S.N.A.P

Scaleable Node Address Protocol

Preface.

Thanks for your interest in the **S.N.A.P** network protocol. We wanted to define a simple and generic network protocol that could be used in many different types of microcontroller applications as well for educational purposes. We are not implying that **S.N.A.P** will solve all network problems in this universe because it won't! There may be other protocols that suit your specific application better. Never the less we think **S.N.A.P** is a great protocol for it's intended use and therefore we decided to share it with the rest of the community.

Our priority has been to keep it as simple as possible and our internal mantra has been KISS (Keep It Simple Stupid). It's harder than one can imagine balancing technical advanced functions with simplicity and we have had many ideas that we scraped due to the reason that it would increase the learning curve and make it harder for beginners to use. The persons involved in the development of **S.N.A.P**, all have many years background as teachers and know how important it is to keep things simple for easy understanding.

One drawback with defining such a flexible network protocol is that it makes it almost impossible to define everything since many parameters (such as timing etc.) depends on what kind of media being used. We could have defined **S.N.A.P** to be used with our **PLM-24** Power Line Modems only and with strict timing information but then it wouldn't be so versatile as we wanted it.

This document version specifies the **S.N.A.P** packet framing format. Consider **S.N.A.P** as an ongoing project that we will continue to work on and add more functionality to over time. We hope you find our work useful and welcome you to e-mail comments, suggestions and ideas. Due to the amount of e-mails received we are not always able to reply to you personally but we do read all e-mails.

Note that the purpose of this **S.N.A.P** documentation is not to be a complete book with answers on every question that may arise (we may eventually do something like that in the future). However, we have done our best in given time to provide enough information to get you started and the best way to learn how **S.N.A.P** works in reality is to start experiment with it and study the source code examples that is available on our web-site.

In short, if you find **S.N.A.P** useful then feel free to use it and all we ask is that you give credit where credit is due and if **S.N.A.P** doesn't suit your specific needs then you should consider using another network protocol.

Good luck with your project!

Features.

- Easy to learn, use and implement.
- Free and open network protocol.
- Scaleable binary protocol with small overhead.
- Requires minimal MCU resources to implement.
- 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 (Checksum, CRC, FEC etc.).
- Can be used in master/slave and/or peer-to-peer.
- Supports broadcast messages.
- Media independent (power line, RF, TP, IR etc.).
- Works with simplex, half-, full- duplex links.
- Header is scaleable from 3-12 bytes.
- Minimum packet size without error detection is 3 bytes.
- Minimum packet size with error detection is 4 bytes.
- User specified number of preamble bytes (0-n).
- Works with synchronous and asynchronous communication.
- Works with our free PLM-24 <-> TCP/IP Gateway software.

1.0 Introduction.

Why yet another protocol? Because we at HTH needed a protocol for our **PLM-24** based home automation system. We wanted a protocol that could easily be implemented in small microcontrollers with very limited computing and memory resources. We also wanted to be able to use the same protocol in larger systems. The solution was to make the protocol scaleable.

S.N.A.P allows for different packet length and different protocol complexity. It can be used as a very simple protocol without any flags or error detection, or the programmer can use up to 24 flags and any of the defined error detection methods, depending on the current need or personal skill. Since **S.N.A.P** is scaleable, both simple (read as cheap) and sophisticated nodes can communicate with each other in the network.

That is what makes **S.N.A.P** unique!

1. 1 What can S.N.A.P be used for?

We developed **S.N.A.P** primary for use in home automation systems but it is a generic protocol and not limited to this. **S.N.A.P** can be used in any type of applications where an easy to learn and flexible network protocol is needed.

1.2 How small MCU's can be used?

S.N.A.P can be implemented in almost any MCU available today. Our first **S.N.A.P** testprogram ran on a BASIC Stamp I from Parallax Inc. It sent 1 Byte data and used 16-bit CRC as error detection method. This tiny microcontroller has only 14 Bytes of available RAM! As another node in the network we used a standard Pentium PC and the nodes were communicating over the mains using **PLM-24** Power Line Modems.

