Communication Protocol Proposal for Daisy Chaining Force-Guiding Particles in Shape-Shifting Displays

Raoul Rubien, BSc
rubienr@sbox.tugraz.at

***Abstract*—Force-Guided Particles are used in Shape-Shifting Displays. A Shape-Shifting Display is made of a number of parallel free hanging particle chains. A particle chain consists of several chain links which we denote particles. They are connected by a special mechanical joint that incorporates a Shape Memory Alloy[1] (SMA) wire. This wires are used as actuators to unlock joints before an externally applied force can fold the affected segments of a chain, but they can also be used for communication of two adjacent particles. Actions performed by this modules need to be coordinated by a microcontroller in an efficient way. This proposal addresses the protocol design and decisions made for particles developed by Matteo Lasagni [1]. We will discuss the network structure, particle addressing, package routing, neighbor discovery, self enumeration, package details and as well the physical transmission layer.**

I. Introduction

To achieve shaping of Force-Guided Particle Chains, particles need to perform their actions synchronously at specific timestamps to unlock joints in between them and thus allow an externally applied force to fold the chain at this pivot/s. A two dimensional Shape-Shifting Display is achieved by processing multiple one dimensional chains in parallel. The underlying protocol requirements are *a*) the usage of particles as proposed by Matteo Lasagni in [1], *b*) exploitation of available SMA wires connecting particles to establish a communication channel, *c*) a predictable protocol timing behavior, *d*) synchronicity of executed actions and *e*) the particle network structure. The physical layer coding is chosen to be self synchronizing since network participants do not share a common clock wire. Thus particles have to derive the communication clock from the transmitted data itself. Physical, data and network layer also network topology, addressing and routing policies are stated with regard to a lightweight communication and narrow resources available on a low level microcontroller. For that reason also a simple data packing and un-packing is considered. The structure applied to the particle network is illustrated in the network topology figure 1. All chains are driven by one bus master (later referred as master) where the first particle of a chain (later referred as head particle) is connected to the next chain's head particle. Communication originating from the master is, depending on the protocol

[1] https://en.wikipedia.org/wiki/Shape-memory_alloy

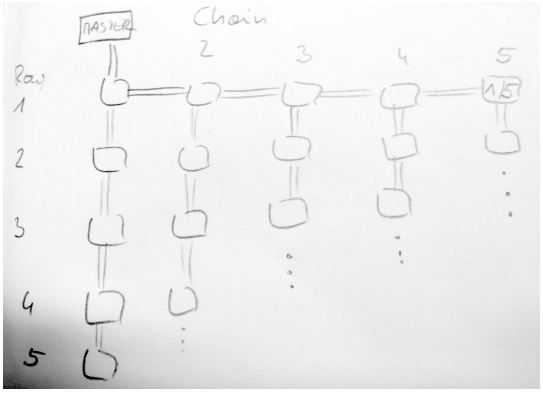


Figure 1: Network topology: Multiple parallel chains are connected at their head-particles. The first chain is also connected to the bus master.

state, routed or simultaneously repeated by the first node (later referenced as origin node) to the bottom and right. The directions, from node's perspective, we denote north, east, and south as shown in figure 2. Node addresses are derived from their row and column in matrix manner $row \times column$. As an example the origin node has address $(1, 1)$ as illustrated in the network topology figure 1.

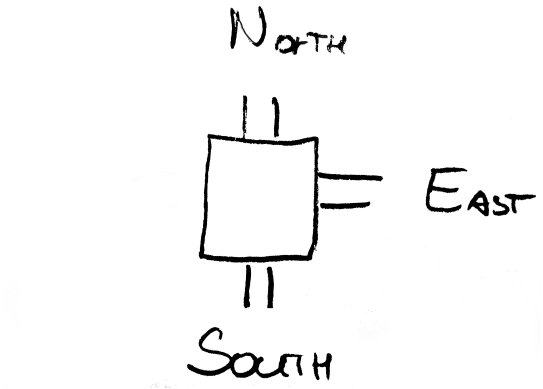


Figure 2: Particle directions: Directions are termed from the node's viewpoint as cardinal directions. North (top), east (right) and south (bottom).

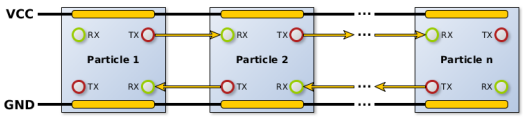


Figure 3: Simple one-dimensional particle chain: SMA actuators are used for one-way transmissions.

II. Motivation

Particles share two connected actuators that are used for unlocking joints. This actuators are SMA wires and remain unused after a short period of activation. The remaining time the SMA wires stay unused whereas they could be nicely used for communication. Using this wires as communication channel not only saves the need of additional wires it also cuts costs, assembly effort, weight and reduces one more error source since extra wires can also become defective.

III. Goals

The goals are to develop *a*) a stable but lightweight Physical Layer, Data link Layer and Network Layer that allows us to establish a bidirectional full duplex communication channel in a two dimensional particle network via SMA wires. Particles will be able to *b*) detect neighbors, *c*) self enumerate themselves and *d*) relay transmissions from or to the network master on their own. They also have to be able to *e*) receive one activator command and *f*) schedule it to be executed at a certain time. Other commands might be i.e. *synchronize time with neighbor* or *heat wire one at a specific time*. More details can be found in table I on the next page.

