Stellantriebe.

### 2.3.2 Adressbyte

Das Adressbyte in einem Protokollpaket steuert die Kommunikation, in einem Informationspaket gibt es an, welcher Wert im Befehlerweiterungsbyte übertragen wird (z.B. Soll-Temperatur, Tag-Temperatur, Nacht-Temperatur, Modus, o.a.).

In einem Protokollpaket kann das Adressbyte einen der fünf folgenden Werte annehmen:

1. 0x53 Kommunikationsanfrage
2. 0x54 FHT80b Regler ist bereit Daten zu empfangen
3. 0x7D Starte Übertragung
4. 0x4B Abschluss Wertepaket
5. 0x7E Beende Übertragung

Der FHT80b Regler sendet nach der Bestätigung für den Erhalt der Kommunikationsanfrage (Adressbyte = 0x53) ein Protokollpaket (Adressbyte = 0x54) und signalisiert dadurch die Bereitschaft Daten zu empfangen. Danach kann die Zentrale mit dem Protokollpaket (Adressbyte = 0x7D) die Übertragung starten. Diesen Sonderfall, dass der Regler 2 verschiedene Protokollpakete hintereinander sendet muss im Programmcode extra berücksichtigt werden. Siehe auch die entsprechenden Protokollbeispiele.

### 2.3.3 Befehlerweiterungsbyte

Das Befehlerweiterungsbyte enthält in einem Informationspaket die eigentliche Information (z.B. Temperaturangabe).

In einem Protokollpaket enthält es den Zentralencode. Bei einem Datenpaket an die Stellantriebe enthält das Befehlerweiterungsbyte die Ventilstellung (0 = ZU, 255 = 100% AUF).

## 2.4 Bekannte Werteregister

### 2.4.1 Wochenprofil

0x14	Montag von 1
0x15	Montag bis 1
0x16	Montag von 2
0x17	Montag bis 2
0x18	Dienstag von 1
0x19	Dienstag bis 1
0x1a	Dienstag von 2
0x1b	Dienstag bis 2
0x1c	Mittwoch von 1
0x1d	Mittwoch bis 1
0x1e	Mittwoch von 2
0x1f	Mittwoch bis 2
0x20	Donnerstag von 1
0x21	Donnerstag bis 1
0x22	Donnerstag von 2
0x23	Donnerstag bis 2
0x24	Freitag von 1
0x25	Freitag bis 1
0x26	Freitag von 2
0x27	Freitag bis 2
0x28	Samstag von 1
0x29	Samstag bis 1
0x2a	Samstag von 2
0x2b	Samstag bis 2
0x2c	Sonntag von 1
0x2d	Sonntag bis 1
0x2e	Sonntag von 2
0x2f	Sonntag bis 2

(Quelle: fhz4linux.info)

Nach der Übertragung aller Werte eines Wochentages wird ein Datenpaket zum Abschluss des Wertepakets gesendet.

### 2.4.2 Sonstige Werteregister

0x3e	0=auto 1=manuell 2=Urlaub
0x3f	Urlaub Endeuhrzeit oder Endetag
0x40	Urlaub Endetag 10=heute 11=morgen 0x=Endemonat
0x41	aktuelle Solltemperatur
0x42	ist Temperatur - LowByte
0x43	ist Temperatur - HighByte
0x44	bit0=Batterie 0=OK 1=leer bit5=Fenster 0=zu 1=offen
0x60	Jahr (0x60h - 0x64) werden anscheinend nur geschrieben
0x61	Monat
0x62	Tag
0x63	Stunde
0x64	Minute
0x65	Abfrageregister für die Register 0x14 - 0x2f (bit 0-6)
0x66	Abfrageregister (0x42,0x43,0x44 (bit 0) - 0x3e (bit 1) - 0x82,0x84,0x8a (bit 2)
0x69	Unbekannt
0x82	Tag Temperatur
0x84	Nacht Temperatur
0x8a	Fenster offen Temperatur

(Quelle: fhz4linux.info)

Nach der Übertragung eines Wertes wird immer noch ein Datenpaket zum Abschluss des Wertepakets gesendet.