1.3 Is S.N.A.P easy to learn?

As mentioned before it can be used as a very simple protocol and is therefore easy to learn. So it's great for educational purposes or for electronic hobbyists. And probably the professional will appreciate it, since it's easy to implement and a very flexible protocol.

1.4 Using S.N.A.P in commercial applications.

S.N.A.P is free for private and commercial (requires a vendor ID#) use. All that we ask in return is that you give credit where credit is due and that you enclose the original PDF-file or a link to our web-site with your applications/products.

If you intend to use **S.N.A.P** in any commercial application you **must** request a vendor ID#. This will not cost anything. For details on how to request your own vendor ID# see our web-site at...

http://www.hth.com/snap/

1.5 Support.

S.N.A.P is released "AS IS" and we are not able to provide any free support.

1.6 Future development of S.N.A.P.

We reserve us the right to change, modify or enhance the **S.N.A.P** network protocol without any prior notice. Our intention is to enhance **S.N.A.P** with more features in the future. To stay up-to-date with the development of **S.N.A.P** feel free to join our mailing list, details at the end of this document. We welcome any feedback and suggestions from users. We are aware about the missing information such as collision detection an recommended time-out values. This will eventually be included in a future revision of this document.

2.0 Protocol description.

Below is the structure of **S.N.A.P** described. Before we begin a few words of explanation. All communication between network nodes, is in the form of packets. These packets can be of different length. The total packet length will depend on how many address bytes (0-6 Bytes) you decide to use, how many flags bytes (0-3 Bytes), how much data you want to send (0-512 Bytes) and what error detection method you use. This is defined in the header definition bytes (**HDB2** and **HDB1**).

Each packet is preceded with optional preamble bytes (0-n). The packet starts with a unique byte, this byte is called the synchronization byte. Any type of preamble characters can be used as long as they are not the same as the synchronization byte.

In the example below you see a small packet with the following structure.

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB1	Destination Address Byte
SAB1	Source Address Byte
DB1	Data Byte 1
CRC2	High byte of CRC-16
CRC1	Low byte of CRC-16

n bytes preamble	**SYNC**	**HDB2**	**HDB1**	**DAB1**	**SAB1**	**DB1**	**CRC2**	**CRC1**

The total length of this packet would be 8 Bytes (excluding the optional preamble bytes). All bytes within a group are positioned with the least significant byte to the right.

2.1 Synchronization byte - SYNC.

This byte is pre-defined to 01010100_2 and indicates the start of each packet.

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

2.2 Overview of header definition bytes (HDB2 and HDB1).

The first two bytes after the synchronization byte are called header definition bytes, they are used to define the structure of the complete packet. The name on the fields are chosen to be easy to remember.

HDB2

Bit	7	6	5	4	3	2	1	0
	D	D	S	S	P	P	A	A

HDB1

Bit	7	6	5	4	3	2	1	0
	C	E	E	E	N	N	N	N

D	**S**	**P**	**A**	**C**	**E**	**N**
A	**A**	**F**	**C**	**M**	**D**	**D**
B	**B**	**B**	**K**	**D**	**M**	**B**

DAB = Number of **D**estination **A**ddress **B**ytes
SAB = Number of **S**ource **A**ddress **B**ytes
PFB = Number of **P**rotocol specific **F**lag **B**ytes
ACK = **ACK**/NAK bits
CMD = **C**o**M**man**D** mode bit
EDM = **E**rror **D**etection **M**ethod
NDB = **N**umber of **D**ata **B**ytes

2.3 Header Definition Byte 2 - HDB2.

Bit	7	6	5	4	3	2	1	0
	D	D	S	S	P	P	A	A

Bit 7 and 6 - Destination Address Bytes (DAB)

These two bits defines number of destination address bytes in the packet. With the maximum size of 3 Bytes it gives a total of 16.7 millions different destination node addresses.

Bit 7	6	
0	0	0 Byte destination address
0	1	1 Byte destination address
1	0	2 Bytes destination address
1	1	3 Bytes destination address

Bit 5 and 4 - Source Address Bytes (SAB)

These two bits defines the number of source address bytes in the packet. With the maximum size of 3 Bytes, it gives a total of 16.7 millions different source node addresses.