IV. Protocol Description

In this section we describe the particle protocol implementation and rules as close as possible in regard to the Open System Intercommunication model (OSI[2] model). In detail we address only the layers *(I)* - Physical Layer, *(II)* - Data link Layer and *(III)* - Network Layer since the very first three layers offer enough functionality to address network participants and communicate with them on a command basis. A host to host communication is not necessary in our use cases.

A. Package Structure

A package is the logic transmission from one network participant to an arbitrary other participant. A package can be considered as data transmission from one node to it's adjacent neighbor or to an arbitrary other node, which implies forwarding throughout other nodes. Commands are indicated either by the header (a built-in command) or the payload. Built-in commands come without payload or payload that fits semantically and format wise in the address field. Error correction nor error detection is considered to be implemented as well as parity bits. Hereafter we denote a package header as h, address as a, data length description as l and payload as d, stream as s whereas a package may be referenced as $p(d, l, a, h)$. A package starts with it's header field describing the package type that enables the parsing of the subsequent fields such as destination address or payload. The header is followed by an optional address field that describes one destination or a destination range in terms of a rectangle span beginning with the top left and ending at the bottom right node address. An address example is illustrated in figure 4 on the following page. The address field, in case of subsequent payload, field is followed by the payload length and the payload. Commands are meant for the addresses matching the address (or range) field a. Otherwise in case of a bit stream the address is derived from the bit position in the stream. As a consequence the bit stream must contain instructions for any node in the network. Furthermore for a given network dimension $M \times N$ we define entities of d, l, a, h to be expressed as

- payload data
  DATA[u:0] $:= d(u) \in C_{data}$
- stream data
  STRM[1:0] = DATA0[1:0]‖...‖ DATAv[1:0] :=
  $s(u) = u_0 \parallel ... \parallel u_v \mid \forall d_i \in C_{stream}, v = MN - 1$
- empty data length field $:= l()$
- data length field
  HDL[1:0] $:= l(|d|)$
- empty address field $:= a()$
- address field
  N0[7:0]‖M0[7:0] $:= a(m, n)$
- address range field
  N1[7:0]‖M1[7:0]‖N0[7:0]‖M0[7:0] :=
  $a([(m_0, n_0), (m_1, n_1)])$
- header type field
  HDR[3:0] $:= h(t_n)$

with C_s as the set of actuator related commands were all commands are $C = C_{stream} \cup C_{built\text{-}in} \cup C_{data}$ (see table IV on page 7) and $t_n \mid n \in header\_id$ (see table II on page 6). Thus the package bits of P[n:0] are defined as concatenation of d, a, h as follows:

$$p(d, l, a, h) = d(...) \parallel l(...) \parallel a(...) \parallel h(...) \tag{1}$$

A general package use case list can be found in table I on the next page.

B. Neighbor Discovery

TODO: reformulate
.TODO: reformulate
. TODO: reformulate
.TODO: reformulate

[2]https://en.wikipedia.org/wiki/OSI_model

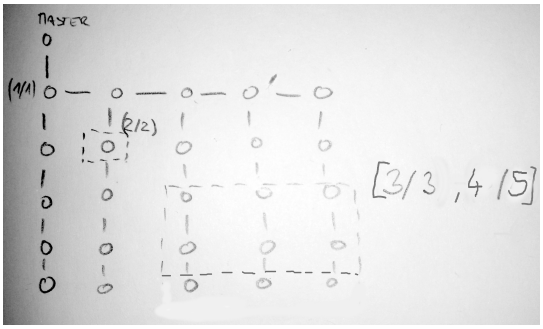


Figure 4: Network addressing: Nodes may be addressed directly as highlighted at address $(row = 2,\ col = 2)$ or within a rectangle span as highlighted from top left $(row = 3, col. = 3)$ to bottom right $(row = 4, col. = 5)$.

. Immediately after particles are powered on they switch to neighbor discovering mode to determine first details about their position in the particle network. According to the network topology decision made (figure 1 on page 1) particles will realize one of the possible chain positions $\{\underline{topmost}, tail, inter\text{-}node, tail, orphan\}$ as illustrated in figure 5 on the next page. Depending on the connectivity we distinguish in between head - the topmost particle of a chain, inter-node - a node that has an neighbor at the north and south, tail - the bottommost node of a chain and for the sake of completeness orphan - a node without any connection. In case of $\underline{topmost}$ we differentiate the sub types, again according to the head's connectivity, as $\{origin, inter\text{-}head, head\}$. If the network has only one chain we call the first node $origin_I$. As a result we can list entirely the node types: $\{origin, origin_I, inter\text{-}head, head, inter\text{-}node, tail, orphan\}$. Physically the discovery is done as follows: each node continuously pulses a signal on all it's transmission wires until they have discovered their own type plus a short safety duration. Receiving nodes count incoming falling edge events until they exceed a certain threshold. If this threshold is exceeded this side is assumed connected. The signal generation and edge counting is interrupt driven. Signals are generated by a counter-compare interrupt. To not lose compare-interrupts this counter is paused during processing other interrupts i.e. with received falling edge interrupt. As a consequence of that, nodes receiving relatively more signals will slow down the signal generation. Such nodes are inter nodes or inter heads. Their slowdown impact depends on the chain length, position in chain or horizontal inter head position. To overcome this issue the exact minimum value of neighbor events threshold and pulsing duration has to be determined in real life experiments using networks incorporating a reasonable number of participants. **TO BE DETERMINED during upcoming development.**

