SPE solution for fast communication with 10BASET1L, 42 Byte in 50 µs in Ring Topology

Dr. Hartmut Schorrig
www.vishia.org

2023-09-05

Table of Contents

1 SPE as standard connection in electrical plants....2
2 Solution: Ring communication with up to 40 data Byte per telegram....3
3 Currently implementation using a FPGA and hard wired driver....4
4 Capability of frame and clock synchronization....5
5 Data exchange just in time....5
6 Using inside a standard Ethernet ring....6
7 Application of this kind of SPE communication....6
8 Links....7

1 SPE as standard connection in electrical plants

The Single Pair Ethernet technology seems to become a proven concept for communication wiring in plants.

For electrical signal transmission, often a cycle of 50 µs are proper. This is fast enough for proper control of electrical grids, and also possible for familiar power switches (IGBT) and for cycle times of controllers. With a longer step time, such as 1 ms, the "*fault ride through*" approach is not possible. On faults, the current and voltage control should be faster than the normal smoothing times of electrical elements (usual inductance).

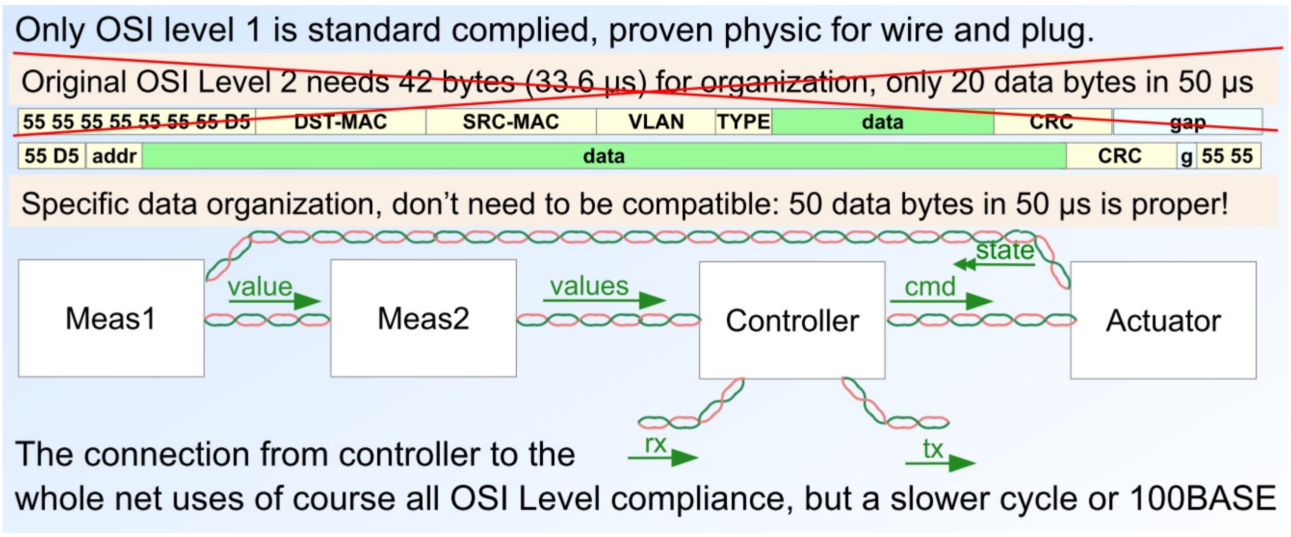
The traditional approach is either using conventional analog lines, or specific solutions often with fiber optic cable. Using the Single Pair Ethernet offers a common proper way for communication.

The length of lines in an electrical plant are often greater than 100 m, and the immunity to electrical interference must be high. For that reason, using Ethernet, a baud rate of 10 MBit/s combined with the noise resistant Manchester Coding of the signals may be proper. This seems to be contrary to the necessary cycle time of 50 µs. But the question is: how many data should be transmitted in one cycle.