## 2.5 Codierung

### 2.5.1 Uhrzeiten Wochenprofil

Die Uhrzeiten des Wochenprofils, also die Register 0x14 - 0x2f sind in 10min Schritten codiert.

0:00	0x00
0:10	0x01
1:00	0x06
12:00	0x48
23:50	0x8F
frei	0x90

Die Umrechnung einer codierten Uhrzeit in das hh:mm Format erfolgt folgendermaßen:

hh = (codierte Uhrzeit) DIV 0x06

mm = [(codierte Uhrzeit) MOD 0x06] \* 10

### 2.5.2 Ist-Temperatur

$0x3C = 60_{10}$	$6,0\text{ °C}$
$0xCD = 205_{10}$	$20,5\text{ °C}$
$0xFF = 255_{10}$	$25,5\text{ °C}$
$0x104 = 260_{10}$	$26,0\text{ °C}$
$0x12C = 300_{10}$	$30,0\text{ °C}$

Bei Temperaturen unter  $25,5\text{ °C}$  reicht es also das Werteregister  $0x42$  auszuwerten (da  $0x43 = 0$ ). Bei Temperaturen über  $25,5\text{ °C}$  muss auch das Register  $0x43$  ausgewertet werden.

### 2.5.3 Soll-Temperatur

Die Codierung für die Soll-Temperatur gilt auch für die Tag- und Nacht-Temperatur. Sie unterscheiden sich von der Codierung für die Ist-Temperatur da sie nur in 0,5er Schritten eingestellt werden können, sie belegen deshalb auch nur 1 Byte. Um Temperaturen zu codieren werden sie mit dem Faktor 2 multipliziert.

$0x0B = 11_{10}$	$5,5\text{ °C} = \text{AUS}$
$0x0C = 12_{10}$	$6,0\text{ °C}$
$0x28 = 40_{10}$	$20,0\text{ °C}$
$0x3E = 61_{10}$	$30,5\text{ °C} = \text{EIN}$

### 3 FS20 Steuerbefehle

FS20 Steuerbefehle (z.B. zum Schalten von Funksteckdosen) werden 3x mal hintereinander mit einem Abstand von 10ms gesendet.



Abb. 4: FS20 Steuerbefehl

#### 3.1 FS20 Funktionen

#### 3.2 Steuerbefehle

0x00	aus
0x01	dimmen 6,25%
0x02	dimmen 12,5%
0x03	dimmen 18,75%
0x04	dimmen 25%
0x05	dimmen 31,25%
0x06	dimmen 37,5%
0x07	dimmen 43,75%
0x08	dimmen 50%
0x09	dimmen 56,25%
0x0a	dimmen 62,5%
0x0b	dimmen 68,75%
0x0c	dimmen 75%
0x0d	dimmen 81,25%
0x0e	dimmen 87,5%
0x0f	dimmen 93,75%
0x10	dimmen 100%
0x11	ein
0x12	aus oder ein
0x13	dimmen heller
0x14	dimmen dunkler

(Quelle: fhz4linux.info)

## 4 KeyMatic®

### 4.1 Ansteuerung des KeyMatic® Türschlossantrieb

Zur Ansteuerung des KeyMatic Türschlossantriebs kann man entweder das ELV KeyMatic KM 300 RI Remote Interface oder das ELV KeyMatic®-Push-Button-Interface KM300 PBI nutzen. Diese geben einem die Möglichkeit, den Keymatic Türschlossantrieb über einen potentialfreien Kontakt (Relais, Optokoppler) zu steuern.

## 5 Senden und Empfangen mit dem Atmega128

Das Ethernet 1.3 Board ist mit einem Atmel Atmega128 Mikrocontroller ausgestattet.