PACKAGE USE CASES

header id_{16}	is contemp.	sender	receiver	use case
1	1	$p(d(u), l(), a(m,n), h(1))$		
		master	node	master sends heat wire u to specific node (m,n))
2	1	$p(d(u), l(), a([(3,3),(4,4)]), h(2))$		
		master	node range	master sends heat wire u to range of nodes $\{(3,3),\ (4,3),\ (3,4),\ (4,4)\}$
3	1	$p(s(v), l(), a(), h(3))$		
		master	all nodes	master sends bit stream v to all nodes
4	1	$p(d(u), l(\|u\|), a(), h(4))$		
		master	origin	master sends sync time u to all nodes
5	1	$p(d(u), l(\|u\|), a(), h(5))$		
		master	all nodes	master sends sync offset u to all nodes
6	1	$p(d(), l(), a(m,n), h(6))$		
		master	all nodes	master discloses network geometry (m,n) to all nodes
7	1	$p(d(), a(m,n), h(7))$		
		node	master	node sends network geometry response (m,n) to master
8	1	$p(d(), l(), a(), h(8))$		
		master	origin	master sends network geometry request to origin
9	1	$p(d(u), l(\|u\|), a(m,n), h(9))$		
		node	master	node (m,n) sends status report u to master
A	1	$p(d(), l(), a(), h(0xA))$		
		master	all nodes	master sends verbose mode to all nodes
B	1	$p(d(), l(), a(), h(0xB))$		
		master	all nodes	master sends reset request to all nodes
C	1	$p(d(), l(), a(), h(0xC))$		
		node	master	node sends ping response to master
D	1	$p(d(), l(), a(m,n), h(0xD))$		
		master	node	master sends ping to node (m,n)
E	1	$p(d(u), l(\|u\|), a(), h(0xE))$		
		master	all nodes	master sends heating mode u to all nodes
F	0	$p(d(u), l(), a(), h(0xF))$		
		node	adjacent node	during enumeration: node sends next address to adjacent neighbor
F	1	$p(d(u), l(), a(), h(0xF))$		
		node	adjacent node	otherwise: represends a header larger than one byte

Table I: Package Description: General listing of package structures. Column is contemp. describes whether this package is reflected contemporarily or otherwise sent subsequently.

C. Self Enumeration

The next step following neighbor sensing is self enumeration. The enumeration defines any node's address before the network is ready for usage. It is triggered by only one node, that is the origin node. It defines it's own address as $(m = 1, n = 1)$ and sends the next address to the neighbors. For example an origin node having neighbors to the south and east, sends firstly the next address $(m + 1, n)$ to the south followed by $(m, n + 1)$ to the east. The neighbors themselves send the

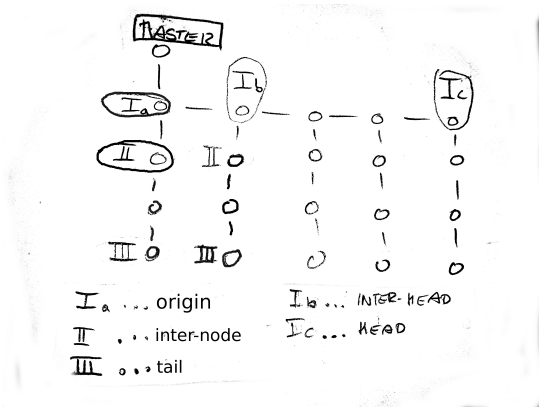


Figure 5: Particle types: Depending on their connectivity we distinguish in between $\{origin, inter\_head, head, inter\_node, tail, orphan\}$ - the topmost particles of chains, inter node - a node that has an neighbor at the north and south and tail - the bottommost node of a chain.

incremented address to their next but one nodes and so forth. That method will propagate the addressing diagonally from the top left to the bottom right corner as illustrated in figure 6 and will require $m+n-1$ steps in total. In the enumeration state nodes send packages sequentially firstly to the south followed by east.

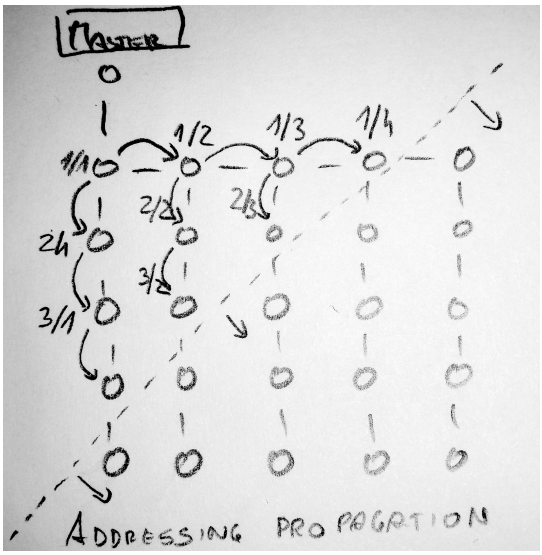


Figure 6: Addressing propagation: The origin node starts addressing. This request is spread diagonally throughout the network.

