Teratronik
elektronische systeme gmbh

BiS

Date	Author	Changes
2008-06-26	O. Wölfelschneider, Teratronik GmbH	LTD16 included. Placeholder for AES.
2008-11-04	O. Wölfelschneider, Teratronik GmbH	AES mode explained.

Date: 11.11.2008

1. Design

BiS was designed as a Query-Response protocol. One device sends query-frames to another and will always get a response frame to each query.

For two devices on a point-to-point link, each side can send its own queries to the peer whenever it sees fit.

If the devices are on a shared medium (e.g. a RS485 network), a designated Master node is the only one who may send queries. A slave may only send data if it has received a query from the master.

The protocol is set up as a layered stack.

2. Transport Layer

START	PID	SEQ	DST	SRC	TLData	CRCH	CRCL	END
0x91 or **0x92**								**0x93**

The maximum size for TLData is 0x505 (1285) Bytes.

Escaping

To allow for binary data transport, the following substitution is used:

0x91 ➔ 0x94 0xD1
0x92 ➔ 0x94 0xD2
0x93 ➔ 0x94 0xD3
0x94 ➔ 0x94 0xD4

In other words, each control character is prefixed by 0x94 and then transmitted xor 0x40.

The CRC calculation is always done on the data without any escaping.

Field descriptions

START	Start sync byte, either 0x91 or 0x92, see below
PID	Protocol ID byte, see below
SEQ	Sequence number, must be incremented by one on each new frame. Used to detect a retransmission of the last frame.
DST/SRC	Destination and source address. Presence and size depend on PID, see below.
TLData	Payload data.
CRCH/CRCL	CRC includes everything but **START**/**END** and done with the escaping not present. Described on Page 13.
END	End sync byte, always 0x93

Description of Start Character

This is a Query-Response protocol. One end sends a **QUERY** to the other, while the other must reply with a **RESPONSE**. The start character for a **QUERY** frame is 0x91, while a **RESPONSE** starts with 0x92.

Description of Protocol ID

7	6	5	4	3	2	1	0
PType						**AMode**	

AMode	Bits 1..0	0 = no addressing (Non-networked connection) 1 - 8 bit address 2 - 16 bit address 3 - 32 bit address (RFU)
PType	Bits 7..2	Protocol type (0x00..0x3F) 0x00 = PAC data payload format 0x01 = LTD data payload format 0x02 = TEA encrypted 0x03 = AES encrypted 0x21 = LTD16 data payload format 0x22 = MTD16 data payload format ... 0x3C..0x3F = User defined

Broadcast

If the Destination address is all-ones (0xFF, 0xFFFF or 0xFFFFFFFF, depending on address size), the data is a broadcast packet. It has to be accepted by all nodes on a network, but no answer is to be sent.

Note that this is different from the no-addres mode, which can only be used on a point-to-point link and requires an answer from the peer.

Query-Response principle

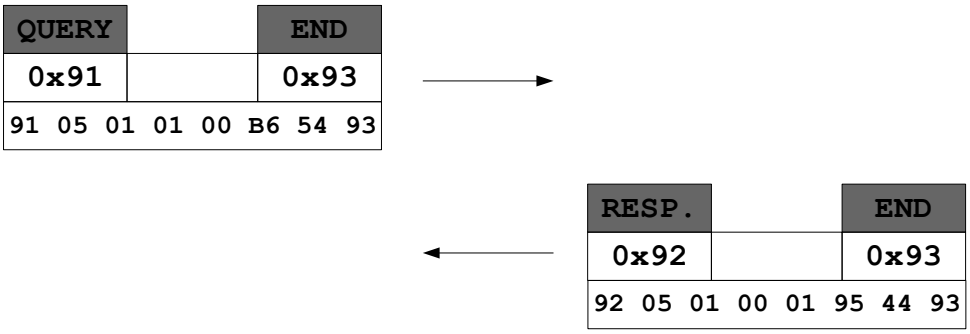
The protocol is based on the query-response principle. One communication device sends a **QUERY** type frame (starting with 0x91), while the peer must respond with a **RESPONSE** type frame (starting with 0x92) within a certain time.

If no **RESPONSE** is received after a certain timeout, the sender of the **QUERY** may repeat the last frame.