### 5.1 Senden

#### 5.1.1 Schaltung zum Senden

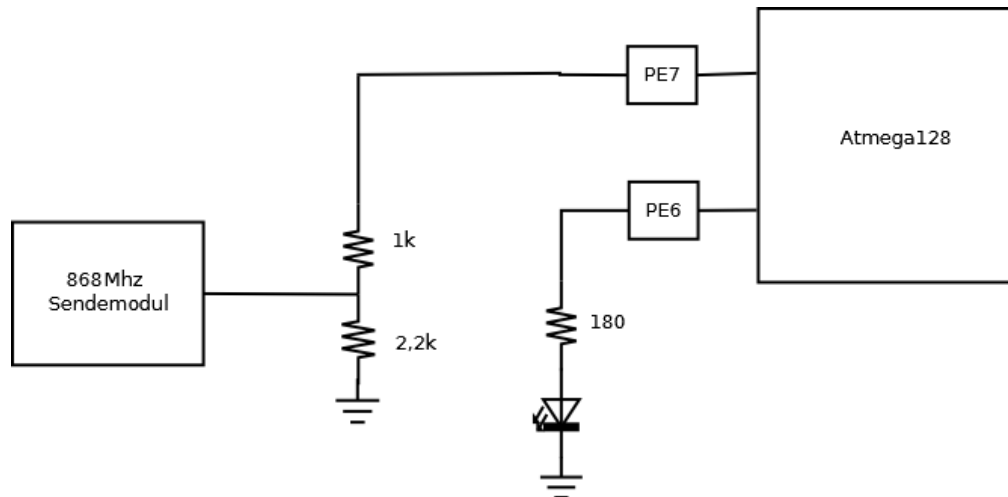


Abb. 5: Sendeschaltung

### 5.1.2 Software zum Senden

```
1 #define FS20_DELAY_ZERO 400 /* 400us */
2 #define FS20_DELAY_ONE 600 /* 600us */
3 /**
4  * Sendet eine logische eins
5  */
6 void fs20_send_one(void) {
7     sbi(PORTE, 7);
8     NutMicroDelay(FS20_DELAY_ONE);
9     cbi(PORTE, 7);
10    NutMicroDelay(FS20_DELAY_ONE);
11 }
12 /**
13  * Sendet eine logische null
14  */
15 void fs20_send_zero(void) {
16     sbi(PORTE, 7);
17     NutMicroDelay(FS20_DELAY_ZERO);
18     cbi(PORTE, 7);
19     NutMicroDelay(FS20_DELAY_ZERO);
20 }
```

Den Quellcode zum senden von FS20 bzw. FHT Befehlen konnte ich größtenteils von Alexander Neumann und seiner Implementation für das etherrape Board übernehmen.

## 5.2 Empfangen

Um das Empfangen generell zu ermöglichen, muss in der main Funktion die Funktion `fs20_fht_init()` aufgerufen werden. Diese nimmt die erforderlichen Einstellungen vor und startet die Threads die zum Empfangen benötigt werden.

