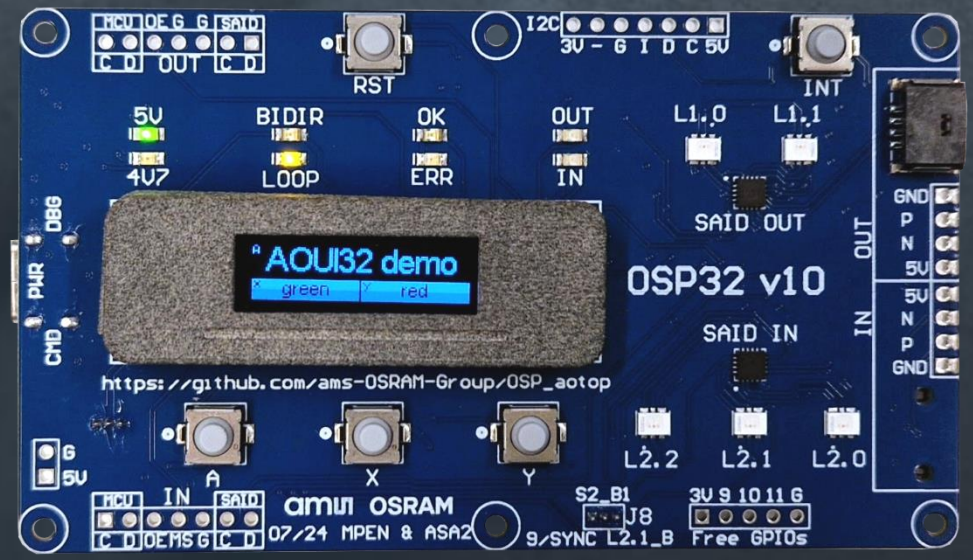
Sense the power of light

OSP on Arduino – Training – Appendix 1

Using the *Arduino OSP evaluation kit* – Star networks



2024 November 28

Part A1 – Star networks

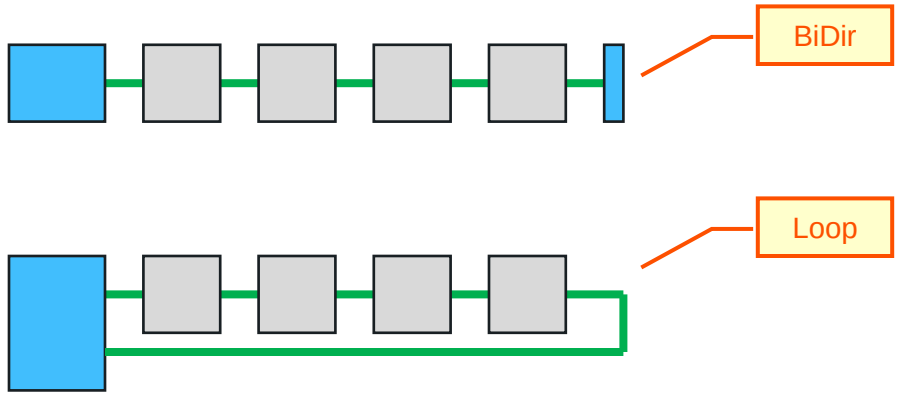
- Introduction
- Hardware
- Topologies
- Examples

OSP topologies

Daisy chain

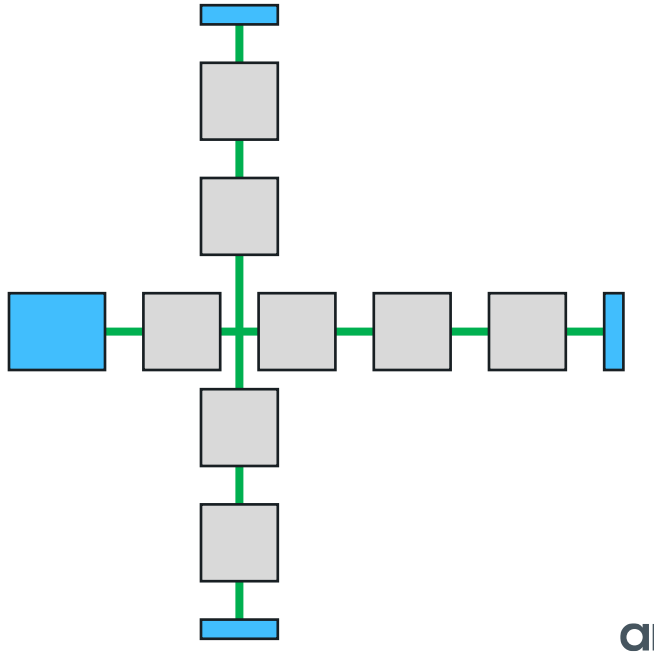
OSP 1.0

- As launched with RGBI's
- Mandates daisy chaining of OSP node
- Either in Bidir or in Loop mode



OSP 1.1

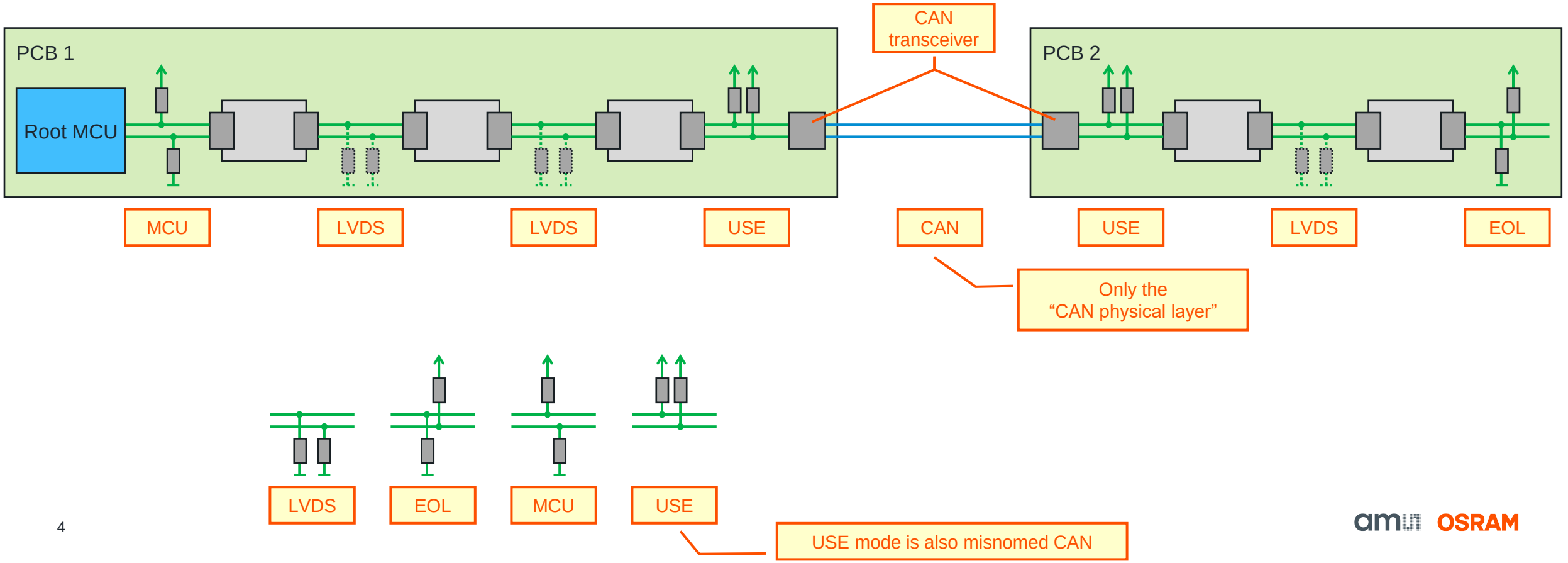
- The SAID offers the feature of star network
- Drawn on-purpose in a strange fashion ("star")
- Because the network does not allow for generic branches, i.e. *no* branch from a branch (no *trees*)
- Good news: a SAID *starts* a branch, but the branch can have nodes that do not support branching (like RGBI)



