AU - HERNING

SNAP

Small Networked Area Protocol

16-12-2011

This document describes the protocol called SNAP (Small Networked Area Protocol). As the name implies, it is a data protocol that can be used for communication between devices in a small networked area. It's first intended use is in the EnergyHub project for the E10 Class at AU-Herning.

Table of contents

About this document **2**

Version table **3**

Outline **4**

Recipient ID's 4

Protocol version and versioning rules 4

Message identifier 4

Example from PRO3 ..... *5*

Data field 5

Checksum 5

Acknowledge 5

Network Outline **6**

Master cycle 6

Pinging *6*

Request new data *6*

Check for new modules *6*

About this document

This document contains the protocol being designed for the PRO3 project for the E10 Class at AU-Herning. The next chapter contains the version table and changelog of the document, where all changes can be seen.

At this moment the protocol is still in a development stage, and the current stage of the protocol is the fundament that will be built upon in the coming semester.

Lars Juhl-Nielsen **Rune Inglev**

Version table

Date	Person	Action	Version
15-12-2011	Rune Inglev Lars Juhl-Nielsen	Creation of document. Took elements from first prototype and updated to a more generic protocol.	> Version 0.1
15-12-2011	Rune Inglev Lars Juhl-Nielsen	Added **Network Outline** after speaking to Morton Opprud. Also added that there must be reserved message identifiers and an extra reserved recipient ID.	> Version 0.2

Outline

The picture below is an outline of the message frame proposed for the data protocol for the PRO3 project. The frame has an SOF (Start of Frame) and an EOF (End of Frame) which begins and ends the message.

The next field is a 5 bit Recipient ID or Address ID used by modules on the databus to determine whether the message is meant for them. After this is a 3 bit Protocol version, which can be used by the receiving module, to determine, whether it has software is compatible with this format of the message. The next field is the 8 bit (1 byte) message identifier. This identifier describes the content of the data field. The data field is 4-8 bytes (still to be decided). The last field is the Checksum field, followed by an acknowledge bit.

SOF	Recipient ID (5 bits)	Protocol version (3 bits)	Msg identifier (1 byte)	Data field (4-8 bytes)	Checksum (? bytes)	Acknowledge (1 bit)	EOF

These fields are the first outline of the protocol. Development is still underway, and thus the strict format of the messages has not yet been decided. But this is a prototype, which lays the foundation for the structure.

Recipient ID's

The recipient ID field is 5 bits wide. This gives the opportunity for 16 different addresses (0-15). The Master Module is defined to have **ID #0** and the last ID is reserved for new modules to be attached (**ID #15**). **ID #1** is reserved as the **network broadcast ID**. Data being broadcast on this ID will result in all connected modules to receive the message. The acknowledgement of a **Network Broadcast** is still to be defined in details.

With these reserved addresses there is a possibility for the master to communicate with 12 slave modules (**IDs #2 - #14**).

Protocol version and versioning rules

The field is 3 bits wide serving the possibility of 8 different versions (0-7). All versions will be in the format of **Y.XX** – where the XX specifies a minor version and the Y specifies a major version. The current versions rules are that **any** major version, **independent** of the minor versions, should be compatible. This means that a module running version 3.4 and a module running 3.1 **must be able to communicate**.

The protocol version field will contain the **major version number**. This gives the possibility of 8 different major versions, before having to rewind and begin from the start again. The idea is that at the time of the release of version 8, version 1 will be deprecated, and the protocol version number (**#0**) is free to be used for version 9.

Message identifier

The message identifier is used to describe the content of the data field. Depending on the usage of the protocol, the identifier can be used to distinguish different messages. By using 8 bits it gives the possibility for any module on the network to send 256 different messages.

Message identifiers 0-15 are reserved for system specific messages that will be defined later. Some of them will be emergency messages, error/warning messages, ID Requests and ID assignment.

Example from PRO3

From the EnergyHub project the idea is that the Hub module routes the information on to tables in a database to be used on a website. By using the message identifier, the Hub can place the information in the specific fields of the database that corresponds to the data, without knowing explicitly, what the information actually is.

Data field

This field contains the payload of the message packet. The data field is a 4-8 byte field. At this stage of development the specific length hasn't been determined yet.

Checksum

At this stage of the development the checksum structure hasn't been determined yet. But it will most likely be the **Two's compliment** of the data in the data field.

The checksum will be **sent by the transmitter** and then **the receiver will compute its own checksum** from the data received and compare it to the checksum.

Acknowledge

If the **received checksum** and the **computed checksum** are equal the receiving module will acknowledge the received data by pulling the line low.

Network Outline

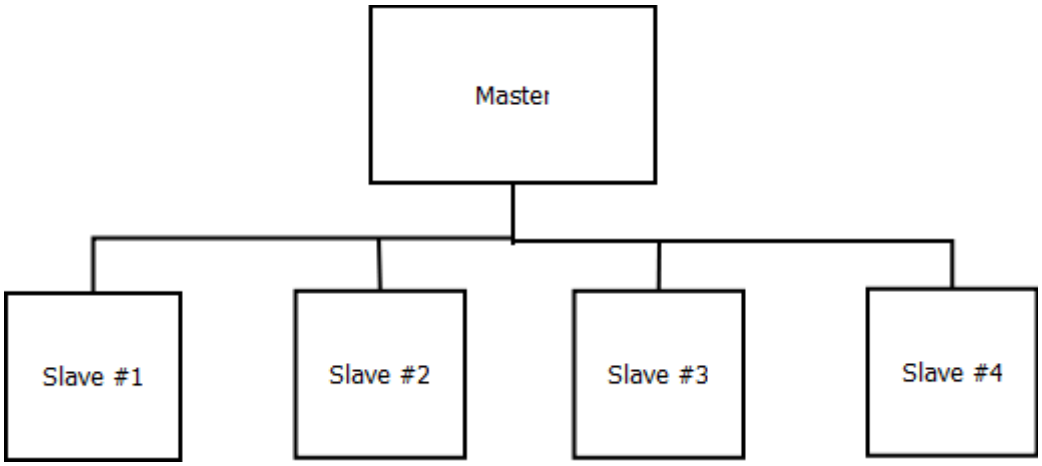
The protocol is used in small networks with a master and several (a maximum of 12 modules) slaves.

Master cycle

The master is setup to do an automatic cycle in which it performs the following tasks:

- Pings the modules
- Requests new data from modules
- Checks for new modules

The timing of these tasks is specified in registers in the master module.



Pinging

The master will ping all modules by broadcasting on the **Network Broadcast ID (ID #1)**. This way all attached modules will hear the message.

Request new data

Every now and then the master will start requesting data from each module. It will start with the lowest ID and go up. If no data is being received the master will retry a couple of times before moving and, and at the same time it will flag the module as unresponsive.

Check for new modules

After each cycle of requesting data, the master will send a special **Respond** message to **ID #15**. If there is a new module connected, the slave will respond with a Request ID message. The master will then send a new ID to the slave, which will then be used in the next cycle of data requests.