Bit 5	4	
0	0	0 Byte source address
0	1	1 Byte source address
1	0	2 Bytes source address
1	1	3 Bytes source address

Bit 3 and 2 - Protocol specific Flag Bytes (PFB)

These two bits defines how many protocol specific flag bytes the packet includes, from 0-3 Bytes which give a total of 24 flags.

Bit 3	2	
0	0	0 Byte flags
0	1	1 Byte flags
1	0	2 Bytes flags
1	1	3 Bytes flags

Bit 1 and 0 - ACK/NACK Bits (ACK)

These two bits defines if the sending node requests an ACK/NAK packet in return. These bits also acts as the actual ACK/NAK response sent from the receiving node.

Bit 1	0	
0	0	No ACK request (Tx)
0	1	ACK request (Tx)
1	0	ACK response (Rx)
1	1	NAK response (Rx)

2.4 Header Definition Byte 1 - HDB1.

Bit	7	6	5	4	3	2	1	0
	C	E	E	E	N	N	N	N

Bit 7 - Command mode bit

This bit indicates what's called command mode. This is an optional feature and if a node is not implementing it this bit should always be set to zero (CMD=0).

A node implementing this feature will be able to respond on queries from other nodes as well as send responses when for example the receiving node can't handle the packet structure in a received packet. It can be used to scan large networks for nodes and have them respond with their capabilities or for two nodes negotiating the right packet structure, among other things.

If this bit is set (CMD=1) it indicates that the data in DB1 contains a command (query or a response). This results in total 256 different commands.

The range is divided in two half's, commands between 1-127 are queries and commands between 128-255 are responses. The commands specified to date are the following. Note this is the value in DB1, not the actual CMD bit.

CMD	Query	CMD	Response
0	Command mode supported?	128	Command mode supported
1	Preferred packet structure?	129	Preferred packet structure
2	Reserved but not yet defined	130	Reserved but not yet defined
...	...	...	...
127	Reserved but not yet defined	255	Reserved but not yet defined

There are some things to think about for this to work properly. The sending node can not use an higher address range than the receiving node. This is not a problem if the receiving nodes that are implementing this feature are capable to handle all the address range (i.e. 1-16.7 millions). Another solution is to assign all masters in the network (in a master/slave network) to the low address range (i.e. between 1-255).

Bit 6 to 4 - Error Detection Method (EDM)

These three bits defines, what kind error detecting method is being used to validate the packet. A node does not need to support any error detection method at all (i.e. EDM = 0) or you can choose to implement any or all of them.

Bit 6	5	4	
0	0	0	No error detection
0	0	1	3 times re-transmission
0	1	0	8-bit checksum
0	1	1	8-bit CRC-CCITT
1	0	0	16-bit CRC-CCITT
1	0	1	32-bit CRC-CCITT
1	1	0	FEC (specific FEC standard TBD)
1	1	1	User specified

TBD = To Be Determined

Bit 3 to 0 - Number of Data Bytes (NDB)

These four bits defines how many bytes data there is in the packet (0-512 Bytes).

Bit 3	2	1	0	
0	0	0	0	0 Byte
0	0	0	1	1 Byte
0	0	1	0	2 Bytes
0	0	1	1	3 Bytes
0	1	0	0	4 Bytes
0	1	0	1	5 Bytes
0	1	1	0	6 Bytes
0	1	1	1	7 Bytes
1	0	0	0	8 Bytes
1	0	0	1	16 Bytes
1	0	1	0	32 Bytes
1	0	1	1	64 Bytes
1	1	0	0	128 Bytes
1	1	0	1	256 Bytes
1	1	1	0	512 Bytes
1	1	1	1	User specified

2.5 Protocol specific Flag Bytes (PFB3-PFB1).

These three flag bytes are **reserved** but not yet defined. They are intended for future enhancements of **S.N.A.P**. They will be defined in a future protocol version and should **not be used**!