Intermezzo on CAN

Off-PCB connections

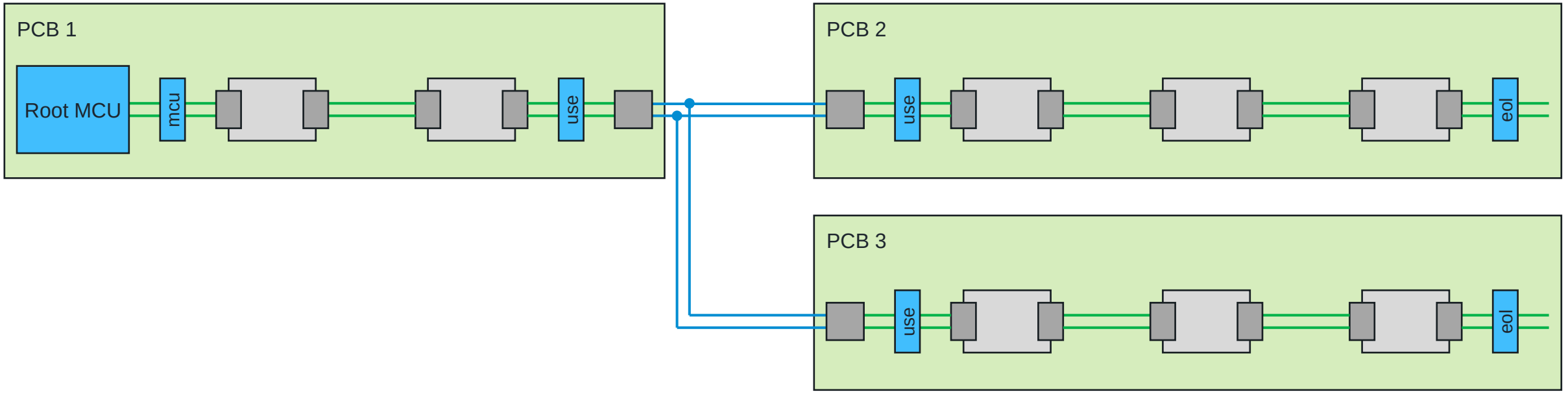
- We saw earlier that SIO ports in a chain are configured for LVDS, except the first (MCU) and the last (EOL)
- There is a fourth flavor: USE (unidirectional single-ended; misnamed CAN in OSP documentation)
- A standard component (a CAN transceiver) translates USE to CAN physical layer for more robust off-PCB connections



Using CAN for branching

Branch in CAN domain

- Splitting branches off the OSP chain is best done in the CAN domain
- Splitting LVDS causes signal degradation



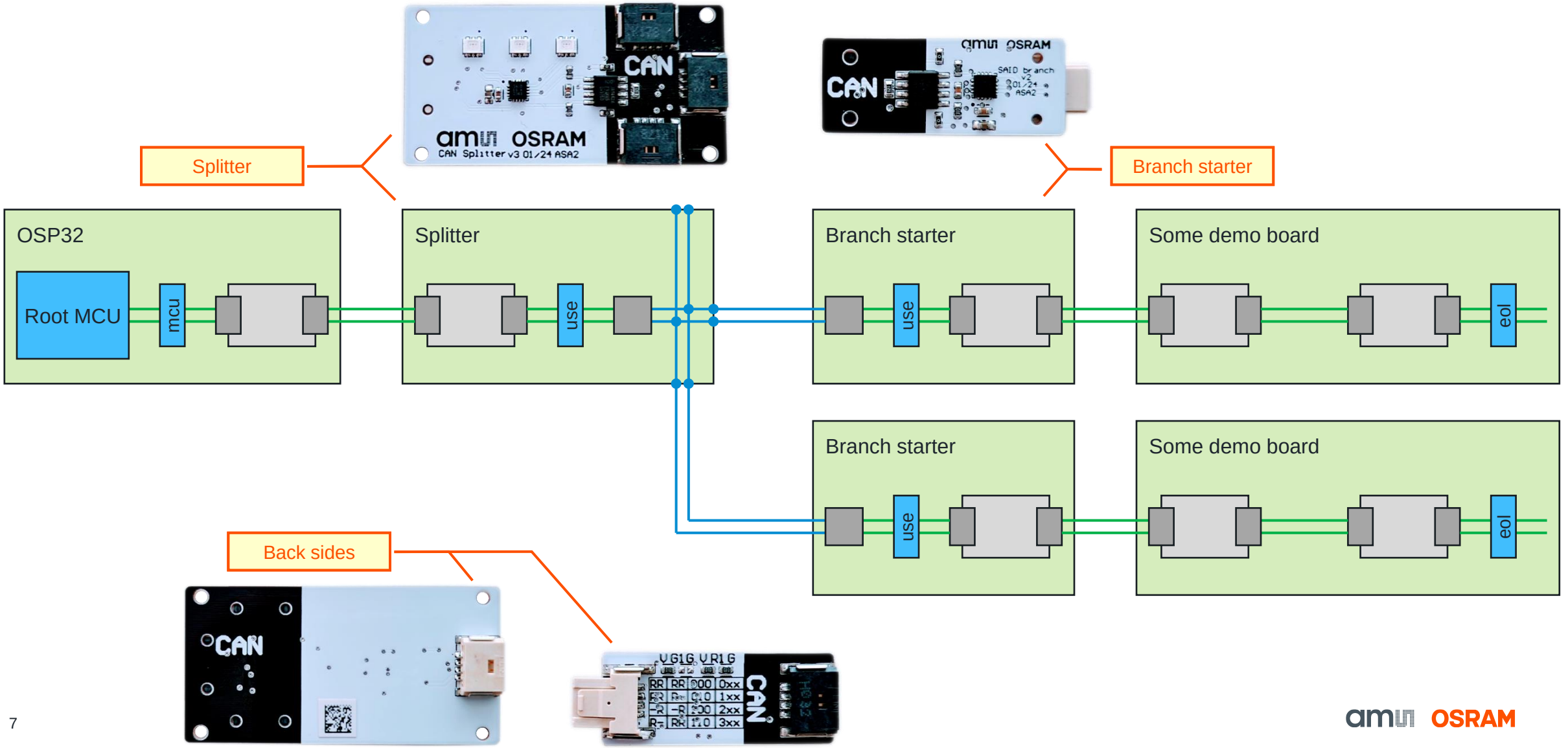
- (Technically it is possible to run branches in loop back mode, but we will skip that in this presentation)