### 5.2.1 Schaltung zum Empfangen

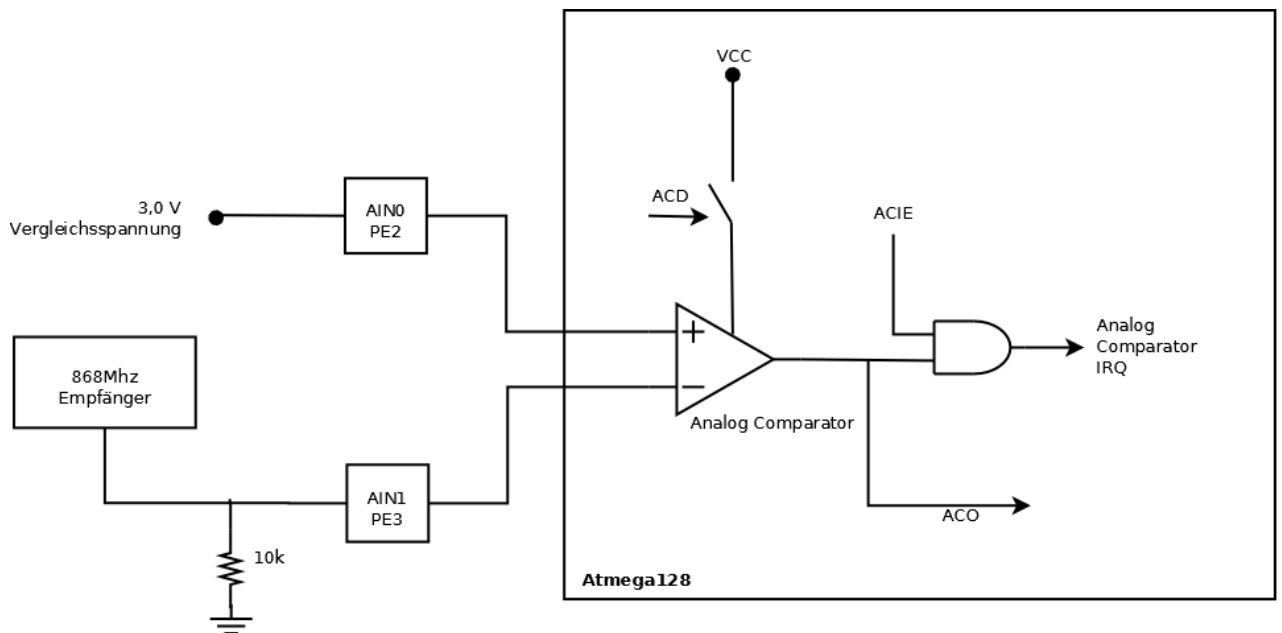


Abb. 6: Empfangsschaltung

### 5.2.2 Software zum Empfangen

Um das Rauschen des 868 Mhz Empfängers so weit es geht zu unterdrücken nutze ich zum Empfangen der Signale den Analog Comparator des Atmega128. Dieser löst ein Interrupt aus wenn die Spannung am PORTE 3 über oder unter die Vergleichsspannung an PORTE 2 steigt oder fällt. Als Vergleichsspannung nutze ich die 3,0 V die zum Betrieb des 868 Mhz Sendemodul benötigt werden.

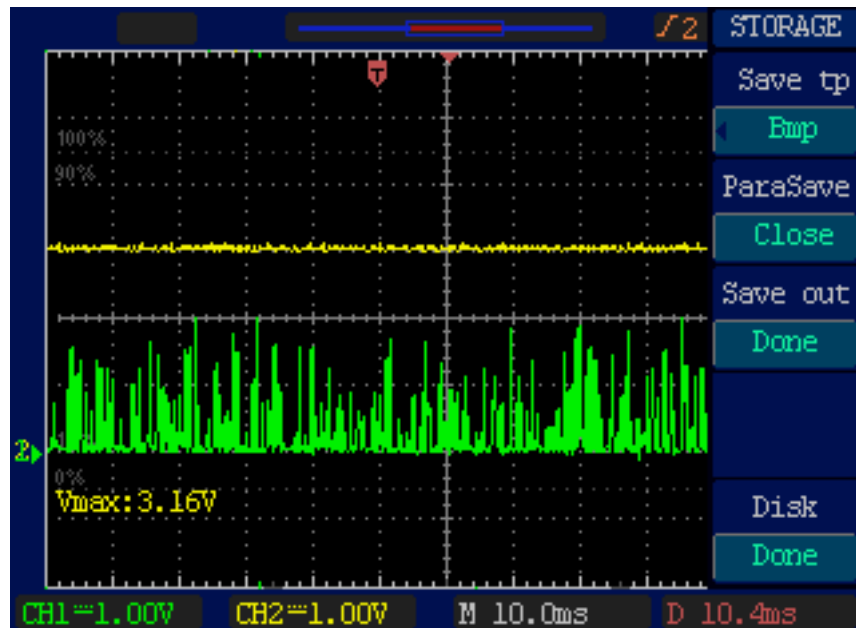


Abb. 7: Rauschen des 868Mhz Empfängers

Tritt am Analog Comparator ein Interrupt auf wird die Prozedur `int_ac` aufgerufen.

```

1 static void int_ac(void *arg) {
2     NutIrqDisable(&sig_COMPARATOR);
3     NutEventPostFromIrq(&event_int_ac);
4 }