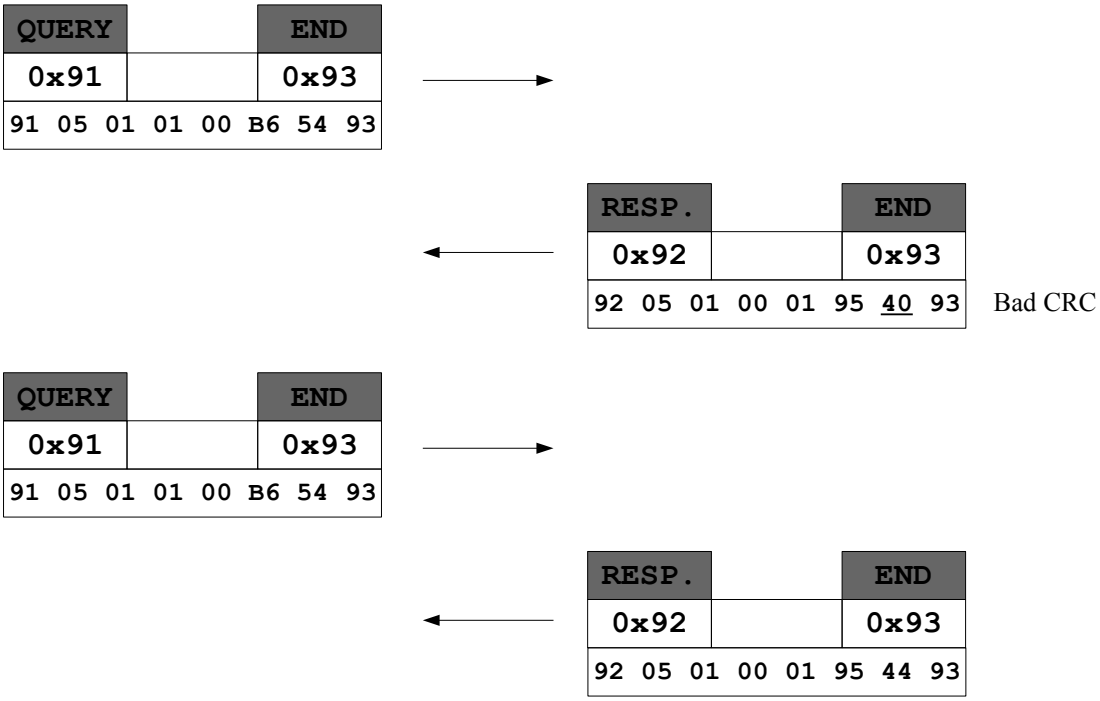
By using sequence numbers, double transmission can be handled properly. (See topic „Sequence Number Logic“ below.)

The timeout for retransmission may be configurable, as well as the number of retransmissions before giving up.

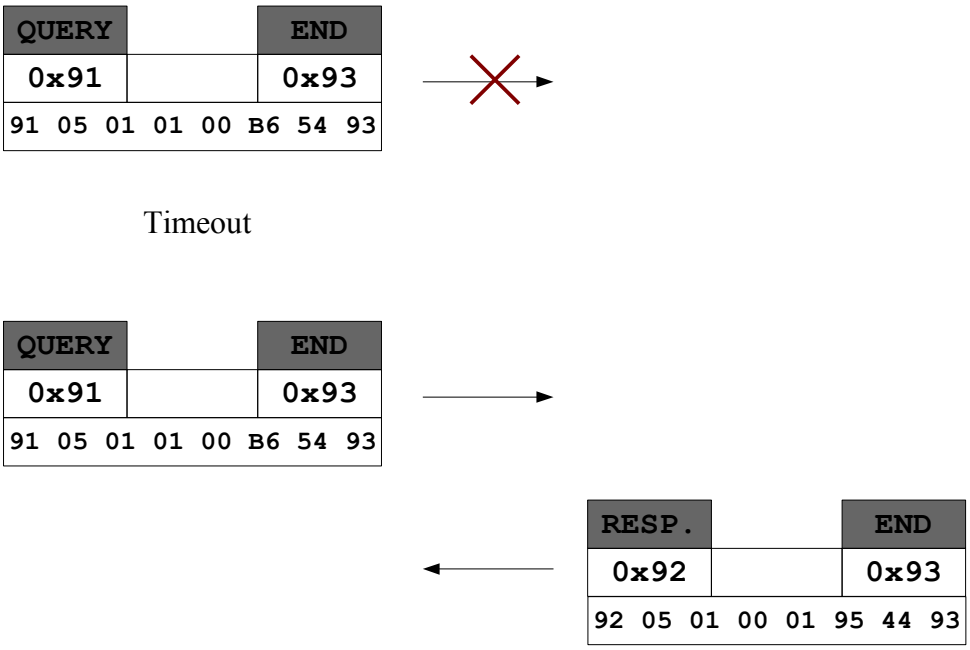
Example for a query-response communication:



Example for a query-response communication with transmission error:



Example for a query-response communication with transmission error:



Bus Masters and query chaining

On a shared network (e.g. RS485) there has to be a Bus Master that has the sole right to use the media at any time. All other nodes on the network may only transmit when receiving a **QUERY** from the master.