Part A1 – Star networks

- Introduction
- Hardware
- Topologies
- Examples

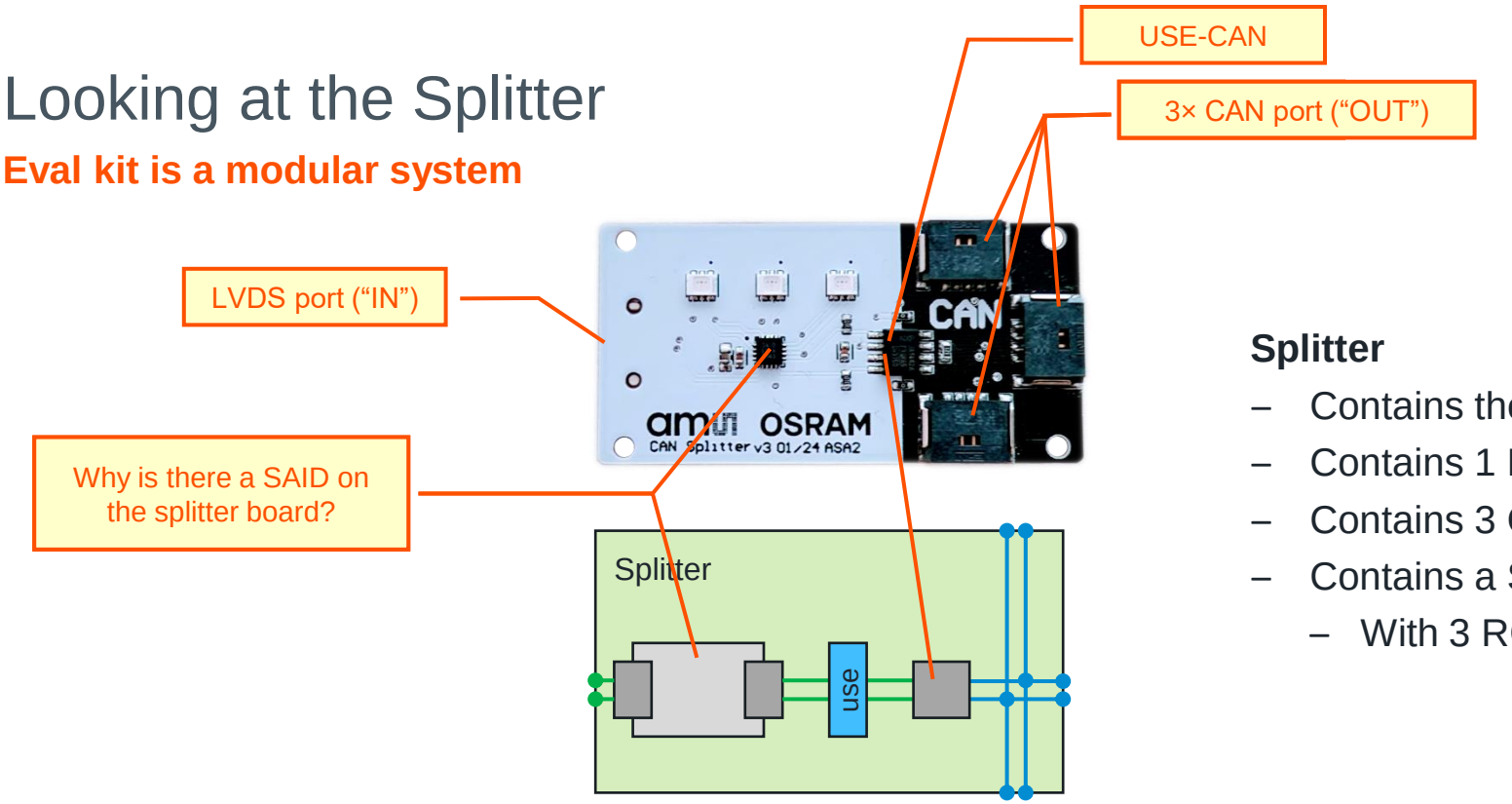
Two modules added: *Splitter* and *Branch starter*

Eval kit is a modular system



Looking at the Splitter

Eval kit is a modular system



Splitter

- Contains the USE to CAN transceiver
- Contains 1 LVDS (input) port
- Contains 3 CAN (output) ports (arbitrary choice for 3)
- Contains a SAID – why?
  - With 3 RGB triplets (just because)

OTP address	Bits							
	7	6	5	4	3	2	1	0
0D	CH_clustering[2:0]			haptic_driver	SPI_mode	sync_pin_en	star_net_en	i2c_bridge_en
0E					otp_add_en	star_net_otp_addr[2:0]		
0F								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
1A								
1B								
1C								
1D								
1E	Cust_lock							
1F	CRC2<7:0> (could also be used for other data)							

customer area

SAID

- The SAID is *not* needed
- It has one small feature
  - In the OTP, bit 0E.7 is reserved (by eval kit)
  - It identifies a SAID that splits the OSP chain
  - Purpose: enable the auto detect of the chain topology “is there a splitter” and “where is the splitter?”

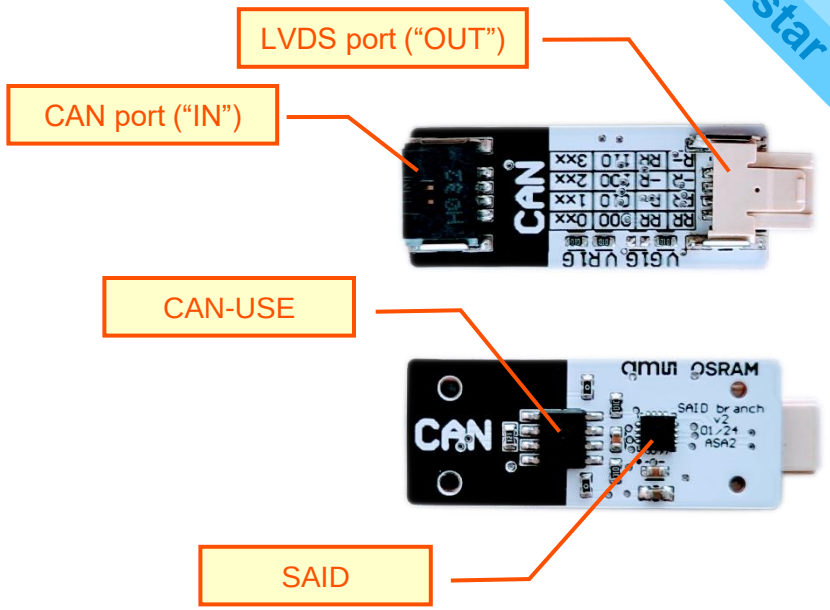
Looking at the Branch starter

Eval kit is a modular system

Branch starter

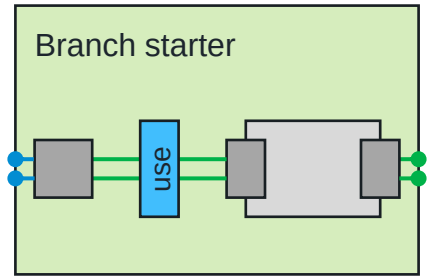
- Contains CAN (input) port
- Contains LVDS (output) port
- Contains CAN to USE transceiver
- Contains a SAID – why?

