

## Uthernet Interface Specification

	Organisatie / Organization	Datum / Date
<b>Auteur(s) / Author(s):</b> Eric Kooistra	ASTRON	
<b>Controle / Checked:</b> Andre Gunst	ASTRON	
<b>Goedkeuring / Approval:</b> Andre Gunst	ASTRON	
<b>Autorisatie / Authorisation:</b> <b>Handtekening / Signature</b> Andre Gunst	ASTRON	

© ASTRON 2011  
All rights are reserved. Reproduction in whole or in part is prohibited without written consent of the copyright owner.

## Distribution list:

---

Group:	Others:
Andre Gunst (AG, ASTRON) Eric Kooistra (EK, ASTRON) Daniel van der Schuur (DS, ASTRON) Harm-Jan Pepping (HJP, ASTRON)	Gijs Schoonderbeek (GS, ASTRON) Jonathan Hargreaves (JH, JIVE) Salvatore Pirrucci (SP, JIVE)

## Document history:

---

Revision	Date	Author	Modification / Change
0.1	2011-10-26	Eric Kooistra	Draft.
0.2	2011-10-31	Eric Kooistra	Added references to other serial data transmission protocols.
0.3	2012-06-29	Eric Kooistra	Define preamble word instead of IDLE word.

## Table of contents:

---

1	Introduction.....	4
1.1	Purpose .....	4
1.2	Scope .....	4
2	Uthernet packet structure .....	5
2.1	Data width.....	5
2.2	Preamble word .....	5
2.3	SFD word.....	5
2.4	Type/Length word.....	5
2.5	Payload.....	5
2.6	CRC.....	6
3	Transporting Uthernet packets.....	7
4	Appendix: Standard serial data transmission protocols .....	8

## Terminology:

---

CRC	Cyclic Redundancy Check
eop	end of packet
ETH	Ethernet
LSBit	Least Significant bit
LVDS	Low Voltage Differential Signaling
MSBit	Most Significant bit
PHY	Physical interface
PRE	Preamble
SFD	Start of Frame Delimiter
sop	start of packet
TLEN	Type / Length
UTH	Uthernet

## References:

---

1. [http://en.wikipedia.org/wiki/Ethernet\\_frame](http://en.wikipedia.org/wiki/Ethernet_frame)
2. <http://www.easics.com/webtools/crctool>
3. "SerialLite II Protocol Reference Manual", [www.altera.com](http://www.altera.com)
4. "Aurora 8B/10B Protocol Specification", [www.xilinx.com](http://www.xilinx.com)
5. "Data Path Interface Description", ASTRON-RP-394, E. Kooistra
6. "Specification for module interfaces using VHDL records", ASTRON-RP-380, E. Kooistra

# 1 Introduction

## 1.1 Purpose

This document specifies the UTH packet structure. UTH is short for Uthernet and it is called Uthernet because the packet definition resembles the Ethernet packet structure [1]. The most important difference is that the UTH packet is intended for point-to-point links and therefore its packet header does not have address fields, hence Uni-Ethernet or Uthernet.

## 1.2 Scope

The Uthernet protocol provides a simple packet interface that can run directly on a PHY interface, because it suits any data width, it does not require frame control signal support and it does not provide lane bonding.

If a standard link-layer protocol like Altera SerialLite II [3] or Xilinx Aurora [4] is used (see Appendix 4) then it is not necessary to use the Uthernet protocol, because thanks to the frame control support these link-layer protocols can directly transport the data payload while still providing the error detection of the decoder.

This Uthernet packet specification makes the DP PHY frame interface that was defined in [5] obsolete.

## 2 Uthernet packet structure

Figure 1 shows the complete Uthernet packet. The header contains 3 data words (PRE, SFD and TLEN), the payload contains at least 1 word and the tail contains 1 word (a CRC).



**Figure 1: Uthernet packet structure**

The Uthernet packet starts with a preamble (PRE) word and a start of frame delimiter (SFD) word. Together these mark the start of a packet. The preamble word that is part of the Uthernet packet is a preamble for the SFD. After the SFD the type/length (TLEN) field indicates the number of payload data words. The payload data contains at least 1 data word and at most as many as the type offset value – 1. The packet ends with a cyclic redundancy check (CRC) which is used to detect any corruption of payload data in transit. Uthernet packets can be transferred without inter frame gap. Hence the total Uthernet packet overhead is 4 words. During inter frame gaps the data is undefined, but typically the same as the last valid data.