D. Package Use Cases

Transmission use cases are listed itemized by sender and receiver in table I on the previous page. Details are provided in the following sub sections. To speed up transfer and

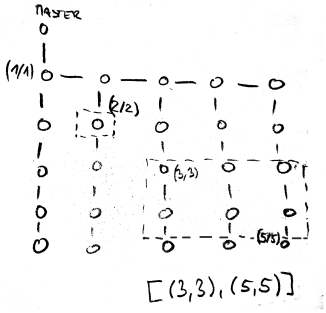


Figure 7: Master to particle range transmission: Master passes command to origin $(1,1)$ issued for nodes in range $[(3,3),(3,5)]$.

simplify the implementation we distinguish between pre- and post-enumeration. When participants are enumerated packages are not routed but simultaneously forwarded. More precise incoming signal transitions are reflected to the respective outgoing sides. In contrast routed packages must be fully received before they can be subsequently routed.

1) Master to particle communication: In the most common cases the network master issues commands to one specific particle. The command is then addressed to exactly one particle but instead of routing a node re-transmits the package contemporarily.

2) Master to range communication: In productive usage it is very likely to fold entire segments. They may not only occur within chains but also span over many chains. For that case one command may be issued to a rectangle shaped particle range defined by its top left and bottom right corner (i.e. figure 7). In that case data is transmitted contemporarily.

3) Master to all particles communication: In case of one command must be addressed to all network nodes, the transmission has to have set the header correctly and the optional address field is to be skipped. Such a package is propagated simultaneously.

4) Master to all particles communication - bit stream: Depending on the shape that is intended to be achieved the number of folded particles differ and so do the amount of transmitted commands. If a high number of commands is to be sent we offer an alternative way for data transmission. In contrast to contemporary forwarding particular commands we broadcast a bit stream. This has a longer but constant transmission to reception delay and is more effective if the number of nodes to be addressed exceed a certain limit.

a) Tradeoff considering data length: Given a network $N \in m \times n$, header h, address a, address range a' and a minimal payload d, the fields' sizes are: $|h| = 1Byte$ $|a| = 2Byte$ $|a'| = 4Byte$ $|d| = 2bit = \frac{1}{4}Byte$. Hence we define the master to node transmission package size:

$$|p| = |h| + |a| + |d| = (1 + 2 + \frac{1}{4})B = \frac{13}{4}B \qquad (2)$$

The master to node range transmission package size:

$$|p_r| = |h| + |a'| + |d| = (1 + 4 + \frac{1}{4})B = \frac{21}{4}B \quad (3)$$

The master to all nodes bit stream size:

$$|p_s| = |h| + |d|mn = (1 + \frac{1}{4}mn)B \quad (4)$$

Thus the limits k_p or k_r where the transmission byte size is equal to the stream size can be calculated as follows:

$$k_p \frac{13}{4} = \frac{5}{4}mn$$
$$k_p = \frac{5}{13}$$
$$k_r \frac{21}{4} = \frac{5}{4}mn$$
$$k_r = \frac{5}{21}mn$$

5) Node to neighbor communication: Nodes issue packages autonomously to their neighbors only during the self enumeration phase. This packages are received, interpreted and eventually forwarded. This is done subsequently.

6) Node to master communication: In case of errors or any arbitrary response addressed to the master must be packed with this header type. The protocol does not provide a collision avoidance for overlapping transmissions. In other words if two nodes issue packages to the master in an overlapping time window, the received result is undefined.

E. Physical Layer [3]

Particles are linked with two SMA wires a side as shown in figure 3 on page 2. We opt out to use a wire for one directional transmissions only. Thus we safely can provide full duplex communication in between two adjacent nodes buy using both wires in opposite directions. As there is no spare wire for a separate clock the modules need to derive the transmission clock from the transmitted data to decode the received bits. Thus we decided to apply the IEEE 802.3 Manchester code[4] as illustrated in figure 8. **TO BE DETERMINED if use manchester or differential manchester coding. I would stick to the much simpler method of capturing delay in between occuring flanks because of simplicity and the package can be interpreted online (not in retrospect).** [5]

F. Data link Layer [6]

Tis layer is kept very simple and does not support error detection or error correction. That allows us to implement this protocol in a manageable way. Another advantage is the possibility to keep transmitted packages small. We also do not expect dynamic network structure changes or temporarily unavailable nodes. For that reason the layer neither supports framing or retransmission.

[3]https://en.wikipedia.org/wiki/Physical_layer
[4]https://en.wikipedia.org/wiki/Manchester_code
[5]https://en.wikipedia.org/wiki/Differential_Manchester_encoding
[6]https://en.wikipedia.org/wiki/Data_link_layer

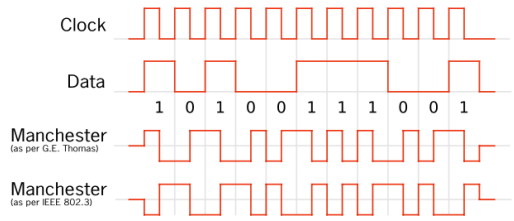


Figure 8: Manchester Coding: The code is a combination of clock and data. Original data can be XORed from the code ($data = clock \oplus code$).