USE to CAN transceiver
is same as
CAN to USE transceiver
(it is a symmetrical device)
“input” and “output” is a helpful but limited view



SAID

- The SAID implements the key feature: the (a) branching mechanism
- Implementation:
  - SAID implements a telegram filter
  - The SAID has a 3-bit mask
  - There is an OTP bit enabling filtering
- SAID lets pass a telegram with
  - a broadcast address (000)
  - a groupcast address (3F0–3FE)
  - a unicast address, only if upper 3 bits (MSB) equal the mask



address	cast
000	broadcast
001 – 3EF	1007 unicast addresses
3F0 – 3FE	15 group addresses
3FF	reserved

Branch mask

Two ways to set the mask

9	8	7	6	5	4	3	2	1	0
Mask 2	Mask 1	Mask 0							
Nibble 2		Nibble 1				Nibble 0			

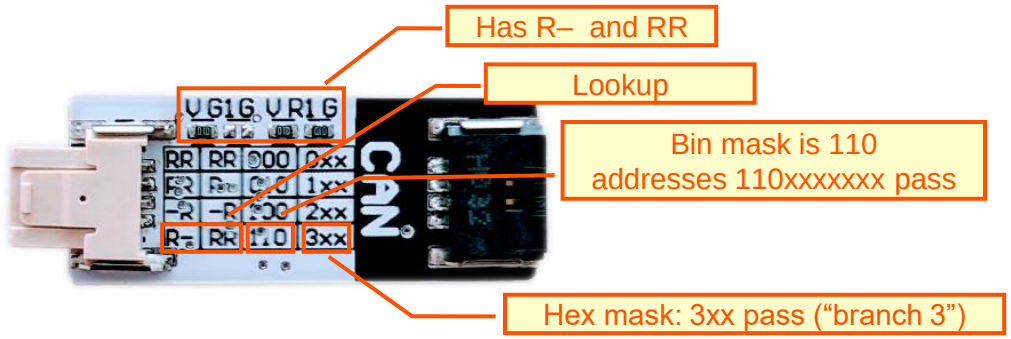
9	8	7	6	5	4	3	2	1	0
1	0	0	x	x	x	x	x	x	x
2		x				x			

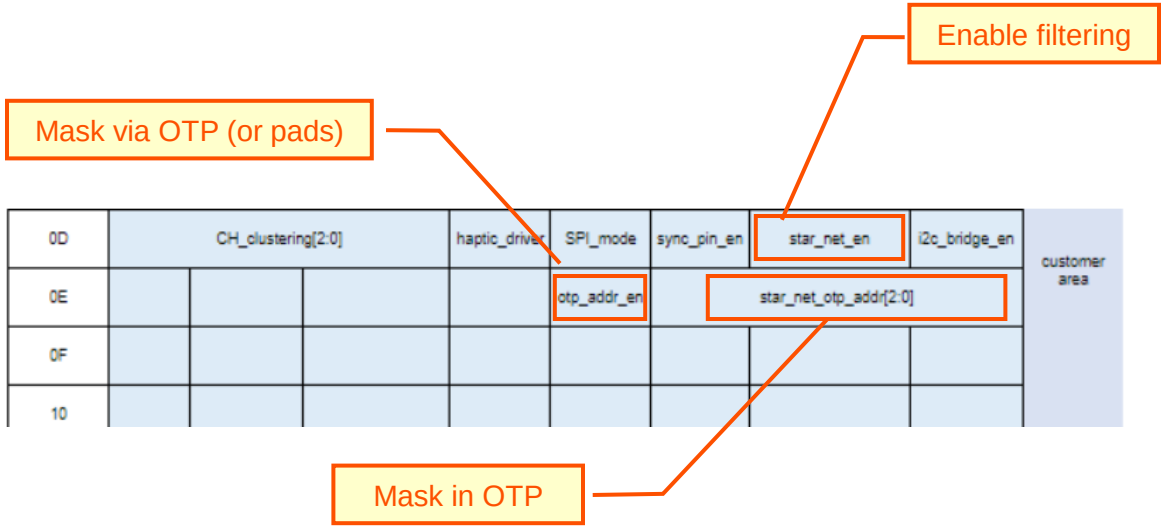
Notation for **mask**

- Write in binary 100xxxxxxx (correct but long)
- Write in hex 2xx (not completely correct)
- or as 000, 080, 100, 180, 200, 280, 300, 380 (add seven 0s)

There are two ways to configure the mask

- In both cases enable filtering/masking (**`star_net_en`**)
- Either enable **`otp_addr_en`**
  - Now the mask comes from OTP (**`star_net_otp_addr`**)
- Or disable **`otp_addr_en`** (as done on the Branch starter PCB)
  - Now mask comes from pads G1 and R1
  - By adding pull-up or pull-down or float
  - At the back of PCB is a table (subset of Table 9 in datasheet)





Part A1 – Star networks

- Introduction
- Hardware
- Topologies
- Examples

Intermezzo on initbidir

Subtle behavior details

The initbidir (and initloop) have subtle behavior

Specification of *OSP*, not specific to *SAID*

Open System Protocol 1.0

A device receives the INIT-command with its new address set in the address field. It then sets its own address to the new address, increments the value by one and sends the INIT-command to the next unit in the chain. The last unit in the chain returns its own address to the master and stops the initialization procedure. The response is either sent in backward or forward direction depending on the command flavor used.

Note: An initialized node receiving an INIT-command with a target address different from its own will fast-forward the INIT-command without incrementing the address and without interpreting the command.

When an init telegram arrives at a node, with destination address A

(1) Process command

- If the node is uninitialized: use A as address, step A
- If the node is initialized and address=A: step A
- If the node is initialized and address≠A: skip – but still do (2)

(2) Send telegram out

- If the node has successor: forward telegram (with new A)
- If the node has EOL: send response telegram with node's status