Below are some ideas for their use and we welcome any further comments and suggestions.

- Remote reset
- Remote re-programming
- Remote configuration
- Media traverse flag
- Data encryption flag
- Extended command mode
- Routing information flag
- Repeater flag
- Packet counter
- Time sync flag
- Packet numbering
- Packet priority levels

Bit 7 to 0 - PFB3

Bit	7	6	5	4	3	2	1	0
	R	E	S	E	R	V	E	D

Eight protocol specific flags. Specific functions TBD.

Bit 7 to 0 - PFB2

Bit	7	6	5	4	3	2	1	0
	R	E	S	E	R	V	E	D

Eight protocol specific flags. Specific functions TBD.

Bit 7 to 0 - PFB1

Bit	7	6	5	4	3	2	1	0
	R	E	S	E	R	V	E	D

Eight protocol specific flags. Specific functions TBD.

2.6 Miscellaneous

- Address 0 is reserved as a broadcast address and should not be used for anything else.

- All packets must start with a synchronization byte (01010100_2).

- The synchronization byte is not included in the error detection calculation.

- If a node is capable to use 2 or 3 Bytes DAB it should also be capable to decode 1 or 2 Bytes DAB for compatibility.

- If you are sending packets over media's like power line, RF or IR keep the packets short (less than 40 Bytes) to improve performance.

3.0 Examples.

Below are some additional examples on how **S.N.A.P** packets may look like. In the examples the optional preamble bytes isn't shown. More examples on bit level can be found in appendix A.

Example 1

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB1	Destination Address Byte
SAB1	Source Address Byte
DB1	Data Byte 1
CHK	8-bit checksum

SYNC	HDB2	HDB1	DAB1	SAB1	DB1	CHK

Example 2

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB2	Destination Address Byte 2
DAB1	Destination Address Byte 1
SAB2	Source Address Byte 2
SAB1	Source Address Byte 1
DB2	Data Byte 2
DB1	Data Byte 1
CRC2	High byte of CRC-16
CRC1	Low byte of CRC-16

SYNC	HDB2	HDB1	DAB2	DAB1	SAB2	SAB1	DB2	DB1	CRC2	CRC1

Example 3

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB1	Destination Address Byte
SAB1	Source Address Byte
DB1	Data Byte 1
CRC4	High byte of most significant word in CRC-32
CRC3	Low byte of most significant word in CRC-32
CRC2	High byte of least significant word in CRC-32
CRC1	Low byte of least significant word in CRC-32

SYNC	HDB2	HDB1	DAB1	SAB1	DB2	DB1	CRC4	CRC3	CRC2	CRC1

Example 4

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB1	Destination Address Byte
SAB1	Source Address Byte
DB8	Data Byte 8
DB7	Data Byte 7
DB6	Data Byte 6
DB5	Data Byte 5
DB4	Data Byte 4
DB3	Data Byte 3
DB2	Data Byte 2
DB1	Data Byte 1
CRC	CRC-8

SYNC	HDB2	HDB1	DAB1	SAB1	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	CRC

Example 5

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB3	Destination Address Byte 3
DAB2	Destination Address Byte 2
DAB1	Destination Address Byte 1
SAB3	Source Address Byte 3
SAB2	Source Address Byte 2
SAB1	Source Address Byte 1
PFB1	Protocol specific Flag Byte
DB1	Data Byte 1
CRC2	High byte of CRC-16
CRC1	Low byte of CRC-16

SYNC	HDB2	HDB1	DAB3	DAB2	DAB1	SAB3	SAB2	SAB1	PFB1	DB1	CRC2	CRC1

4.0 FAQ - Frequently Asked Questions.

Below is an FAQ that gives you answers to common questions.

Coming in future document revision...

5.0 S.N.A.P network protocol history

980909 - **S.N.A.P** Version 0.91 [Preliminary Draft].

First public release.

990120 - **S.N.A.P** Version 1.00 [Initial release].

Changes/additions to previous protocol version:

ACK/NAK structure changed.
Removed user specific flag bytes.
Added user specified number of databytes.
Command mode implemented.