Together the PRE word and SFD word reliably mark start of the Uthernet packet. The TLEN field allows reliable determination of the end of the Uthernet packet. The CRC word allows detection of payload errors at the receiving side and implicitly also of TLEN errors. By avoiding PRE and SFD during the inter frame gaps the chance of falsely detecting a start of Uthernet packet is minimized. However it is allowed to fill the inter frame gaps with any word.

### 2.1 Data width

The Uthernet data width is arbitrary and can have any width  $w \geq 1$ . The data bits are indexed as  $[w-1:0]$  and the MSbit  $[w-1]$  is transferred first. In case the data word contains multiple bytes then the data word is transferred as big-endian, so MSByte first.

### 2.2 Preamble word

The preamble (PRE) word is 0xA...AAA, so for 1 bit data it becomes 0x0, for 2 bit or 3 bit data it becomes 0x2, for 4 bit data it becomes 0xA, etc.

### 2.3 SFD word

The SFD (start of frame delimiter) word is 0xA...AAB, so it is the same as the PRE word except for the LSBit [0] that is '1'.

### 2.4 Type/Length word

The TLEN word indicates the number of data words in the payload data. For TLEN smaller then a certain type offset value the TLEN is interpreted directly as length value, else the TLEN field is interpreted as type value that refers to a known length. Per TLEN type there can only be one payload length.

TLEN is 0 is reserved for future use. It may be used in future for flow control at Uthernet level.

### 2.5 Payload

The payload data contain the user data that is carried by the Uthernet packet.

## 2.6 CRC

The CRC field is a cyclic redundancy check which enables detection of corrupted data within the payload data. The Uthernet packet can use four different CRC widths (8, 16, 32 or 64 bit) listed in Table 1.

CRC	Reference [2]	Polynomial
8	ATM HEC	0 1 2 8
16	USB, Modbus	0 2 15 16
32	Ethernet	0 1 2 4 5 7 8 10 11 12 16 22 23 26 32
64	EMCA-182	0 1 4 7 9 10 12 13 17 19 21 22 23 24 27 29 31 32 33 35 37 38 39 40 45 46 47 52 53 54 55 57 62 64

**Table 1: Supported CRC polynomials**

The CRC calculation is initialized to all '1'-s. Table 2 clarifies how the input data width  $w$  is used to select the CRC polynomial and to derive the internal CRC width  $crc\_w$ .

Data width $w$	CRC polynomial	$crc\_w$	Comment
0	-	-	Illegal data width
1 : 8	8	8	Resize data to 8
9 : 16	16	16	Resize data to 16
17 : 32	32	32	Resize data to 32
> 32	64	64 * N	Resize data to 64 * N and apply CRC-64 in parallel to each of the N sections of 64 data bits

**Table 2: CRC input data width**

The data width  $w$  is always  $\leq$  the internal CRC width  $crc\_w$ . For  $w < crc\_w$  only the  $w$  LSBits of the internal CRC will get transferred into the CRC data field.

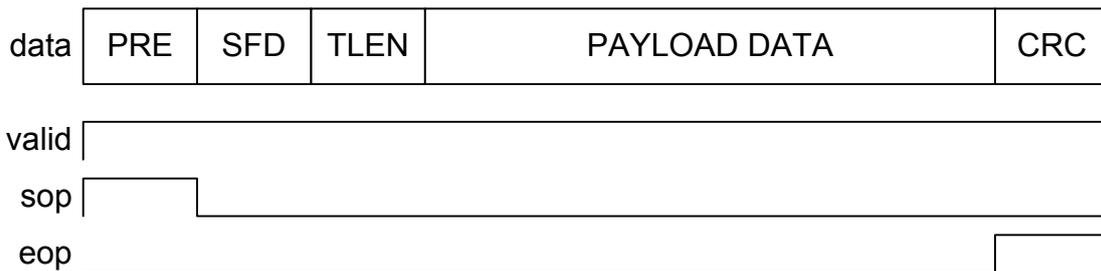
## 3 Transporting Uthernet packets

The Uthernet packet data as defined in Figure 1 can be directly transported on a link. The link can be any physical link, e.g. a connection via a gigabit transceiver, via some parallel LVDS bus, but also e.g. to an external DDR3 memory.

The data width of the Uthernet packet can be chosen such that it fits best with the PHY interface of the link. Typical data widths  $w$  are  $\geq 8$  bit. Note that for small data widths (e.g.  $w < 8$ ) it can take some time until the UTH receiver will find the true PRE and SFD to achieve proper packet alignment, but once it has it will maintain the packet alignment, because the TLEN all represent known lengths.