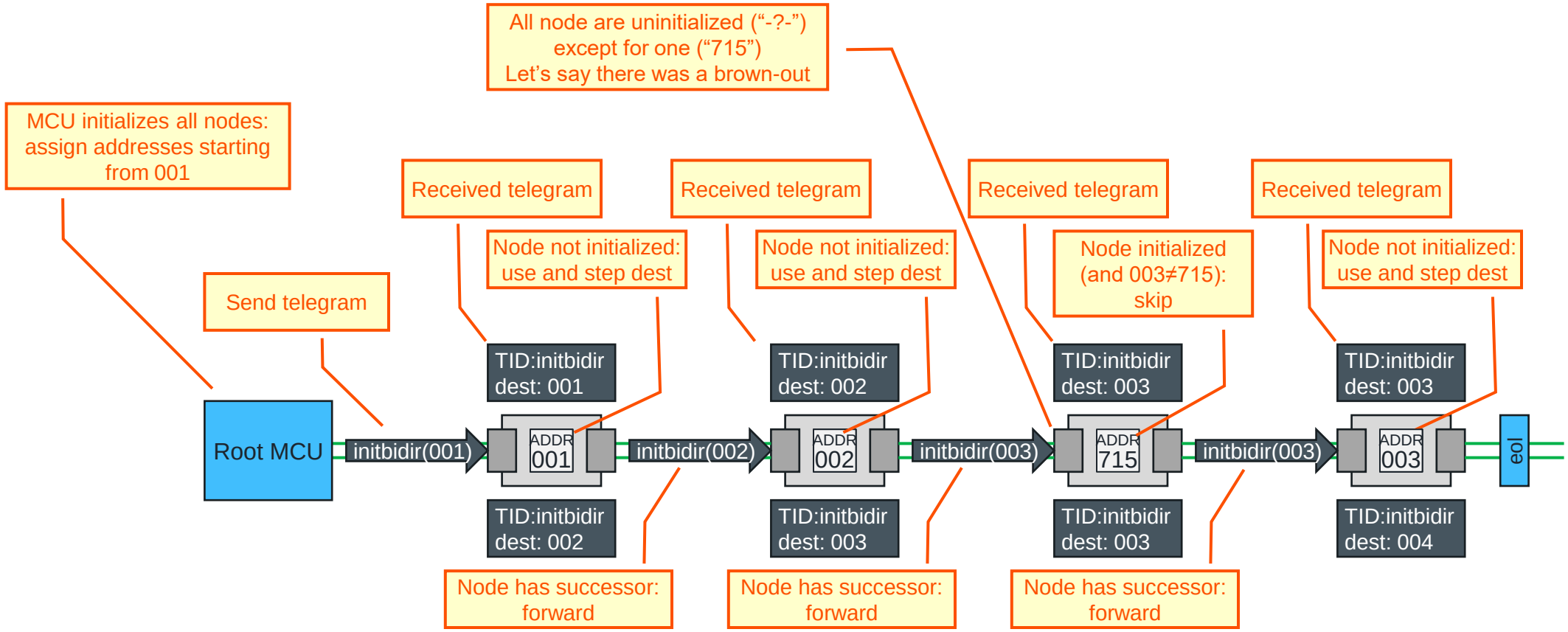
Normal case: ensures that in a "reset" chain all nodes get consecutive addresses

Repair case: ensures that a second identical init telegram leaves all as-is except nodes that had PoR

Branch case: ensures that a second init telegram with other base address skips all inited nodes

