`libsnap`

Scaleable Node Address Protocol

Stefan Böhmann
`<sboehm@technik-emden.de>`

Fachhochschule Emden/Leer

19. August 2010

DRAFT

Except where otherwise noted, this work is licensed under the Creative Commons *Attribution-Noncommercial-Share Alike* 3.0 License

S.N.A.P

Protokollspezifikation der schwedischen Firma HTC.
Entwickelt für die Produktpalette der Hausautomatisierungssysteme rund um das Power Line Modem `PLM-24`.

S.N.A.P Features

- Easy to learn, use and implement
- Free and open network protocol
- Scaleable binary protocol with small overhead
- Up to 16.7 million node addresses
- Up to 24 protocol specific flags
- Optional ACK/NAK request
- Optional command mode
- 8 different error detecting methods
- Media independent (power line, RF, TP, IR etc.)
- Works with simplex, half-, full- duplex links
- Header is scaleable from 3-12 bytes
- User specified number of preamble bytes (0-n)
- ...

S.N.A.P Features

- Easy to learn, use and implement
- Free and open network protocol
- Scaleable binary protocol with small overhead
- Up to 16.7 million node addresses
- Up to 24 protocol specific flags
- Optional ACK/NAK request
- Optional command mode
- 8 different error detecting methods
- Media independent (power line, RF, TP, IR etc.)
- Works with simplex, half-, full- duplex links
- Header is scaleable from 3-12 bytes
- User specified number of preamble bytes (0-n)
- ...

S.N.A.P Features

- Easy to learn, use and implement
- Free and open network protocol
- Scaleable binary protocol with small overhead
- Up to 16.7 million node addresses
- Up to 24 protocol specific flags
- Optional ACK/NAK request
- Optional command mode
- 8 different error detecting methods
- Media independent (power line, RF, TP, IR etc.)
- Works with simplex, half-, full- duplex links
- Header is scaleable from 3-12 bytes
- User specified number of preamble bytes (0-n)
- ...

S.N.A.P Features

- Easy to learn, use and implement
- Free and open network protocol
- Scaleable binary protocol with small overhead
- Up to 16.7 million node addresses
- Up to 24 protocol specific flags
- Optional ACK/NAK request
- Optional command mode
- 8 different error detecting methods
- Media independent (power line, RF, TP, IR etc.)
- Works with simplex, half-, full- duplex links
- Header is scaleable from 3-12 bytes
- User specified number of preamble bytes (0-n)
- ...

S.N.A.P Features

- Easy to learn, use and implement
- Free and open network protocol
- Scaleable binary protocol with small overhead
- Up to 16.7 million node addresses
- Up to 24 protocol specific flags
- Optional ACK/NAK request
- Optional command mode
- 8 different error detecting methods
- Media independent (power line, RF, TP, IR etc.)
- Works with simplex, half-, full- duplex links
- Header is scaleable from 3-12 bytes
- User specified number of preamble bytes (0-n)
- ...

libsnap

- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*

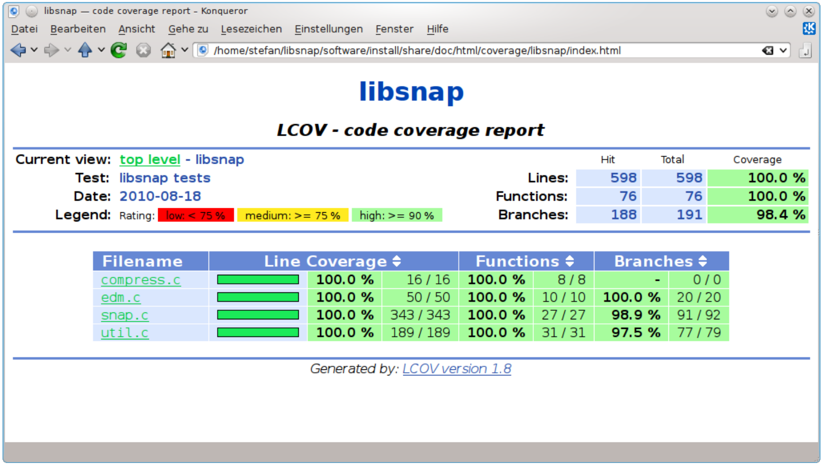
- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*
- Hardware- und Plattformunabhngig
  - geschrieben in C *(C99, ISO/IEC 9899:1999)*
  - Buildsystem basiert auf CMake

- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*
- Hardware- und Plattformunabhngig
  - geschrieben in C *(C99, ISO/IEC 9899:1999)*
  - Buildsystem basiert auf CMake
- kleine Codebasis *($\approx$ 3600 LOC)*

- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*
- Hardware- und Plattformunabhngig
  - geschrieben in C *(C99, ISO/IEC 9899:1999)*
  - Buildsystem basiert auf CMake
- kleine Codebasis *(≈ 3600 LOC)*
- einheitliche, saubere API

- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*
- Hardware- und Plattformunabhngig
  - geschrieben in C *(C99, ISO/IEC 9899:1999)*
  - Buildsystem basiert auf CMake
- kleine Codebasis *(≈ 3600 LOC)*
- einheitliche, saubere API
- keine dynamische Speicherallokierung

- freie Implementierung
  - GNU Lesser General Public License *(LGPL)*
- Hardware- und Plattformunabhngig
  - geschrieben in C *(C99, ISO/IEC 9899:1999)*
  - Buildsystem basiert auf CMake
- kleine Codebasis *($\approx$ 3600 LOC)*
- einheitliche, saubere API
- keine dynamische Speicherallokierung
- testgetriebenen Entwicklung
  - basiert auf dem CUnit Testing Framework
  - CTest - integriert im Build-Prozess
  - 60% Code sind Tests *($\approx$ 2200 LOC)*
  - gcov/lcov basierter Coverage Report
  - dadurch $\approx$ 100% Testabdeckung


libsnap — code coverage report - Konqueror
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Einstellungen Fenster Hilfe
/home/stefan/libsnap/software/install/share/doc/html/coverage/libsnap/index.html

libsnap

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level - libsnap				
Test:	libsnap tests	Lines:	598	598	100.0 %
Date:	2010-08-18	Functions:	76	76	100.0 %
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %	Branches:	188	191	98.4 %

Filename	Line Coverage ⬍			Functions ⬍		Branches ⬍	
compress.c	🟩	100.0 %	16 / 16	100.0 %	8 / 8	-	0 / 0
edm.c	🟩	100.0 %	50 / 50	100.0 %	10 / 10	100.0 %	20 / 20
snap.c	🟩	100.0 %	343 / 343	100.0 %	27 / 27	98.9 %	91 / 92
util.c	🟩	100.0 %	189 / 189	100.0 %	31 / 31	97.5 %	77 / 79

Generated by: LCOV version 1.8

```
bool snap_encode(
    snap_t *ptr,
    const uint8_t *in_data,
    size_t in_data_size,
    uint8_t *out_data,
    size_t *out_data_pos,
    size_t out_data_size
);
```

```
size_t snap_encode_bound(
    snap_t *ptr,
    size_t in_data_size
);
```

```
bool snap_decode(
    snap_t *ptr,
    const uint8_t *in_data,
    size_t *in_data_pos,
    size_t in_data_size,
    uint8_t *result,
    size_t *result_pos,
    size_t result_size,
    uint8_t *response,
    size_t *response_pos,
    size_t response_size
);
```

```
size_t snap_decode_result_bound(
    snap_t *ptr,
    size_t in_data_size
);

size_t snap_decode_response_bound(
    snap_t *ptr,
    size_t in_data_size
);
```

```
#include <libsnap/snap.h>

int main(int argc, char **argv)
{
    snap_t snap;
    snap_init( &snap );

    char in[] = "If a packet hits a pocket on a socket on a port, "
        "and the bus is interrupted and the interrupt's not caught, "
        "then the socket packet pocket has an error to report.";

    size_t in_size = strlen( in );

    size_t out_size = snap_encode_bound( &snap, in_size );
    uint8_t *out = malloc( out_size );
    size_t out_pos = 0;

    snap_encode( &snap, (uint8_t*) in, in_size,
                 out, &out_pos, out_size );
}
```

```
void snap_init(snap_t *ptr);
void snap_reset(snap_t *ptr);

size_t snap_get_error_detection_mode_size(snap_t *ptr);
SnapErrorDetectionMode snap_get_error_detection_mode(snap_t *ptr);
void snap_set_error_detection_mode(snap_t *ptr, SnapErrorDetectionMode edm);

size_t snap_get_local_address_size(snap_t *ptr);
int32_t snap_get_local_address(snap_t *ptr);
void snap_set_local_address(snap_t *ptr, int32_t address);

size_t snap_get_peer_address_size(snap_t *ptr);
int32_t snap_get_peer_address(snap_t *ptr);
void snap_set_peer_address(snap_t *ptr, int32_t address);

//...
```

- Das `EDM`-Flag kann unbemerkt kippen

- Das `EDM`-Flag kann unbemerkt kippen
- Vorwärtsfehlerkorrektur `FEC` wird erwähnt, aber nicht spezifiziert

- Das `EDM`-Flag kann unbemerkt kippen
- Vorwärtsfehlerkorrektur `FEC` wird erwähnt, aber nicht spezifiziert
- ein C-String wird ohne `NULL`-Terminierung dekodiert!