```

Diese deaktiviert in Nut/OS den Interrupt vom Analog Comparator und löst das Event `event_int_ac` aus. Die eigentliche Verarbeitung eines Interrupts übernimmt der Thread `AC_INT`.

```

1  THREAD(AC_INT, arg)
2  {
3    memset(&fs20_global, 0, sizeof(fs20_global));
4    uint8_t time = 0;
5    for (;;) {
6      NutEventWait(&event_int_ac, NUT_WAIT_INFINITE);
7      time = TCNT0;
8      TCNT0 = 0; /* Bei jeder Flankenänderung Timer0 wieder auf 0 setzen */
9      if (!(ACSR & (1 << ACO))) {
10         if ((time >= 9) & (time <= 14)) { /* null empfangen */
11             fs20_global.buffer[fs20_global.buff_index] = 0;
12             fs20_global.buff_index++;
13             if (fs20_global.count_zero >= 0) {
14                 fs20_global.count_zero++;
15             }
16         } else if ((time > 14) & (time <= 22)) { /* eins empfangen */
17             fs20_global.buffer[fs20_global.buff_index] = 1;
18             fs20_global.buff_index++;
19             if (fs20_global.count_zero > 0) {
20                 fs20_global.count_zero = 0;
21             }
22         }
23         /* 10 Nullen hintereinander empfangen -> fs20 Synchronisation */
24         if (fs20_global.count_zero == 10) {
25             fs20_global.buff_index = 0;
26             memset(&fs20_global.buffer, 0, 256);
27             fs20_global.pFs20_cmd = &fs20_global.buffer[0];
28             fs20_global.count_zero = -1; /* Zaehler sperren */
29             NutIrqEnable(&sig_OVERFLOW0);
30         }
31     }
32     NutIrqEnable(&sig_COMPARATOR);
33 }
34 }

```

Um die Zeit zwischen zwei Interrupts des Analog Comparators zu messen nutze ich den 8-Bit Timer0 des Atmega128. Auf dem Ethernut 1.3 wird dieser mit 32,768kHz (Y2) betrieben. Die Zeit für einen Zählschritt ergibt sich so zu ungefähr:

$$\frac{1}{32,768 \cdot 10^6 Hz} \approx 31\mu s$$

Das bedeutet, dass für die Dauer eines 400 $\mu$ s Pulses (logische 0) der Timer ungefähr 12 Zählschritte ausführt. Für die Dauer eines 600 $\mu$ s Pulses (logische 1) führt der Timer ungefähr 19 Zählschritte aus. Daran erkennt der Thread ac\_int ob eine logische 0 oder eine logische 1 empfangen wurde. Wird 10x hintereinander eine logische 0 empfangen, wird davon ausgegangen dass die Übertragung eines FS20 Befehls begonnen wurde (FS20 Synchronisation). Daraufhin wird der Überlauf Interrupt des Timer0 aktiviert. D.h. wird dann für die Dauer von ca. 8ms (255 x 31 $\mu$ s) kein Interrupt vom Analog Comparator ausgelöst kommt es zu einem Überlauf des Timer0. Der Überlauf Interrupt von Timer0 wird von dem Thread TIMER0\_OVF bearbeitet wird. Der Thread überprüft (Paritybit, Prüfsumme) die empfangenen Daten und verarbeitet sie weiter.