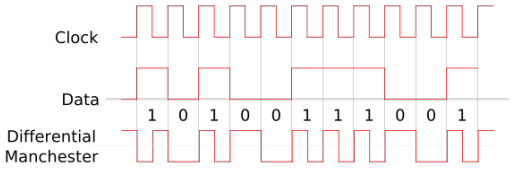
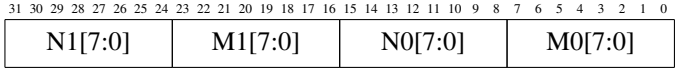


Figure 9: Differential Manchester Coding: The code is a combination of clock and data. Data is encoded in the middle of the clock period. At the beginning of any clock period there is a transition.

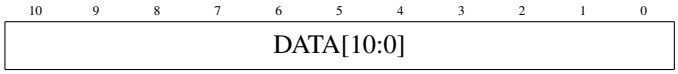
a) Header field: The header field HDR[3:0] carries details about the optional address and command size. Header ids that also describe commands and do not need payload are termed as protocol built-in commands. This packages may carry arguments in the optional address field but never payload.

b) Address field - direct addressing: The address field a addresses a node $(M0, N0) \in m \times n$ in the network:

c) Address field - range addressing: The address range field a addresses nodes: $[(M0, N0), (M1, N1)] \in m \times n \mid \{M0 < M1, N0 \leqslant N1\}$ or $\{M0 \leqslant M1, N0 < N1\}$.



d) Payload field: The optional payload field's d bit size may vary as listed in table III on the next page and is defined by the header in HDL[1:0]. For built-in commands both fields are skipped. Payload field by an exemplary 11 bit payload length:



HEADER TYPES

id 16	HDR3	HDR2	HDR1	HDR0	package direction/type
0	0	0	0	0	reserved
1	0	0	0	1	master → node: heat wires
2	0	0	1	0	master → node range: heat wires
3	0	0	1	1	master → all nodes: heat wires bit stream
4	0	1	0	0	master → origin: built-in sync time
5	0	1	0	1	master → all nodes: build-in sync offset
6	0	1	1	0	master → node: built-in disclose network geometry
7	0	1	1	1	node → master: build-in network geometry response
8	1	0	0	0	master → last node: built-in geometry request
9	1	0	0	1	node ↔ master
A	1	0	1	0	master → all nodes: built-in verbose mode
B	1	0	1	1	master → all nodes: built-in reset
C	1	1	0	0	node → master: built-in ping response
D	1	1	0	1	master → node: built-in ping node
E	1	1	1	0	master → node: built-in heating mode
F	1	1	1	1	node → neighbor node: while enumeration, otherwise extended header (reserved for future header types)

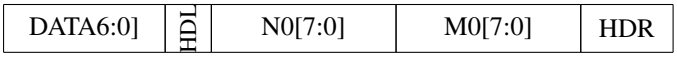
Table II: HDR[3:0]: Describing the header type; header id is base 16.

DATA LENGTH DESCRIPTION

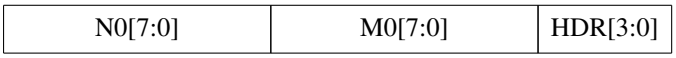
HDL1	HDL0	data field length $[bit]$
0	0	7
0	1	11
1	0	19
1	1	reserved

Table III: HDL[1:0]: Describing the payload field size.

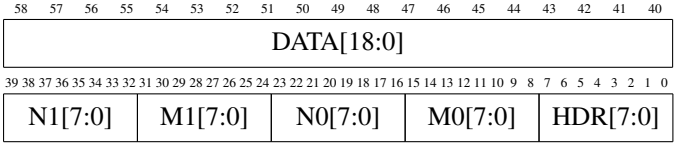
e) Master to single node package: A usual package addressed from master to a single node in the network by an exemplary 7 bit payload field:



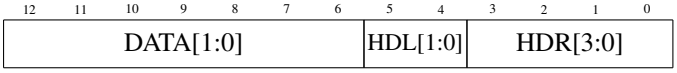
A built-in command package does not contain the payload field as the header type describes also the command but has an optional address field M0, N0:



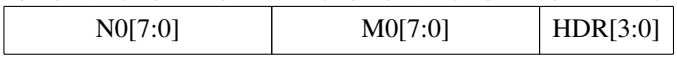
f) Master to node range package: A package addressed from master to a node range by an exemplary 19 bit payload field:



g) Master to all nodes package: A package addressed from master all network participants by an exemplary 7 bit payload field. The destination address is skipped:



A built-in command package does not contain the payload field as the header type describes the command but the address field M0, N0 is optional:

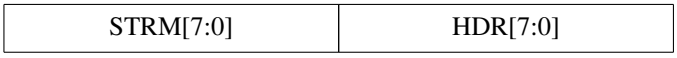


h) Master to all nodes bit stream: A bit stream is passed to all participants beginning at the origin issued by the master. The stream follows the header and provides a two bit payload per node. The tuple position $s \in \mathbb{N}_0 \mid s < |s|$ in the payload bit stream corresponds to the network address as explained in equation (5). The inverse mapping from node address to stream position interval can be read as equation (6). Due to the fact that these functions need the network dimension (M, N) the master need to send this information beforehand.