- Das `EDM`-Flag kann unbemerkt kippen
- Vorwärtsfehlerkorrektur `FEC` wird erwähnt, aber nicht spezifiziert
- ein C-String wird ohne `NULL`-Terminierung dekodiert!
- `encode` kann über 100% Overhead erzeugen

`libsnap++`

- libsnap für C++ Programmierer

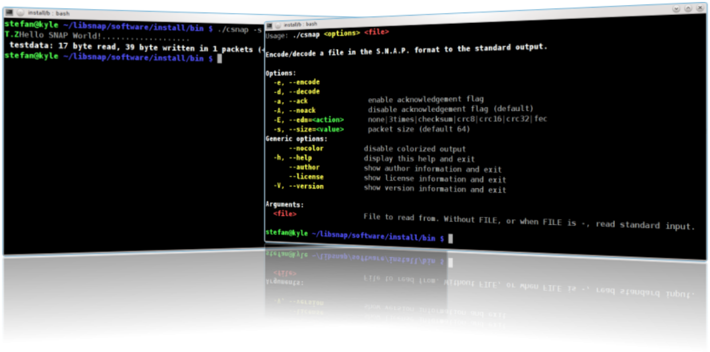
- libsnap für C++ Programmierer
- sehr kleiner Wrapper *(≈ 260 LOC)*

- libsnap für C++ Programmierer
- sehr kleiner Wrapper *(≈ 260 LOC)*
- objektorientierte Schnittstelle

```
Snap *snap = new Snap();
snap->setErrorDetectionMode( EDM_CRC32 );
snap->setAcknowledgementEnabled( true );

std::vector<uint8_t> data = readDataSomewhere();
std::vector<uint8_t> result = snap->encode( data );

//...
```