## 6 Steuerung über den embedded Webserver

### 6.1 Integration in das Beispielprojekt HTTP Daemon (httpd)

Zur Steuerung über einen Internet-Browser nutze ich das Beispielprojekt von Nut/OS httpd. Es ermöglicht uns beim Aufruf einer bestimmten Internetseite eine bestimmte Funktion auszuführen. Zum Registrieren solcher Seiten nutzt man die Funktion `NutRegisterCgi()`. Hier ein Ausschnitt der `main` Funktion:

```
1 int main(void) {
2     ...
3     NutRegisterCgi("fht.eih", fht);
4     NutRegisterCgi("fht.xml", fhtxml);
5     NutRegisterCgi("fht.cgi", fhtcgi);
6     NutRegisterCgi("fhtprofil.cgi", fhtprofilcgi);
7     NutRegisterCgi("fs20.eih", fs20);
8     fs20_fht_init();
9     ...
10 }
```

Beim Aufruf der Seite `fht.xml` wird die Funktion `fhtxml` ausgeführt. Diese bearbeitet die Anfrage und sendet Informationen zurück an den Internet-Browser.

### 6.2 Steuerung der FHT80b Regler

#### 6.2.1 Status der Regler abfragen

Der Aufruf von `http://192.168.2.10/cgi-bin/fht.xml` ohne zusätzliche Parameter liefert eine XML-Datei mit allen angemeldeten FHT80b Regler und deren Hauscode.

Beispiel:

```
1 <?xml version="1.0" ?>
2 <FHT80b>
3   <fht hc1="2b" hc2="44"/>
4   <fht hc1="45" hc2="55"/>
5   <fht hc1="5b" hc2="54"/>
6 </FHT80b>
```

Der Aufruf von `http://192.168.2.10/cgi-bin/fht.xml?hc1=2b&hc2=44` liefert einem dann die aktuellen Werte aller relevanten Register. Die Register sind in Hexadezimaler Schreibweise angegeben, die dazugehörigen Werte in Dezimaler Schreibweise.

```
1 <?xml version="1.0" ?>
2 <FHT80b>
3   <fht hc1="2b" hc2="44"/>
4   <wochenprofil>
5     <fht register="14" value="33"/>
6     <fht register="15" value="66"/>
7     <fht register="16" value="144"/>
8     <fht register="17" value="143"/>
9     <fht register="18" value="33"/>
10    <fht register="19" value="66"/>
11    <fht register="1a" value="144"/>
12    <fht register="1b" value="143"/>
13    <fht register="1c" value="33"/>
14    <fht register="1d" value="66"/>
15    <fht register="1e" value="144"/>
16    <fht register="1f" value="143"/>
17    <fht register="20" value="33"/>
18    <fht register="21" value="66"/>
19    <fht register="22" value="144"/>
20    <fht register="23" value="143"/>
21    <fht register="24" value="33"/>
22    <fht register="25" value="66"/>
23    <fht register="26" value="144"/>
24    <fht register="27" value="143"/>
25    <fht register="28" value="33"/>
26    <fht register="29" value="66"/>
27    <fht register="2a" value="144"/>
28    <fht register="2b" value="143"/>
29    <fht register="2c" value="33"/>
30    <fht register="2d" value="66"/>
31    <fht register="2e" value="132"/>
32    <fht register="2f" value="143"/>
33  </wochenprofil>
34  <status>
35    <fht register="3e" value="0"/>
36    <fht register="41" value="40"/>
37    <fht register="42" value="203"/>
38    <fht register="82" value="46"/>
39    <fht register="84" value="40"/>
40    <fht register="8a" value="0"/>
41  </status>
42 </FHT80b>
```

Über den Aufruf von <http://192.168.2.10/cgi-bin/fht.cgi?hc1=2b&hc2=44> gelangt man zu einer kleinen Internetseite, über die sich die wichtigsten Einstellungen eines Regler einsehen und verändern lassen.