6.0 PLM-News mailing list.

If you want to stay up-to-date with **S.N.A.P** you can subscribe to our PLM-News mailing list. We will announce updates, new products, tips & tricks when available.

To subscribe to the PLM-News mailing list send an e-mail to...

listserv@hth.com

with the following in the body...

subscribe plm-news <e-mail address>

Substitute "<e-mail address>" with your own e-mail address.

After you subscribed you will receive a confirmation via e-mail and there after receive every issue of PLM-News.

6.1 More information.

More information about **S.N.A.P** and the **PLM-24** Power Line Modem can be found at the URL's given below. If you are interested to implement **S.N.A.P** in any commercial application please send e-mail for free licensee registration.

info@hth.com

Information about **S.N.A.P** and **PLM-24** Power Line Modem can be found at...

http://www.hth.com/snap/
http://www.hth.com/plm-24/

6.2 Feedback and suggestions.

If you have any comments or suggestions or have any program examples that are using **S.N.A.P** and want to share them with others please feel free to contact us by e-mail. We are looking forward to hear from different people using **S.N.A.P** and if you send us a short story about your use we will publish it on our web.

rz@hth.com

6.3 Special thanks.

Special thanks to the following people for ideas and comments.

Claus Kühnel
Mats Ekberg

Disclaimer of Liability.

Overview of header definition bytes (HDB2 and HDB1)

	HDB2									HDB1							
Bit	7	6	5	4	3	2	1	0	Bit	7	6	5	4	3	2	1	0
	D	D	S	S	P	P	A	A		C	E	E	E	N	N	N	N

See section 2.0 in the manual for a detailed description of each bit position in HDB2 and HDB1. The following examples shows several packets on a bit level, to make it easier for the **S.N.A.P** beginner.

Example 1 - Packet size 8 Bytes

In the example below the transmitting node has address 00000001_2 and is sending data 11111111_2 to node 00000010_2. Since no acknowledge is required the transmitting node will not expect any ACK or NAK packet in return.

Packet structure.

DD=01	- 1 Byte destination address
SS=01	- 1 Byte source address
PP=00	- No protocol specific flags
AA=00	- No ACK request
C=0	- Command mode not supported
EEE=100	- 16-bit CRC-CCITT
NNNN=0001	- 1 Byte data

Packet sent from node 00000001_2

SYNC	HDB2	HDB1	DAB1	SAB1	DB1	CRC2	CRC1
0 1 0 1 0 1 0 0	0 1 0 1 0 0 0 0	0 1 0 0 0 0 0 1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1	1 1 1 1 1 1 1 1	0 1 0 0 1 1 1 0	1 0 1 1 1 0 1 1

Example 2 - Packet size 8 Bytes

In the example below the transmitting node has address 00000001_2 and is sending data 11110000_2 to node 00000011_2. This time acknowledge is requested and the transmitting node expect to receive an ACK or NAK packet, else it should time-out and take proper action.

Packet structure.

DD=01	- 1 Byte destination address
SS=01	- 1 Byte source address
PP=00	- No protocol specific flags
AA=01	- ACK request
C=0	- Command mode not supported
EEE=100	- 16-bit CRC-CCITT
NNNN=0001	- 1 Byte data

Packet sent from node 00000001_2

SYNC	HDB2	HDB1	DAB1	SAB1	DB1	CRC2	CRC1
01010100	01010001	01000001	00000011	00000001	11110000	00100010	00110101

ACK packet returned from node 00000011_2

SYNC	HDB2	HDB1	DAB1	SAB1	DB1	CRC2	CRC1
01010100	010100**10**	01000001	**00000001**	**00000011**	**00000000**	**00101011**	**11111010**

NAK packet returned from node 00000011_2

SYNC	HDB2	HDB1	DAB1	SAB1	DB1	CRC2	CRC1
01010100	010100**11**	01000001	**00000001**	**00000011**	**00000000**	**10000001**	**10101011**

Note: Red color indicates changed bits/bytes.