2 Solution: Ring communication with up to 40 data Byte per telegram

From the Standards of Ethernet Communication due to the known OSI-7-Level model, only the first level, the physic, is used. The other levels are changed.

Base on the physical level, all elements can be used due its standards: Cable, plugs, and also the PHY chips.


Only OSI level 1 is standard complied, proven physic for wire and plug.

~~Original OSI Level 2 needs 42 bytes (33.6 µs) for organization, only 20 data bytes in 50 µs~~

Specific data organization, don't need to be compatible: 50 data bytes in 50 µs is proper!

The connection from controller to the whole net uses of course all OSI Level compliance, but a slower cycle or 100BASE

Figure 1: TelgTopology/SpeRring50usOsi.png

The image above shows the data content due to OSI level two. Because of the ring topology, not using a router, all addressing bytes are not necessary. The addressing of stations is determined by the hardware connection, there is no routing. Only one data word, two bytes are used for an sender identification to check the connection and help detect faulty connections.

Also the gap is reduced to a very small gap. The gap is necessary for standard ethernet communication with bus topology to detect a free line, unnecessary for ring communicaiton. But a small gap (~ 1 µs) remains. But the synchronizing bits remains. They are necessary for clock synchronization. And the CRC remains. Does the shortened gap violate the physical standard of transmission. That should be anwered by the properties of the given PHY chips. If they support a shorter gap, it is proper. Influences to signal qualities are not expected.

With that decision 40 byte can be transmitted as the payload for the telegram. This needs 32 µs, with the sender address bytes it is ~34 µs. CRC needs 3.2 µs, the gap 1 µs. Hence 12 µs remains for the synchronization phase. This is enough to exchange the few values for current, voltage and power for three phases, some state signals and also transmit actor signals for converter inputs.

With 40 byte for example 20 values á 16 bit are able to transmit. The ring topology allows exchange bytes from one station to the next of the telegram.

3 Currently implementation using a FPGA and hard wired driver

The solution used and tested now was developed from 2021. In this time a specific PHY chip for the communication with the given requirements for the prototype tests was not found. Hence, the decision was, implementing the necessary logic for communication transfer and also for the driver in an FPGA (Lattice XO-4000). The signal driver itself are implementing using standard magnetics components (ferrite) for signal potential separation and noise suppression, but using conventional driver for the current to the line of 20 mA (level 1 V, resistance 2 * 100 Ohm). The signal input was proper supported by the FPGA itself with its differential input capability.