The TLEN type offset value and the length per supported TLEN type need to be chosen dependent on the application of the point to point link. Note that for  $w=1$  only TLEN = 1 can be used. Hence with TLEN type offset  $> 1$  the payload then has a fixed length of 1 bit, so not very useful, but with TLEN type offset = 1 the payload can then have an arbitrary but fixed length.

For internal handling of the Uthernet packets it is useful to also define the valid, start of packet (sop) and end of packet (eop) signals [6] as shown in Figure 2. Default the Uthernet packet must not have not-valid data words between the sop and the eop during the packet. However if the link interface does support the valid signal (see Appendix 4) then the Uthernet packet is allowed to have not-valid data words between the sop and the eop during the packet, because these not-valid data words can then be recognized as such at the receiving end.



**Figure 2: Uthernet packet with optional control signals**

## 4 Appendix: Standard serial data transmission protocols

This appendix provides some back ground information on standard serial data transmission protocols that are relevant for understanding the scope of the Uthernet protocol. Many high speed serial data transmission standards utilize 8B/10B coding (e.g. XAUI, 10GBASE-CX4) to ensure sufficient data transitions on the line and to support transport of control signals like valid, sop, eop and comma for aligning data transmission over multiple lanes. Table 3 list some typical serial data link-layer terminology.

Term	Description
Lane	A set of differential pairs, one pair for transmission and one pair for reception
Bonding	Data payload mapping across multiple lanes which take place at the transmitter side
Link	Communications path between two components, a link is composed of $N$ lanes
Ordered set	Symbol that is transmitted simultaneously on all lanes at once
PDU	Protocol Data Unit (payload)
Symbol	Symbolic representation of a specific code group or sequence notated with a letter or letters
XAUI	Ten gigabit Attachment Unit Interface

**Table 3: Serial data link layer terminology [3, 4]**

The 8B/10B coding is typically used with transceivers that can run at 3.125 Gbps, using 4 lane bonding this provides 10 Gbps effectively (excluding the protocol overhead due to the control symbols). For transceivers that can run at more than 10.3125 Gbps typically 64B/66B coding is used to reduce the coding overhead (e.g. for 1 lane 10GbE). The 64B/66B coding uses a quite different scheme than the 8B/10B coding, but its purpose is similar. Therefore here only 8B/10B coding is addressed further.

The 8B/10B decoder can detect single bit line errors and it signals an error when it receives an invalid code. Apart from coding the 256 possible data bytes 'D' the 8B/10B coding can code 12 special code bytes 'K'. These 'K'-code bytes are listed in Table 4. The comma characters /K28.1, /K28.5 and /K28.7 are the only 10 bit words that have 5 consecutive '1's or '0's. A link layer symbol consists off one or more 'K' bytes possibly in combination with 'D' bytes.

Kx.y	8 bit binary
K28.0	000_11100
K28.1	001_11100
K28.2	010_11100
K28.3	011_11100
K28.4	100_11100
K28.5	101_11100
K28.6	110_11100
K28.7	111_11100
K23.7	111_10111
K27.7	111_11011
K29.7	111_11101
K30.7	111_11110

**Table 4: The 12 special K-codes for 8B/10B coding**

The Altera 8B/10B coder component and Xilinx 8B/10B coder component expect the data bytes 'D' and control bytes 'K' in series whereby a control signal indicates whether the data is byte represents user data or a control byte. Hence the user must prepend the sop and append the eop as a control symbol in the stream.

Instead of using an 8B/10B coder directly, the Altera SerialLite II protocol and Xilinx Aurora protocol offer a complete serial data transmission protocol that is similar to XAUI but suitable for an arbitrary number of lanes

and at all lane speeds that are supported by the transceiver technology. The Xilinx Aurora protocol can also use 64B/66B coding. Both protocols define packet encapsulation at the link layer and data encoding at the physical layer. For example with the 8B/10B coder the Aurora frame uses /K28.2/K27.7 as sop and /K29.7/K30.7 for the eop. The Aurora frame may include embedded idles from the ordered sets of comma /K28.5, skip /K28.0 and lane bonding /K28.3 symbols. The SerialLite II protocol uses /K28.2/Dx.y as sop and /K29.7/Dx.y as eop, whereby the 'D' byte indicates a channel number. Both protocols support link initialization using a training sequence, lane alignment (in case of bonded lanes), +-100ppm clock difference compensation by inserting idle symbols and flow control.

The CRC16 and CRC32 from Table 2 are optionally also used by SerialLite II. According to [3] the CRC16 polynomial provides excellent protection for packets smaller than about 1 kByte. The CRC32 should only be used for longer packets, because it takes significant extra resources.