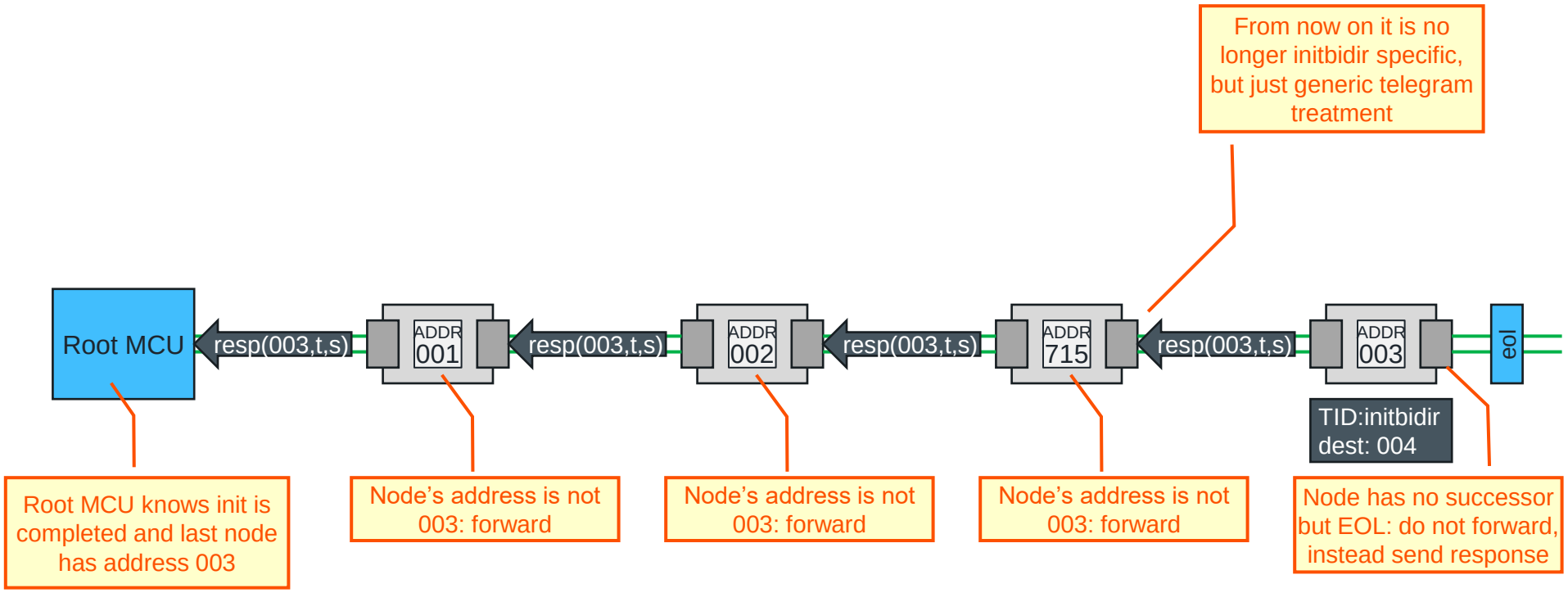
Intermezzo on initbidir

Seeing the subtleties



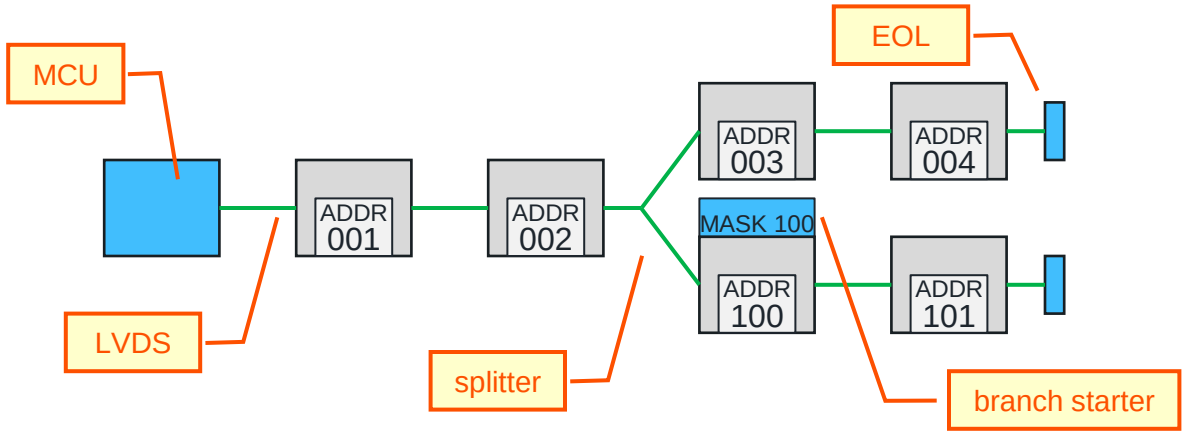
Intermezzo on initbidir

Seeing the subtleties



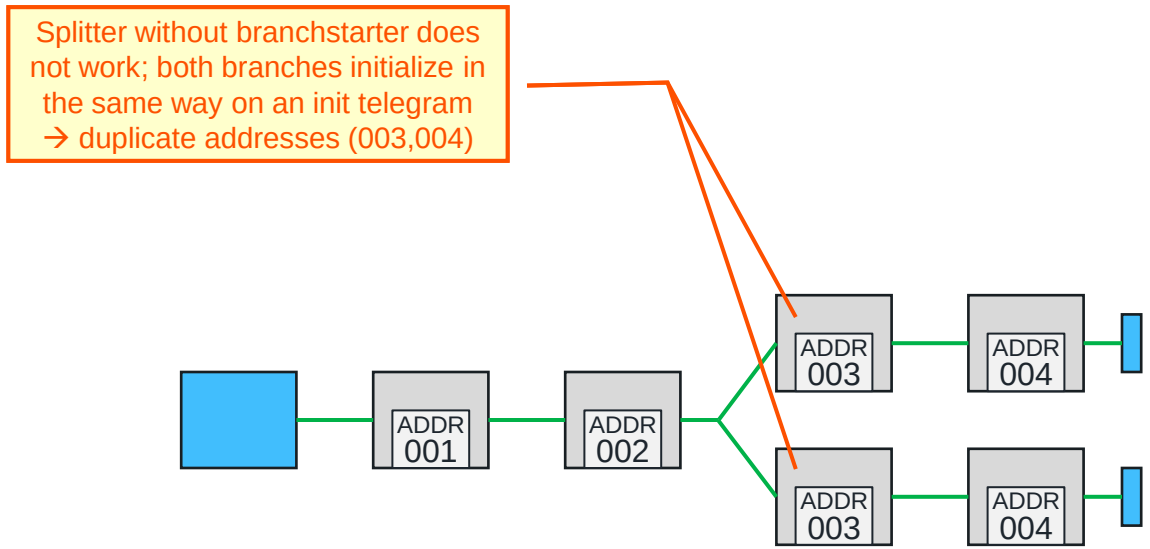
Branching topologies

Simplified drawing – legend



Branching topologies – Wrong

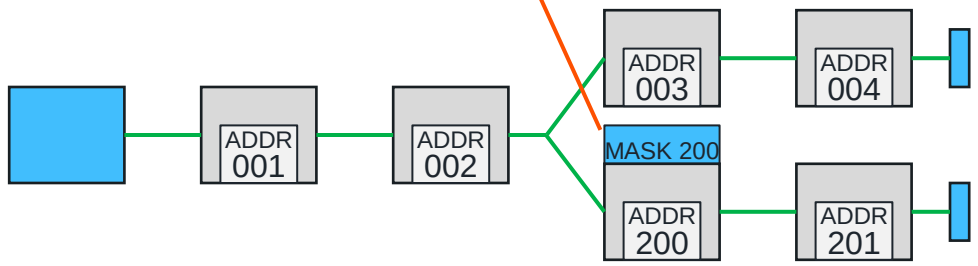
Splitter



Branching topologies – Wrong

Splitter plus branch starter for branch (not for trunk)

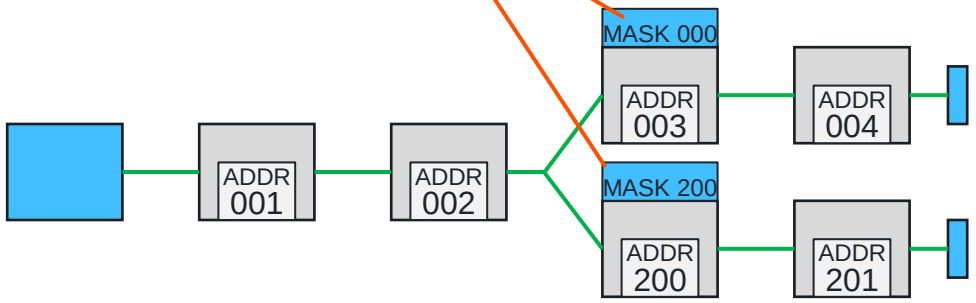
First send initbidir(001), then initbidir(200).
In branch, the filter stops the first init, but allows the second init.
Problem: second init causes response from *both* branches.
They collide at node 002, MCU doesn't see 2nd response.



Branching topologies – Success

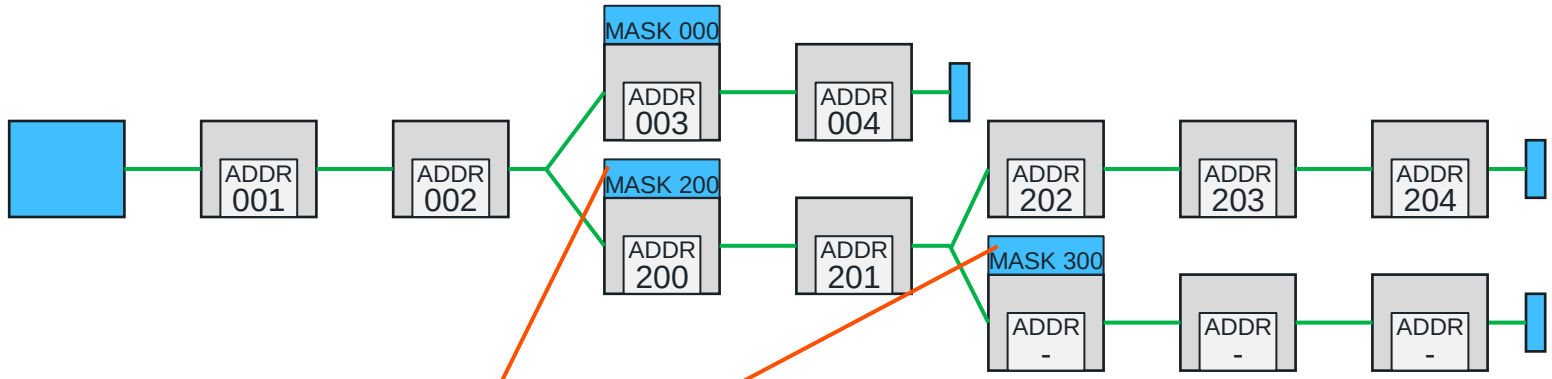
Splitter plus branch starter for branch and trunk

First send initbidir(001), then initbidir(200).
The trunk only sees the initbidir(001),
the branch only sees initbidir(200).
MCU sees both responses.