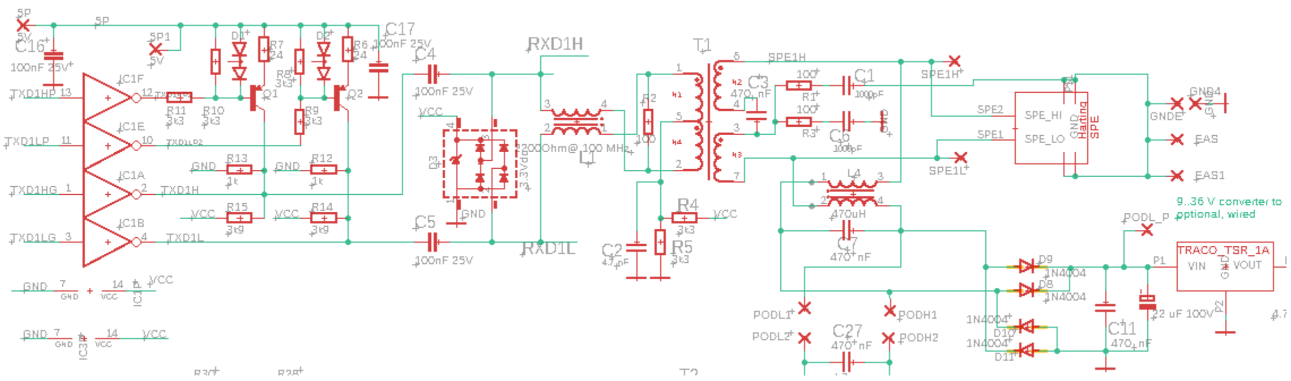


Figure 2: Board/boardAsch_CurrSrc.png

The image above shows the hardware for signal driving. It consists of switched current sources and open-drain switches, which forces + or – 20 mA in the line. The inputs of the open drain driver circuits are connected to FPGA outputs. The RXD1H and ~L are immediately a FPGA difference input. Right side you see also a solution for "*Power over Dataline*". This solution works, tested with a 100 m cable without problems, and tested also with a PC connection with the standard ethernet protocol, see chapter Using inside a standard Ethernet ring

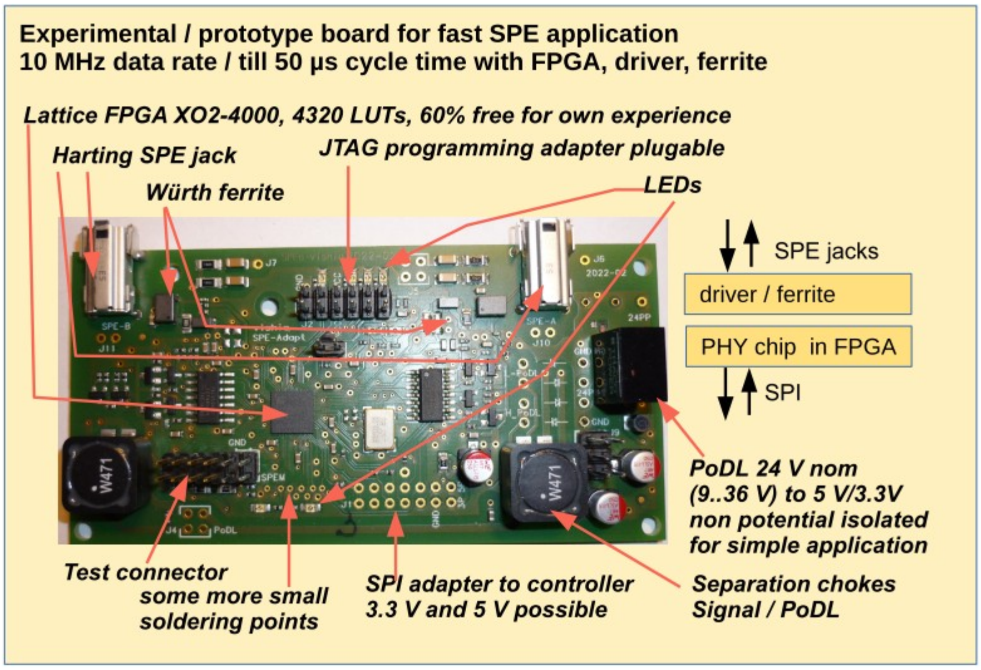

Experimental / prototype board for fast SPE application
10 MHz data rate / till 50 µs cycle time with FPGA, driver, ferrite
Lattice FPGA XO2-4000, 4320 LUTs, 60% free for own experience
Harting SPE jack
JTAG programming adapter plugable
Würth ferrite
LEDs
SPE jacks
driver / ferrite
PHY chip in FPGA
SPI
PoDL 24 V nom (9..36 V) to 5 V/3.3V non potential isolated for simple application
Test connector
some more small soldering points
SPI adapter to controller 3.3 V and 5 V possible
Separation chokes Signal / PoDL

Figure 3: SpeA-board-overview.png

Now, doing in the future, it is necessary to check whether the given driver can be replaced by the outputs of a physical chip, whereby the ADIN1100 from Analog Devices will be preferred.

4 Capability of frame and clock synchronization

The clock synchronization is an important capability: The stations connected via SPE should have a synchronous clock for building Pulse Width Modulation (PWM) signals and average filters for connected A/D converter synchronous to the controlling cycle. This means a clock which is provided from the FPGA should be synchronous with a less jitter of 30..300 ns (30 ns between immediately neighboring two stations). This clock synchronization is done by the FPGA solution. It is to check whether a used PHY chip can support it also.