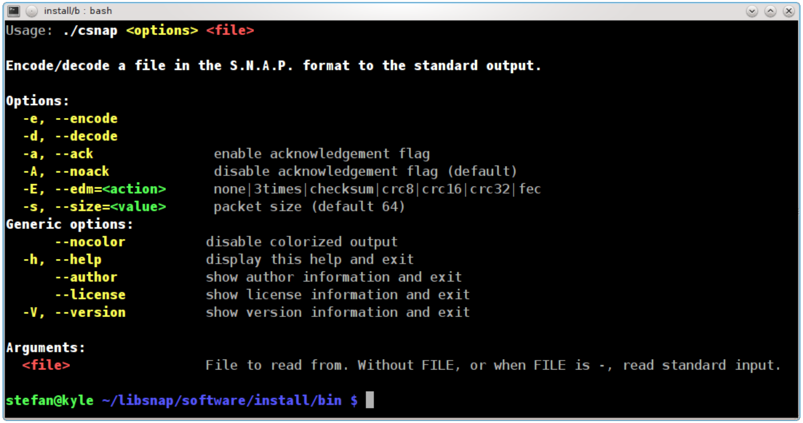
`csnap`

CLI, Programmiersprache C, 650 LOC



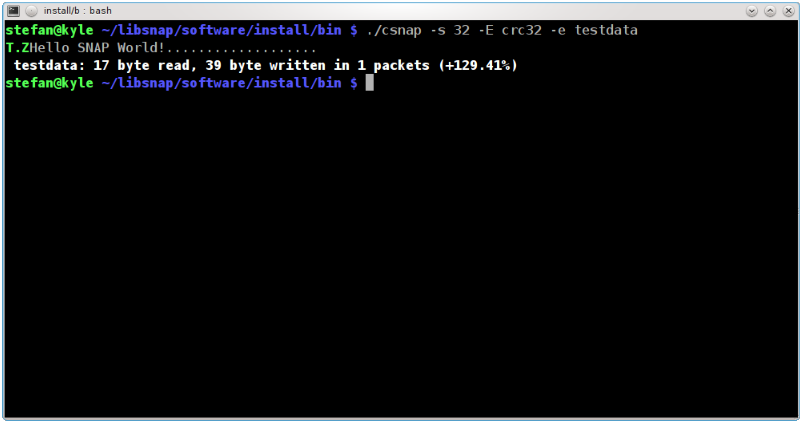
```
stefan@kyle ~/libsnap/software/install/bin $ ./csnap -s
T.ZHello SNAP World!.......................
 testdata: 17 byte read, 39 byte written in 1 packets (
stefan@kyle ~/libsnap/software/install/bin $ 
```

```
Usage: ./csnap <options> <file>

Encode/decode a file in the S.N.A.P. format to the standard output.

Options:
  -e, --encode
  -d, --decode
  -a, --ack               enable acknowledgement flag
  -A, --noack             disable acknowledgement flag (default)
  -E, --edm=<action>      none|3times|checksum|crc8|crc16|crc32|fec
  -s, --size=<value>      packet size (default 64)
Generic options:
      --nocolor          disable colorized output
  -h, --help             display this help and exit
      --author           show author information and exit
      --license          show license information and exit
  -V, --version          show version information and exit

Arguments:
  <file>
                         File to read from. Without FILE, or when FILE is -, read standard input.

stefan@kyle ~/libsnap/software/install/bin $ 
```

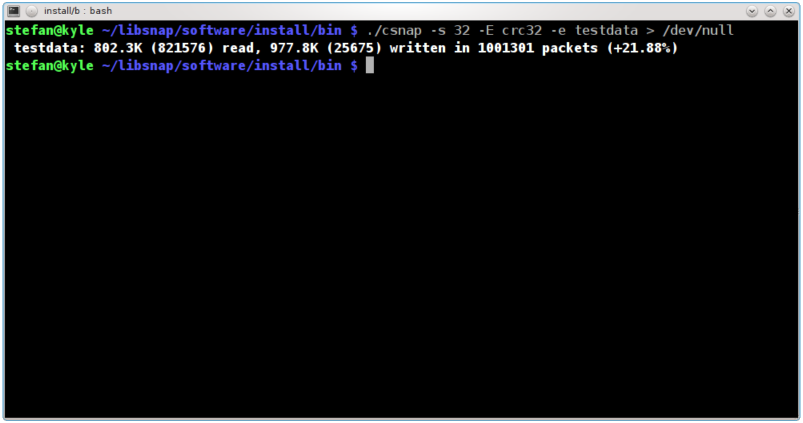


install/b : bash

```
Usage: ./csnap <options> <file>

Encode/decode a file in the S.N.A.P. format to the standard output.

Options:
  -e, --encode
  -d, --decode
  -a, --ack                 enable acknowledgement flag
  -A, --noack               disable acknowledgement flag (default)
  -E, --edm=<action>        none|3times|checksum|crc8|crc16|crc32|fec
  -s, --size=<value>        packet size (default 64)
Generic options:
      --nocolor            disable colorized output
  -h, --help               display this help and exit
      --author             show author information and exit
      --license            show license information and exit
  -V, --version            show version information and exit

Arguments:
  <file>                   File to read from. Without FILE, or when FILE is -, read standard input.

stefan@kyle ~/libsnap/software/install/bin $
```



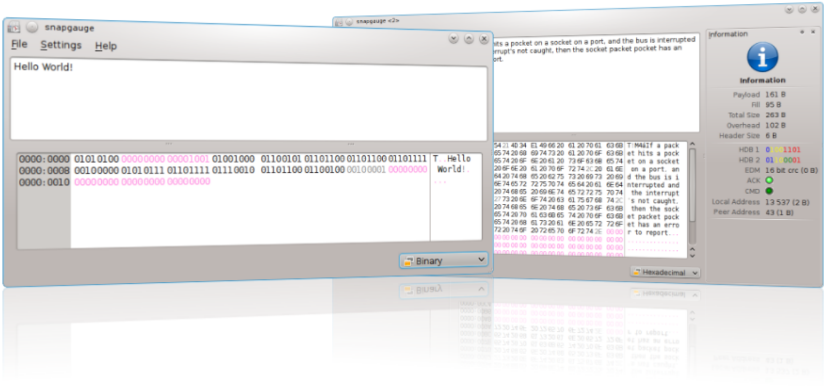
```
stefan@kyle ~/libsnap/software/install/bin $ ./csnap -s 32 -E crc32 -e testdata
T.ZHello SNAP World!..................
 testdata: 17 byte read, 39 byte written in 1 packets (+129.41%)
stefan@kyle ~/libsnap/software/install/bin $
```



```
stefan@kyle ~/libsnap/software/install/bin $ ./csnap -s 32 -E crc32 -e testdata > /dev/null
 testdata: 802.3K (821576) read, 977.8K (25675) written in 1001301 packets (+21.88%)
stefan@kyle ~/libsnap/software/install/bin $ 
```