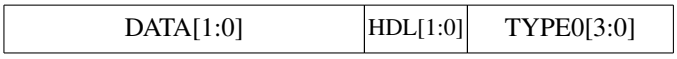
$$\begin{aligned} address(s) &\longmapsto (m, n) \mid m, n \in \mathbb{N}_+ \\ address(s) &= \begin{pmatrix} M - \lfloor \frac{s}{2} \mod M \rfloor \\ N - \lfloor \frac{s}{2(N+1)} \rfloor \end{pmatrix}^T \end{aligned} \tag{5}$$

$$\begin{aligned} pos(m, n) &\longmapsto p \in \mathbb{N}_0 \mid p < 2MN \\ s &\in [p, p+1] \\ pos(m, n) &= 2(MN - (m + M(n-1))) \end{aligned} \tag{6}$$

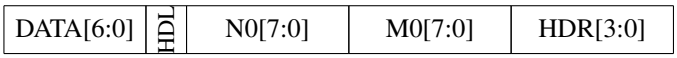
Given an example data stream STRM[$2MN - 1 : 0$] for a network size $(M = 2, N = 2)$ the stream package structure is:



i) Node to neighbor package: A package issued by a regular node to an neighbor. A 7 bit payload example with skipped address field:



j) Node to master package: A package send by an arbitrary node to the master. This package carries a fixed 2 byte width address field explaining the sender and a variable payload field. A 7 bit payload example package:



G. Network Layer [7]

This section covers routing, a full command listing and error handling.

1) Routing: The predefined network structure allows us to remain simple within this layer implementation. We do not have dynamically changing paths, multi paths or temporary unavailable path segments. Also we do not have cyclic paths within the network and fortunately the addressing is very cooperative. Thus routing is solved in a basic way. With the predefined network structure each node knows exactly which

[7] https://en.wikipedia.org/wiki/Network_layer

path is to be taken. To breakdown all possible routing use cases we can distinguish between 1) sequential routing and 2) contemporary forwarding. In further proceedings we denote the routing function as $R(...) \mapsto o$ and the simultaneous forwarding $F(...) \mapsto o$ with o being the output side list.

a) Sequential routing: Sequential routing implies that a package is fully received, processed before it may be routed to one or more sides. This can only happen prior the network addressing initialization because from that point on packages are simultaneously forwarded to each side.

$$R(type, i \in \{north\}, header\_type = 0\text{x}6) \mapsto o \subset \{south, east\}$$

$$R(...) = \begin{cases} \{south, east\} & if\ type \in \{origin, inter\_head\} \\ \{south\} & if\ type \in \{origin_I, head, inter\_node\} \\ \{\} & otherwise \end{cases} \quad (7)$$

where $type$ the node type; i the input sides; o the output sides

b) Contemporary forwarding: From Network Layer viewpoint contemporary forwarding is alike broadcasting. An incoming package is simultaneously forwarded before it is fully received and interpreted. If the header type equals to bit stream the package is not stored locally. Packages sent from master to nodes are forwarded as described in equation (8). For packages sent from node to master equation (9) is to be applied. Nevertheless we generally refer to broadcast if a package is addressed to all network participants, regardless to the underlying contemporary forwarding.

$$F(type, i \in \{north\}) \mapsto o \subset \{south, east\}$$

$$F(...) = \begin{cases} \{south, east\} & if\ type \in \{origin, inter\_head\} \\ \{south\} & if\ type \in \{origin_I, head, inter\_node\} \\ \{\} & otherwise \end{cases} \quad (8)$$

$$F(i \in \{south, east\}) \mapsto o \subset \{north\} \quad (9)$$

where $type$ the node type; i the input sides; o the output sides

2) Error Handling: The network layer does not detect neither handle transmission errors as well media access. Timings and eventually retransmissions have to be implemented by the bus master. For development purposes we offer the possibility to switch nodes into a verbose mode. They will in case of errors or special events send packages back to the master to express their faultiness.

3) Command Listing: Commands mentioned so far are listed in regard to their properties in table IV. Column is built-in characterizes whether a command $c \in C_{built\text{-}in}$ or C_{data} and column is stream whether $c \in C_{stream}$. Built-in commands are expressed by their header HDR[3:0]. The header not only implies the command issued but also routing hints. The address field is optional and depends how the package is addressed. All other commands are carried by the payload package and identified by the CID[2:0] flags. In that case the header does not indicate any routing hints. Commands do never trigger a response on the receiver side except of: 1) the built-in commands ping node 2) request network geometry and if verbose mode is active 3) on errors or special events.