The image right shows the received *Start Frame Delimiter* (SFD) in blue and the transmitted SFD in the red track on the cursor positions. Both have a jitter from max 30 ns. The scope is synchronized with a frame signal from the master coming from the controller, which determines the 50 µs cycle. The internal clock of the FPGA of 100 MHz is not controlled, it is free running. But the Clock Enable signal (CE) which controls the FlipFlop switching is synchronized, using typical 10 clock edges for pre dividing, but sometimes 9 or 11 clock edges controlled by a state machine. This logic evaluates the rising edges of all synchronization bits till just the SFD. This pre divided controlled CE signal of ~ 10 MHz is offered also to the controller for PWM signal building. It is synchronized over all stations in the SPE ring. In this manner all stations works with the same 50 µs frame with a 10 MHz clock. This also offers a time synchronization. You can view a living gif image on [1] or open the image immediately with [2].

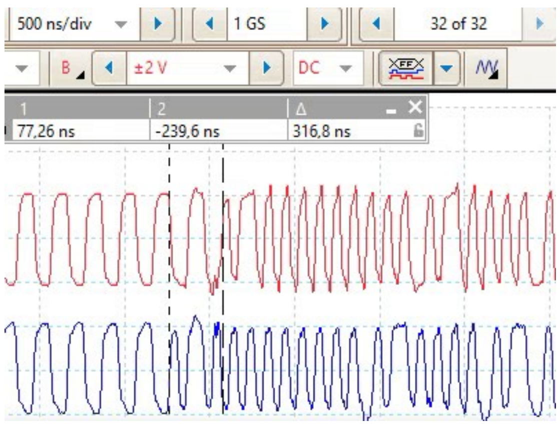


Figure 4: SFDdelayJitter.gif

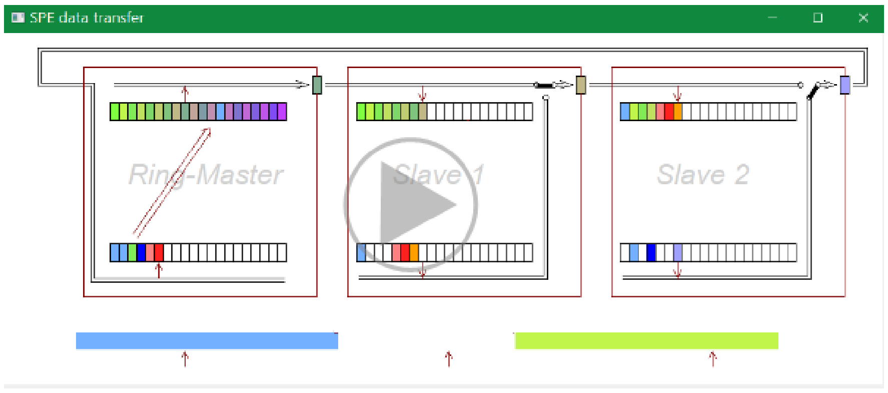
5 Data exchange just in time

Normally an Ethernet telegram is received as a whole, the CRC check is done, then the data can be used. This offers checked and consistent data. But it means also, that additonal one cycle (50 µs) dead time occurs between transmitting and receiving, additionally to the given dead times for data averaging and calculation. It means at least 150 µs dead time is given in the controlling cycle.

If the fine timing is known where data are processed to transmit, and where data can be taken from the received telegram to process it, the one cycle dead time can be ommitted. The reaction of the system in the loop is faster. For that a "*data just in time*" or "*data on the fly*" approach is implemented:

The data to transmit via SPE are taken from the RAM of the controller immediately before they are transmitted. This is done via DMA (Direct Memory Access) forced by the FPGA via SPI interface (Serial Peripheral Interface to the controller). The FPGA is the Master of this SPI communication. It determines the timing of access. The SPI communication should be prepared before by the controller software.