`snapgauge`

GUI, Programmiersprache C++, 1500 LOC



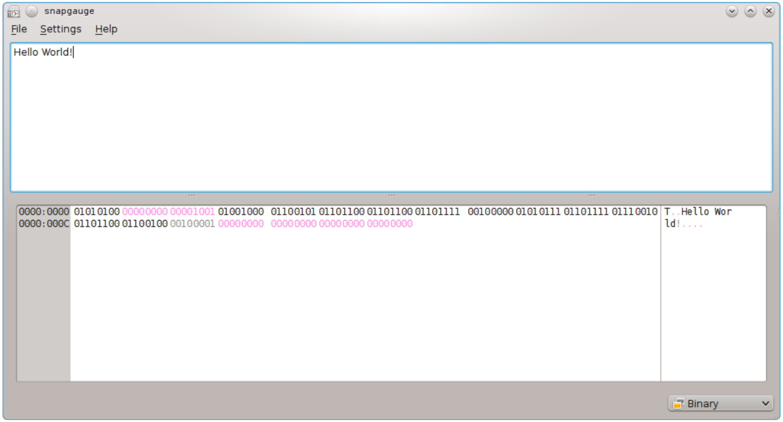
Hello World!

```
0000:0000 01010100 00000000 00001001 01001000 01100101 01101100 01101100 01101111 T..Hello
0000:0008 00100000 01010111 01101111 01110010 01101100 01100100 00100001 00000000  World!.
0000:0010 00000000 00000000 00000000                                              ...
```

its a pocket on a socket on a port, and the bus is interrupted
errupt's not caught, then the socket packet pocket has an
ort.

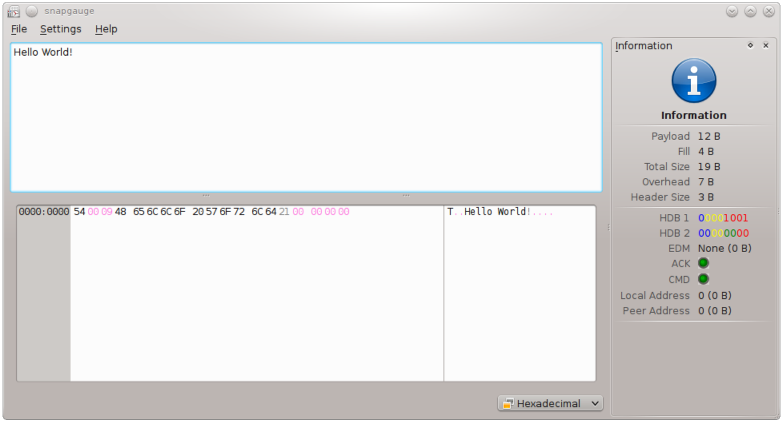
```
54 21 4D 34  E1 49 66 20  61 20 70 61  63 6B  T!M4áIf a pack
65 74 20 68  69 74 73 20  61 20 70 6F  63 6B  et hits a pock
65 74 20 6F  6E 20 61 20  73 6F 63 6B  65 74  et on a socket
20 6F 6E 20  61 20 70 6F  72 74 2C 20  61 6E   on a port. an
64 20 74 68  65 20 62 75  73 20 69 73  20 69  d the bus is i
6E 74 65 72  72 75 70 74  65 64 20 61  6E 64  nterrupted and
20 74 68 65  20 69 6E 74  65 72 72 75  70 74   the interrupt
27 73 20 6E  6F 74 20 63  61 75 67 68  74 2C  's not caught.
20 74 68 65  6E 20 74 68  65 20 73 6F  63 6B   then the sock
65 74 20 70  61 63 6B 65  74 20 70 6F  63 6B  et packet pock
65 74 20 68  61 73 20 61  6E 20 65 72  72 6F  et has an erro
72 20 74 6F  20 72 65 70  6F 72 74 2E  00 00  r to report...
00 00 00 00  00 00 00 00  00 00 00 00  00 00  ..............
00 00 00 00  00 00 00 00  00 00 00 00  00 00  ..............
```