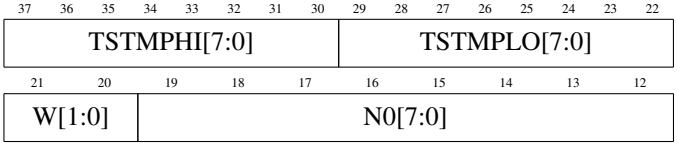
command	header $_{16}$	addr. length [Byte]	payload length [bit]	is built-in [bool]	to addr. [bool]	to addr. range [bool]	is stream [bool]	package length [bit]
heat wire	1	2	18	0	1	0	0	38
heat wire	2	4	18	0	0	1	0	54
heat wire stream	4	0	$\frac{mn}{4}$	0	0	0	1	$3 + \frac{mn}{4}$
sync. time	3	0	19	0	0	0	0	20
sync. offset	8	0	19	0	0	0	0	20
heating mode	E	0	19	0	0	0	0	20
payload	5	2	x	0	1	0	0	$20 + x$
net. geometry	D	2	0	1	0	0	0	20
geometry resp.	7	2	0	1	1	0	0	20
geometry req.	9	0	0	1	0	0	0	3
verbose mode	A	0	0	1	0	0	0	3
reset	B	0	0	1	0	0	0	3
ping node	C	2	0	1	1	0	0	20
ping response	D	2	0	1	1	0	0	20

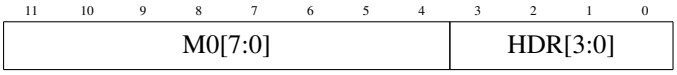
Table IV: Command Listing: This table gives an outline of command properties. Header type is base 16. The payload command payload length may occupy all possible sizes.

command	id$_{16}$	CID[2]	CID[1]	CID[0]
status report	0	0	0	0
TO BE DETERMINED .	x	x	x	x

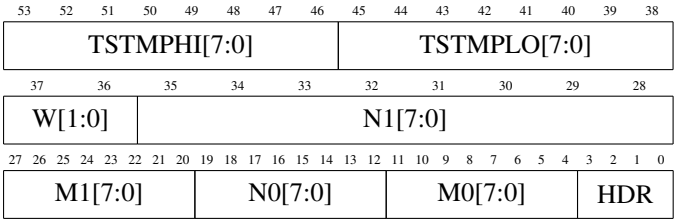
Table V: Payload Id Listing: **TO BE DETERMINED .**

a) Heat wire: Heating a wire has the condition that two adjacent nodes $\{n_1, n_2\} \in m \times n \mid n_1 = (y, x) \wedge n_2 = (y + 1, x)$ synchronously switch the wire ends to VCC and GND accordingly. Instead of sending a heat command twice the master sends it to the first node only. Because of the broadcast nature of the protocol this command is also received by the adjacent neighbor who consecutively reacts on the same command. This command affects wires in between nodes within a particle chain. The node address corresponds to the node that will activate it's south wires. Hence it can be issued to particles of type $\{origin, origin_I, inter\_head, head, inter\_node\}$ but not tail. Same applies if a heat command is addressed to a range of nodes. A heating command package issued to one node:





A heating command package issued to a range is defined as:



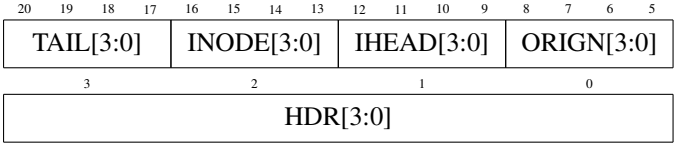
W1	W0	cardinal direction	wire
0	0	south	reserved (no wire active)
0	1	south	tx wire
1	0	south	rx wire
1	1	south	both

Table VI: Heat Wire Command Flags: Table describes the flag to wire mapping.

b) Heat wire - stream: The exact mapping of address to bit stream position and vice versa are declared in equation (5) and equation (6) on page 6. Before a bit stream can be issued to the network all particles need to have received a network geometry package. Instructions sent by the stream do not provide a timestamp field. The execution time is automatically planned immediately after the stream is passed through.

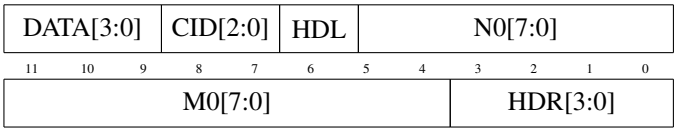
c) Sync time: This package is issued by the master and spreads contemporary all nodes. When this package is received the internal node timestamp is reseted to zero. Since the timestamp is granular in relation to the flank transmission delay, the reception time shift in between the first and last node is negligible. In very large networks this can be corrected with a sync offset. The offset of an arbitrary node is the sum of offsets per nodes on the path from origin to the current node. The internal timestamp is updated every millisecond.

d) Sync offset: Two byte payload describe 4 sync offset values for the following types of nodes: $\{origin, inter\_head, inter\_node, tail\}$, where head is seen to be equivalent to inter_node. Internally the offset is stored in one byte and interpreted as μs. If the offset is needed to be higher than the package field size allows this command may be issued multiple times. The total offset is the sum of all offsets sent so far. A subsequent sync timestamp resets the offset and the timestamp.