It means a value used for SPE should be written to the RAM by the controller software a small time before. A Jitter in software should be regarded, approximately ~ 5 µs should be possible. This new calculated data are immediately (5..1 µs later) transmitted, needs ~ 4 µs for bit shifting and transmission itself, and are written after ~9..5 µs in the RAM of the receiving controller. This is not only for the next neighbour station, it is also for more far stations, because the receiving signal is transmitted immediately forward in the ring topology as seen in Figure 4: SFDdelayJitter.gif with a delay of ~ 0.3 µs per station and a delay of 0.5 µs for a 100 m cable. Because of also regarding a jitter in software of ~ 5 µs (or a little bit more), the signal is available just in time after 14… µs in the other station and can be used. Of course this is only possible if the software execution is fine tuned to this timing. The position of the data should be adjusted to the software execution. But now the dead time of a controller cycle is meaningful reduced.



The image right shows a snapshot of this communication. The Ring-Master left writes just data in the 11. data word. The transmission processes just the 8th data word, the 11th. one comes in 4.8 µs later. The 7th data word is written just now in the RAM of the next station. The newly written 11th word will come 6.4 µs later and can be processed after a proper waiting time because jitter, or it can be checked by polling also till it was written. In the right box for *Slave 2* you see not forwarding the received data bytes (as in *Slave 1*) but just reading data bytes from the RAM of this controller, which is forwarded replacing the original received data word. In the same kind the orange and red words in Slave 2 and also back to the Ring-Master comes from the Slave 1. The replacing of data words in the ring topology telegrams is similar as also known in the EtherCAT communication, see [3].

You can visit a living image and also more explanation in [4].

6 Using inside a standard Ethernet ring

A test application was built using a PC in the ring with a standard Ethernet plug. The communication parameter are set under Windows to 10 Mbit/s. From the 4 twisted wires two ones are used, one for transmit, one for receive, immediately connected (without any switch) to the SPE plugs. This works because the signal level is compatible. The main goal for this test was the compatibility test of the data protocol. The negotiation signals for connection (NLP pulses, [5]) are also supported by the transmission station. Of course this does not work with the reduced telegram for the 50 µs cycle. It needs the OSI-7-level full compatibility. But this is supported by the specific software of the controller, writing correct information to the data positions. The cycle for this test was 1 ms with 1000 Data bytes in the telegram. The standard gap was used.

You can see some details in [6] in the chapter 3. Usage with standard communication PC in ring / as receiver.

7 Application of this kind of SPE communication

Till now this approach is only used in students work on the TU Ilmenau in Germany, by the master thesis of Paulami Das [7].

The usage of the PHY chip preferred the AD1100 should be tested in the future.

For more application ideas you can write immediately to the developer and author of this article: hartmut.schorrig@vishia.de (Freelancer, Germany).

8 Links

[1] https://vishia.org/spe/index.html Web page by the author given a overview over this SPE topic.

[2] https://vishia.org/spe/videos/SFDdelayJitter.gif Gif shows the jitter from receiving and forward transmitting signals in the ring

[3] https://en.wikipedia.org/wiki/EtherCAT.

[4] https://vishia.org/spe/html/SpeA-Manual/SpeA-telgRingCommJustintime.html.Explanation and demonstration of the data just in time concept from the author.

[5] https://en.wikipedia.org/wiki/Autonegotiation Explanation of NLP and FLP for standard Ethernet communication for auto negotiation and living signals.

[6] https://vishia.org/spe/html/SpeA-Manual/SpeA-Videos.html Explanation to some videos for the authors work. The videos are partially in German language.

[7] Das, Paulami: "Commissioning of an electric grid controller in Simulink in Embedded Control programming in C++ along with distributed processing on several microcontrollers with Single Pair Ethernet on FPGA" Master thesis (en) TU Ilmenau, EI, Elektrische Energie- und Steuerungstechnik, December 2021