Example 3 - Packet size 9 Bytes

In the example below the transmitting node has address 00000001_2 and is sending data 11110000_2 to node 00000011_2. This time to, acknowledge is requested and the transmitting node expect to receive an ACK or NAK packet, else it should time-out and take proper action. Further more one byte of protocol specific flags are used, these flags are set to 00000011_2 in PFB1. Also note that the ACK/NAK packet returned contains 0 Byte data.

Packet structure.

DD=01	- 1 Byte destination address
SS=01	- 1 Byte source address
PP=01	- 1 Byte protocol specific flags
AA=01	- Acknowledge is required
C=0	- Command mode not supported
EEE=100	- 16-bit CRC-CCITT
NNNN=0001	- 1 Byte data

Packet sent from node 00000001_2

SYNC	HDB2	HDB1	DAB1	SAB1	PFB1	DB1	CRC2	CRC1
01010100	01010101	01000001	00000011	00000001	00000011	11110000	10011110	00001100

ACK packet returned from node 00000011_2

SYNC	HDB2	HDB1	DAB1	SAB1	PFB1	CRC2	CRC1
01010100	01010110	01000000	00000001	00000011	00000011	11100100	00101011

NAK packet returned from node 00000011_2

SYNC	HDB2	HDB1	DAB1	SAB1	PFB1	CRC2	CRC1
01010100	01010111	01000000	00000001	00000011	00000011	01001110	01111010

Note: Red color indicates changed bits/bytes.

S.N.A.P examples.

Below is a list of examples currently available for download. The **S.N.A.P** examples includes schematics and well documented source code so they should give you an easy start. In some examples there are room for much improvement and possibility to add more functionality, we kept it as simple as possible for easy understanding.

WIP = Work In Progress

Name	MCU	Ver.	Description
SNAP-001	BS1-IC	1.02	Turn a LED on and off
SNAP-002	BS1-IC	1.02	Lampdimmer node for PLM-24
SNAP-003	BS1-IC	1.02	Domestic AC current meter with PLM-24
SNAP-004	BS1-IC	1.02	Simple temperature node for PLM-24
SNAP-005	BS1-IC	1.02	Simple light measuring node for PLM-24
SNAP-006	BS1-IC	1.02	Air quality node for PLM-24
SNAP-007	BS1-IC	1.02	Simple humidity node for PLM-24
SNAP-008	BS1-IC	1.02	Simple 4-bit input node for PLM-24
SNAP-009	BS1-IC	1.02	WakeUp alarm node for PLM-24
SNAP-010	BS1-IC	1.02	Four channel plant moisture sensor I
SNAP-011	BS2-IC	1.02	Turn a LED on and off
SNAP-012	BS2-IC	1.02	Shows how to implement background tasks
SNAP-013	BS2-IC	1.02	PLM-24 to X-10 Gateway
SNAP-014	BS2-IC	1.02	8-bit parallel input node for PLM-24
SNAP-015	BS2-IC	WIP	Programmable light monitor node for PLM-24
SNAP-016	89C2051	1.02	Turn a LED on and off
SNAP-017	BS2-IC	1.02	IR detector alarm node for PLM-24
SNAP-018	BS2-IC	1.02	Four channel relay node with local control
SNAP-019	BS2-IC	1.02	1-8 zones security system node for PLM-24
SNAP-020	BS2-IC	WIP	DCF-77 atomic clock node for PLM-24
SNAP-021	BS2-IC	1.02	Fire alarm node for PLM-24
SNAP-022	BS1-IC	1.02	1-channel 8-bit A/D converter node
SNAP-023	89C2051	1.02	Simple 16 x 1 LCD terminal node for PLM-24
SNAP-024	89C2051	1.02	Simple 16 x 1 LCD info node for PLM-24
SNAP-025	89C2051	1.02	S.N.A.P packet spy node for PLM-24
-	-	-	More to come...

High Tech Horizon
Asbogatan 29 C
S-262 51 Angelholm
SWEDEN

E-mail: info@hth.com
WWW: http://www.hth.com