Heizung	<a href="#">Wochenprofil</a>	
Ist-Temp	22,3°C	
Soll-Temp	23,0°C	23.0 ▾
Modus	Auto	Auto ▾
Nacht	20,0°C	20.0 ▾
Tag	23,0°C	23.0 ▾
<a href="#">Aktualisieren</a>	<input type="button" value="Absenden"/>	

Ventile									
0	1	2	3	4	5	6	7	8	
93 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %

(a)

Wochenprofil	von	bis	von	bis
Mo	05:30 ▾	11:00 ▾	▾	23:50 ▾
Di	05:30 ▾	11:00 ▾	▾	23:50 ▾
Mi	05:30 ▾	11:00 ▾	▾	23:50 ▾
Do	05:30 ▾	11:00 ▾	▾	23:50 ▾
Fr	05:30 ▾	11:00 ▾	▾	23:50 ▾
Sa	05:30 ▾	11:00 ▾	▾	23:50 ▾
So	05:30 ▾	11:00 ▾	22:00 ▾	23:50 ▾
<a href="#">Zurueck</a>	<input type="button" value="Absenden"/>			

(b)

Abb. 8: Screenshot: Einstellungen eines FHT80b Reglers verändern

### 6.2.2 Werte eines Reglers verändern

Um einzelne Werte von Registern zu verändern ruft man

```
http://192.168.2.10/cgi-bin/fht.eih?hc1=2b&hc2=44&Register1=Wert1&Register2=Wert2
```

auf. Hier ist wieder darauf zu achten, die Register in Hexadezimaler Schreibweise anzugeben, die neuen Werte in Dezimaler Schreibweise.

Der Aufruf von

```
http://192.168.2.10/cgi-bin/fht.eih?hc1=2b&hc2=44&41=40
```

ändert z.B. die aktuelle Solltemperatur auf 20 °C. Die neuen Werte werden bei der nächsten Gelegenheit an die FHT80b Regler übertragen.

### 6.3 FS20 Steuerbefehle senden

FS20 Steuerbefehle können ganz einfach über den Aufruf folgender Seite gesendet werden:

```
http://192.168.2.10/cgi-bin/fs20.eih?hc1=18&hc2=4b&adr=0&cmd=11
```

Mit diesem Befehl kann z.B. eine Funksteckdose eingeschaltet werden. Die Werte für hc1, hc2, adr und cmd sind in hexadezimaler Schreibweise anzugeben!

## 6.4 FS20 Steuerbefehle empfangen

Um auf empfangene FS20 Steuerbefehle zu reagieren, muss der entsprechende Befehl zuerst registriert werden.

Dies geschieht mit der Funktion `fs20_add_sensor(hc1, hc2, adr, cmd, ext, void(*ptr)(void))` in der Funktion `fs20_init_sensors()`.

```
1 void fs20_init_sensors(void) {
2     fs20_add_sensor(0x4b, 0x18, 0x00, 0x12, 0x00, Makro1);
3 }
```

Makro1 ist in diesem Fall der Zeiger zu der Funktion die ausgeführt werden soll, wenn der entsprechende Befehl empfangen wurde.

Makro1 könnte z.B. so aussehen:

```
1 void Makro1(void) {
2     NutDelay(255); /* 255ms warten */
3     fs20_send_cmd(0x18, 0x4b, 0x00, 0x11, 0x00);
4     NutDelay(255); /* 255ms warten */
5     fs20_send_cmd(0x18, 0x4b, 0x01, 0x11, 0x00);
6     NutDelay(255); /* 255ms warten */
7     fs20_send_cmd(0x18, 0x4b, 0x02, 0x11, 0x00);
8     NutDelay(255); /* 255ms warten */
9     fs20_send_cmd(0x18, 0x4b, 0x03, 0x11, 0x00);
10 }
```

In diesem Fall werden also vier FS20 Steuerbefehle mit einem Abstand von je 255ms der Reihe nach gesendet, wenn ein bestimmter Steuerbefehl empfangen wurde.

## 7 Protokollbeispiele

### 7.1 Anmeldung der Zentrale an einem FHT80b Regler

Der FHT80b Regler übermittelt dann beim nächsten Sendezyklus seinen kompletten Status.  
Zentralencode = 0x32