Branching topologies – Wrong

Cascaded filters

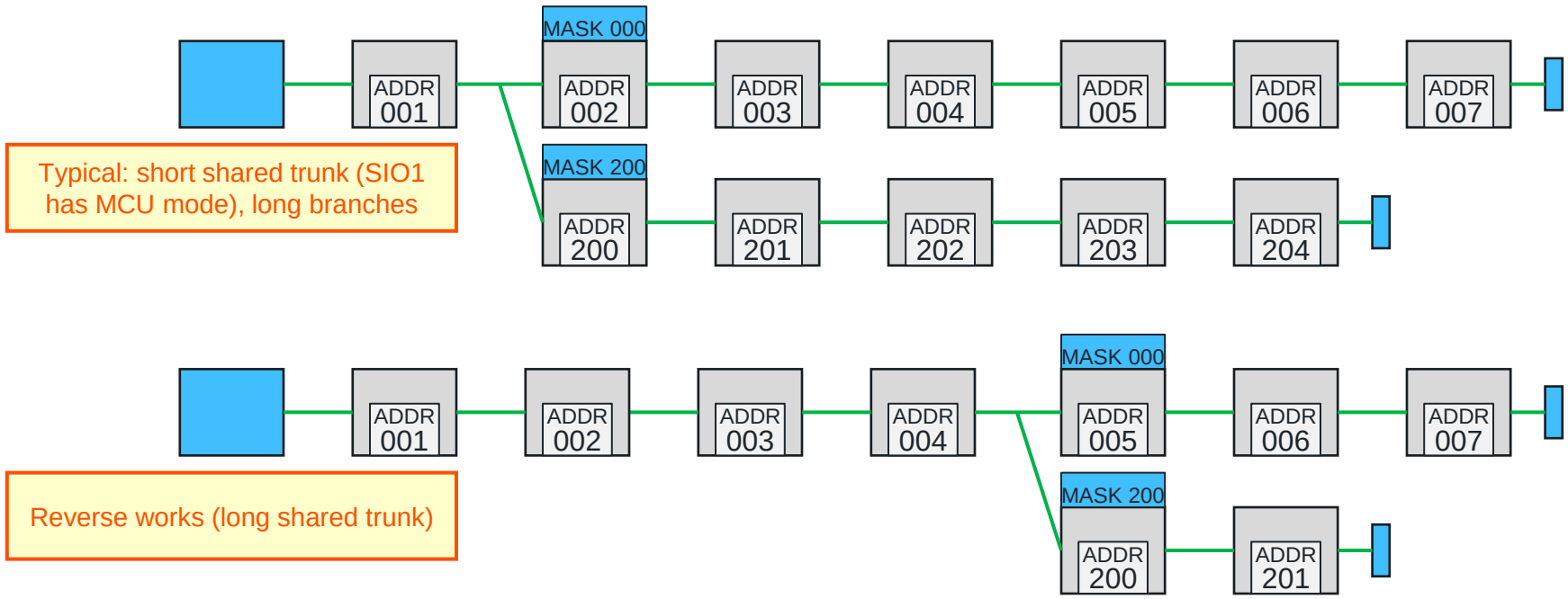


Two filters in a row, the 300 branch does not receive any (unicast) telegram

Branching topologies – simple

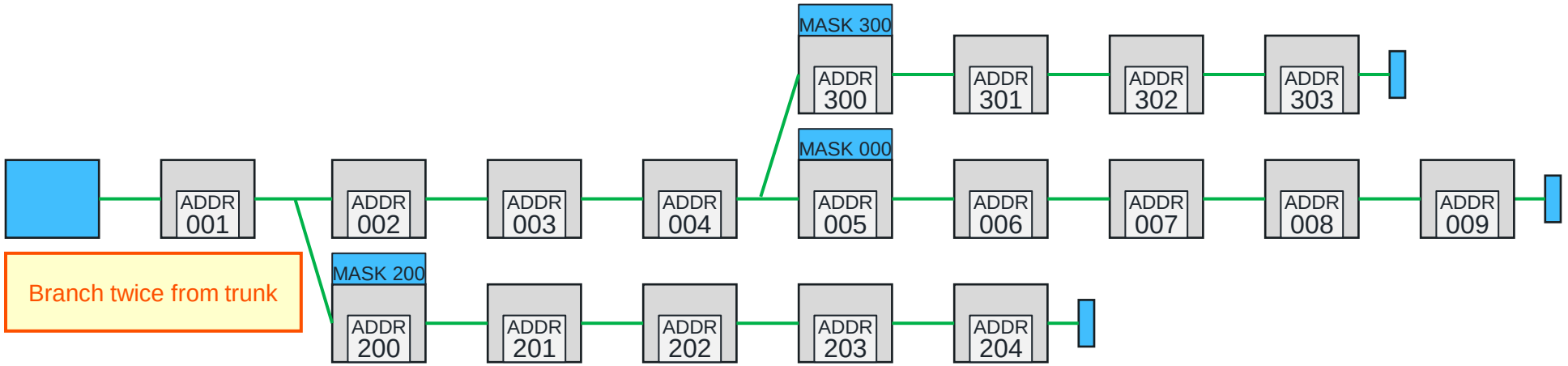
Splitter plus branch starter for all branches (including trunk)

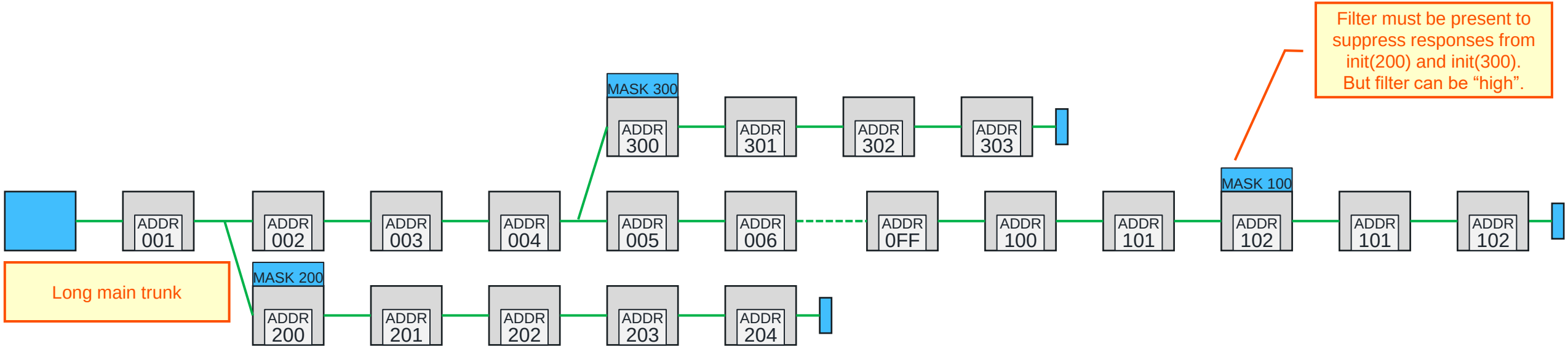
Note trunk does not have to be branch 0xx; trunk is the init(txx) that comes first after reset (but convention dictates 0xx)



Branching topologies - advanced

Splitter plus branch starter for all branches (including trunk)