Payload	161 B
Fill	95 B
Total Size	263 B
Overhead	102 B
Header Size	6 B
HDB 1	01001101
HDB 2	01100001
EDM	16 bit crc (0 B)
ACK	●
CMD	●
Local Address	13 537 (2 B)
Peer Address	43 (1 B)



Hello World!

```
0000:0000 01010100 00000000 00001001 01001000  01100101 01101100 01101100 01101111  00100000 01010111 01101111 01110010 T..Hello Wor
0000:000C 01101100 01100100 00100001 00000000  00000000 00000000 00000000                                               ld!....
```

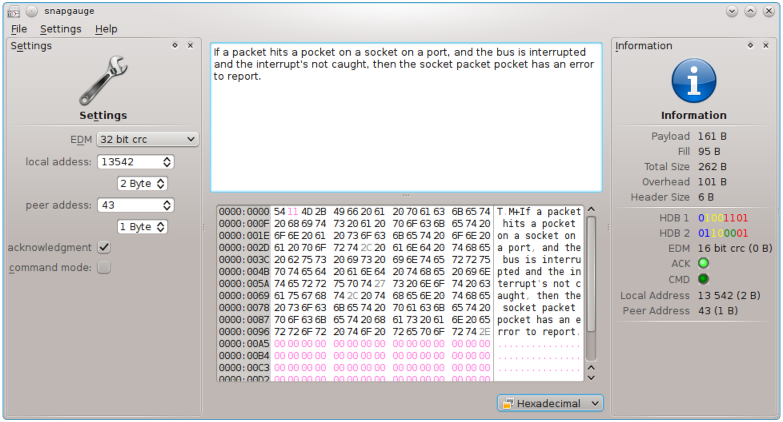


Hello World!

```
0000:0000 54 00 09 48  65 6C 6C 6F  20 57 6F 72  6C 64 21 00  00 00 00    T..Hello World!....
```

Information

Payload	12 B
Fill	4 B
Total Size	19 B
Overhead	7 B
Header Size	3 B
HDB 1	00001001
HDB 2	00000000
EDM	None (0 B)
ACK	●
CMD	●
Local Address	0 (0 B)
Peer Address	0 (0 B)



If a packet hits a pocket on a socket on a port, and the bus is interrupted and the interrupt's not caught, then the socket packet pocket has an error to report.

EDM	32 bit crc
local address:	13542
	2 Byte
peer address:	43
	1 Byte
acknowledgment	✓
command mode:	

```
0000:0000 54 11 4D 2B  49 66 20 61  20 70 61 63  6B 65 74 T.M+If a packet
0000:000F 20 68 69 74  73 20 61 20  70 6F 63 6B  65 74 20  hits a pocket 
0000:001E 6F 6E 20 61  20 73 6F 63  6B 65 74 20  6F 6E 20 on a socket on 
0000:002D 61 20 70 6F  72 74 2C 20  61 6E 64 20  74 68 65 a port, and the
0000:003C 20 62 75 73  20 69 73 20  69 6E 74 65  72 72 75  bus is interru
0000:004B 70 74 65 64  20 61 6E 64  20 74 68 65  20 69 6E pted and the in
0000:005A 74 65 72 72  75 70 74 27  73 20 6E 6F  74 20 63 terrupt's not c
0000:0069 61 75 67 68  74 2C 20 74  68 65 6E 20  74 68 65 aught, then the
0000:0078 20 73 6F 63  6B 65 74 20  70 61 63 6B  65 74 20  socket packet 
0000:0087 70 6F 63 6B  65 74 20 68  61 73 20 61  6E 20 65 pocket has an e
0000:0096 72 72 6F 72  20 74 6F 20  72 65 70 6F  72 74 2E rror to report.
0000:00A5 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 ...............
0000:00B4 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 ...............
0000:00C3 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 ...............
0000:00D2 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00
```

Payload	161 B
Fill	95 B
Total Size	262 B
Overhead	101 B
Header Size	6 B
HDB 1	01001101
HDB 2	01100001
EDM	16 bit crc (0 B)
ACK	●
CMD	●
Local Address	13 542 (2 B)
Peer Address	43 (1 B)

Fragen?

Thank you!