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x00
cmd	0x20
ext	0x00

FHZ	
hc1	0x2b
hc2	0x44
adr	0x53
cmd	0x77
ext	0x32

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x53
cmd	0x67
ext	0x32

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x54
cmd	0x67
ext	0x32

FHZ	
hc1	0x2b
hc2	0x44
adr	0x7d
cmd	0x77
ext	0x32

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x7d
cmd	0x67
ext	0x32

FHZ	
hc1	0x2b
hc2	0x44
adr	0x65
cmd	0x79
ext	0xff

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x65
cmd	0x69
ext	0xff

FHZ	
hc1	0x2b
hc2	0x44
adr	0x66
cmd	0x79
ext	0xff

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x66
cmd	0x69
ext	0xff

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0xff

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0xff

FHZ	
hc1	0x2b
hc2	0x44
adr	0x7e
cmd	0x77
ext	0xff

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x7e
cmd	0x67
ext	0xff

## 7.2 FHT80b Regler übermittelt kompletten Status

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x0
cmd	0x20
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x7d
cmd	0x67
ext	0x32

FHZ	
hc1	0x2b
hc2	0x44
adr	0x7d
cmd	0x77
ext	0x32

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x14
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x14
cmd	0x79
ext	0x27

FHT80B	
hc1	0x2b
hc2	0x44
adr	0x15
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x15
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x16
cmd	0x69
ext	0x90

FHZ	
hc1	0x2b
hc2	0x44
adr	0x16
cmd	0x79
ext	0x90

FHT80B	
hc1	0x2b
hc2	0x44
adr	0x17
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x17
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x18
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x18
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x19
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x19
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1a
cmd	0x69
ext	0x90

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1a
cmd	0x79
ext	0x90

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1b
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1b
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1c
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1c
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1d
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1d
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1e
cmd	0x69
ext	0x90

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1e
cmd	0x79
ext	0x90

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x1f
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x1f
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x20
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x20
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x21
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x21
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x22
cmd	0x69

FHZ	
hc1	0x2b
hc2	0x44
adr	0x22
cmd	0x79
ext	0x90

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x23
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x23
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x24
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x24
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x25
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x25
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x26
cmd	0x69
ext	0x78

FHZ	
hc1	0x2b
hc2	0x44
adr	0x26
cmd	0x79
ext	0x78

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x27
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x27
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x28
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x28
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x29
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x29
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2a
cmd	0x69
ext	0x90

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2a
cmd	0x79
ext	0x90

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2b
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2b
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2c
cmd	0x69
ext	0x27

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2c
cmd	0x79
ext	0x27

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2d
cmd	0x69
ext	0x3c

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2d
cmd	0x79
ext	0x3c

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2e
cmd	0x69
ext	0x90

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2e
cmd	0x79
ext	0x90

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x2f
cmd	0x69
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x2f
cmd	0x79
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x8f

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x8f

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x82
cmd	0x69
ext	0x2e

FHZ	
hc1	0x2b
hc2	0x44
adr	0x82
cmd	0x79
ext	0x2e

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x84
cmd	0x69
ext	0x22

FHZ	
hc1	0x2b
hc2	0x44
adr	0x84
cmd	0x79
ext	0x22

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x8a
cmd	0x69
ext	0x18

FHZ	
hc1	0x2b
hc2	0x44
adr	0x8a
cmd	0x79
ext	0x18

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x18

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x18

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x85
cmd	0x69
ext	0x4

FHZ	
hc1	0x2b
hc2	0x44
adr	0x85
cmd	0x79
ext	0x4

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x4

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x4

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x41
cmd	0x69
ext	0x2e

FHZ	
hc1	0x2b
hc2	0x44
adr	0x41
cmd	0x79
ext	0x2e

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x2e

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x2e

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x42
cmd	0x69
ext	0xfa

FHZ	
hc1	0x2b
hc2	0x44
adr	0x42
cmd	0x79
ext	0xfa

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x43
cmd	0x69
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x43
cmd	0x79
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x3e
cmd	0x69
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x3e
cmd	0x79
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x44
cmd	0x69
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x44
cmd	0x79
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x67
ext	0x0

FHZ	
hc1	0x2b
hc2	0x44
adr	0x4b
cmd	0x77
ext	0x0

FHT80b	
hc1	0x2b
hc2	0x44
adr	0x7e
cmd	0x67
ext	0x0