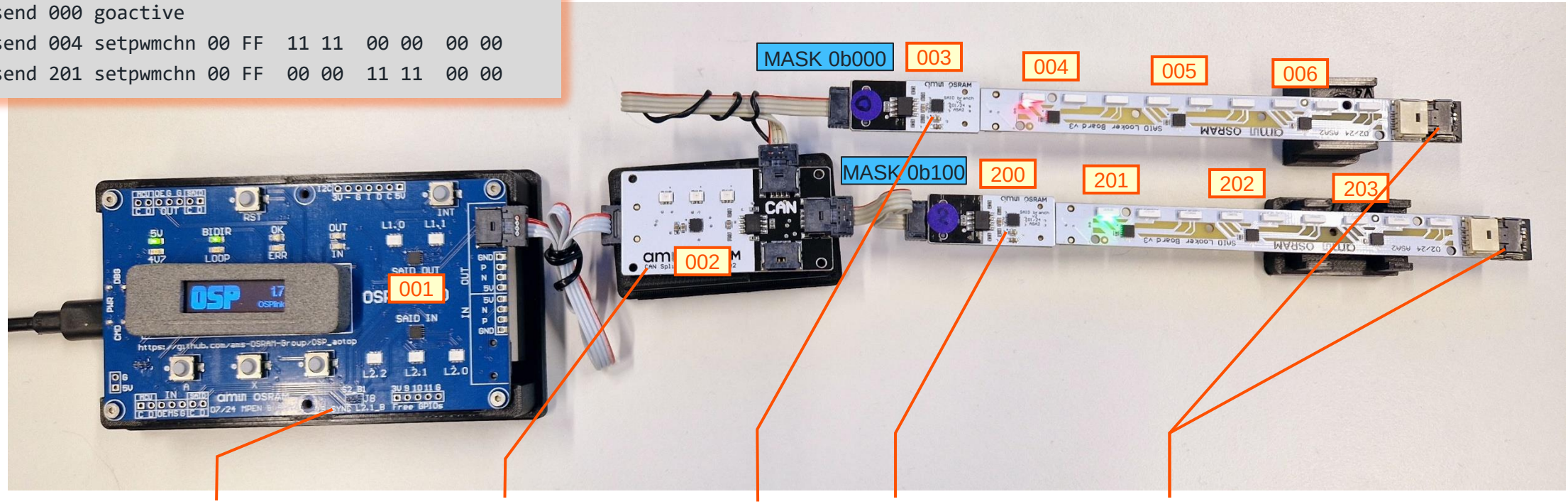
Part A1 – Star networks

- Introduction
- Hardware
- Topologies
- Examples

Demonstrating simple branching

Using OSPlink (from eval kit)

```
>> osp dirmux bidir
>> osp send 000 reset
>> osp send 001 initbidir
>> osp send 200 initbidir
>> osp send 000 clrerror
>> osp send 000 goactive
>> osp send 004 setpwmchn 00 FF  11 11  00 00  00 00
>> osp send 201 setpwmchn 00 FF  00 00  11 11  00 00
```

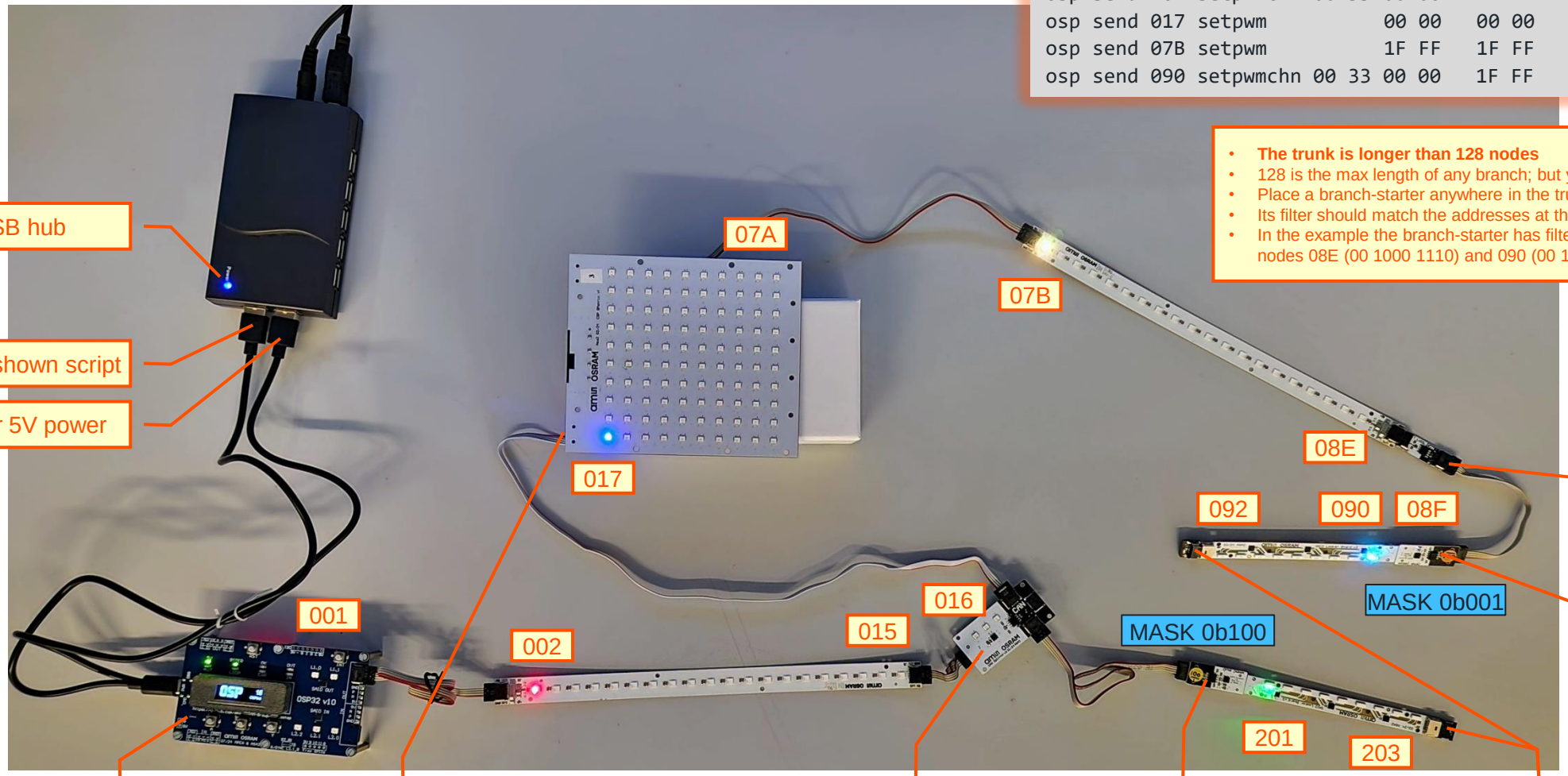


Demonstrating advanced branching

Using OSPlink (from eval kit)

```
osp send 000 reset
osp send 001 initbidir
osp send 200 initbidir
osp send 000 clrerror
osp send 000 goactive
osp send 002 setpwm           1F FF   00 00   00 00 // red
osp send 201 setpwmchn 00 33 00 00   1F FF   00 00 // green
osp send 017 setpwm           00 00   00 00   1F FF // blue
osp send 07B setpwm           1F FF   1F FF   00 00 // yellow
osp send 090 setpwmchn 00 33 00 00   1F FF   1F FF // cyan
```

- **The trunk is longer than 128 nodes**
- 128 is the max length of any branch; but you can extend the trunk
- Place a branch-starter anywhere in the trunk after the splitter
- Its filter should match the addresses at that spot in the chain.
- In the example the branch-starter has filter 0b001; this is between nodes 08E (00 1000 1110) and 090 (00 1001 0000)



ams OSRAM


ams
OSRAM