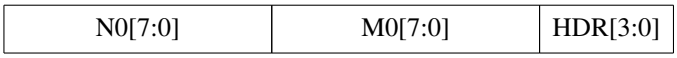
e) Heating mode: The heating mode defines in two bytes which delay and duty cycle to use for one heating command. A duty cycle of DCLE[7:0] $\in$ [0,255] corresponds to [1,100]% where the heating duration is scaled by $4ms$ and allows a maximum heating time of 1.02 seconds. The cycle length is fixed in the firmware and cannot be changed on runtime. A package reads as follows:

f) Status report: A status report can be issued from any node to the master if the particle was previously switched to verbose mode. Usual cases are errors or special events. Reports are not collision save and will cause undefined results if they occur at the same time. Therefore reports are suitable for development purpose but not release. The address field in this example package with 7 bit payload reflects the sender:



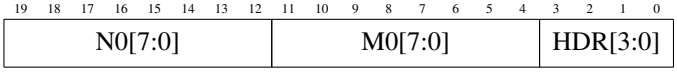
TODO: place here a list of all status reports. TO BE DETERMINED - the full list of status reports will be available after the protocol is fully implemented.

g) Geometry response: To determine the network dimension at runtime the master request the highest address in the network. A response package carries the network dimension in the address field and reads as follows:

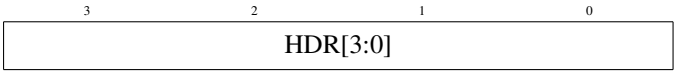


h) Geometry request: To determine the network dimension at runtime the master requests the highest address from the network. The request package reads in the following way:

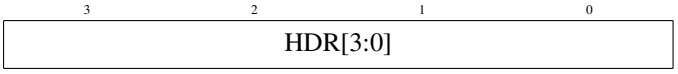
i) Network geometry: This command is used to announce the network geometry. Once it is broadcasted to any node subsequent broadcasts are ignored. A geometry that exceeds the real geometry will end in protocol malfunction whereas any subsets of the real geometry may be used to deactivate network parts. A reset must be performed if the geometry is to be updated. The dimension $m \times n$ is carried in the address fields M0[7], N0[7:0]. This package is defined as:



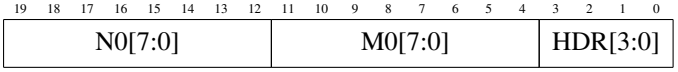
j) Verbose mode: For development purposes it is possible to allow nodes posting their status on errors or special events. Per default this mode is turned off but can be toggled by this command. This activates the mode of all network participants. A subsequent verbose mode command will toggle the mode again to disabled. The command is defined as follows:



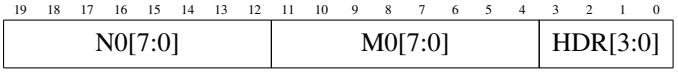
k) Reset: If the whole network is to be reset this command can be used to perform microcontroller reset on any node. The reset command does not provide a timestamp. Thus the reset timestamp is scheduled similarly as for a bit stream in section IV-G3b on the preceding page.



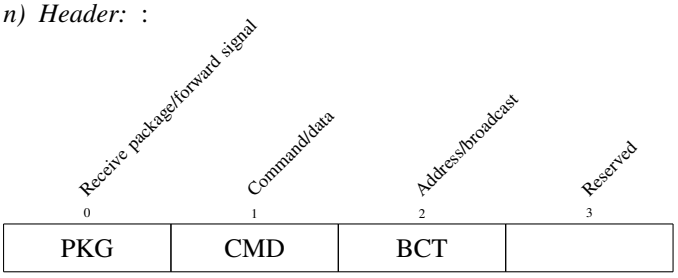
l) Ping node: For analysis this command allows the master to ping arbitrary nodes in the network. The addressed node responses with a ping response package. The package is read as follows:

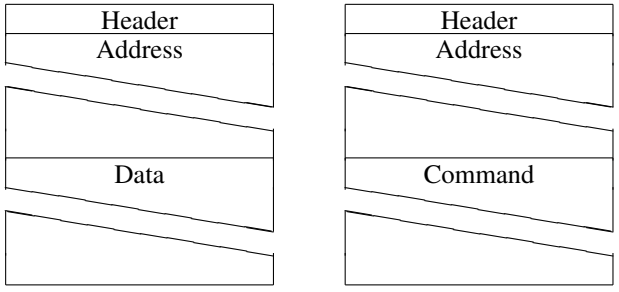


m) Ping response: For analysis the master may ping arbitrary nodes in the network. The response package contains the responding node's address in the address field and can be read as follows:



n) Header: :

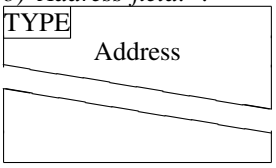


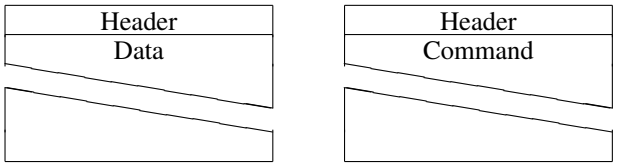


(a) data package (b) command package

Figure 10: A data or command package for a certain address or address range.

o) Address field: :

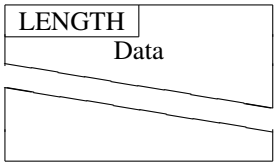




(a) data package (b) command package

Figure 11: A broadcast data or command package.

p) Data field: :



REFERENCES

[1] Matteo Lasagni and Kay Römer. Force-guiding particle chains for shape-shifting displays. *CoRR*, abs/1402.2507, 2014.

LIST OF FIGURES

List of Tables

Contents