A slave may chain a **QUERY** to the end of a **RESPONSE** it is sending to the master. This is done by not sending the **END** character at the end of the slave's response, but attaching a complete **QUERY** frame instead.

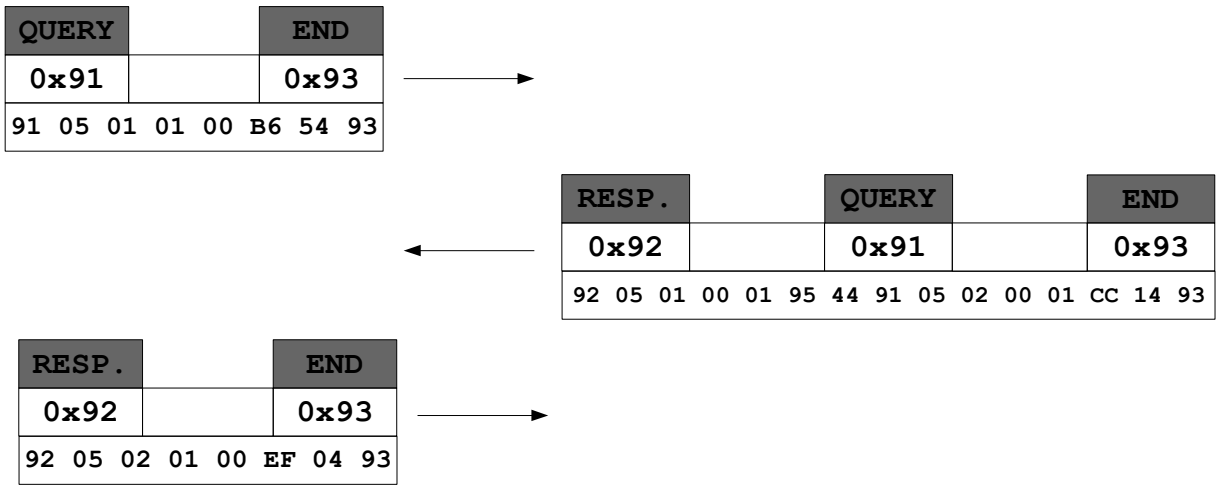
(If the slave would send two separate frames instead, a race condition can occur in the space between the END and next START that is sent by the slave. In that space, the master would not know that the slave wants to continue sending.)

The master must poll each slave on the network with **QUERY** frames so each slave has a chance to do its own queries. The master may use empty **QUERY** frames (with no data payload) for this purpose.

If a master receives a chained **QUERY** from a slave, it also must reply with a **RESPONSE** frame.

Note that a slave cannot easily repeat a **QUERY** if it fails to receive the **RESPONSE**; in that case, the slave must wait for another poll from the master so it can repeat its **QUERY**. The slave has to use a timeout that is long enough.

Example for a chained communication:



Sequence number logic

For each ***new*** request a device sends to another, it must use a different sequence number than for the last frame. It is recommended to increment the sequence number on each go.

The peer must use the same same **`SQN`** in it's reply.

On a communication error, a device may repeat the last frame completely. The repetition must be identical to the last frame, so the peer may detect a retransmission.

General case:

> If a device sees the same complete request frame (from **`START`** to **`END`** and inbetween) again, it must repeat the last reply to that frame, without re-processing the frame.
>
> (Easily done on small controllers, as the controller walks the receive buffer with a pointer on each reception anyway. Doing an additional compare on each byte is no big deal.)

Simple case:

> On small devices, where commands do not have side effects, the device may just repeatedly execute the command without regard of the **`SQN`**.
>
> (A command without a side effect is a command where it does not make a difference if it is executed once or several times.)

Examples:

If a switch device sees the frame "Turn Relay 0 on" several times, it makes no difference. On is on... so this device can go for the simple case.

If a payment device sees the frame "Pay out 42,- EUR" several times, it must make sure to do it only once or money is lost. This device must check the **`SQN`**.

3. Encryption

TEA encryption -- PType = 0x02

TEA is a symmetric cipher using 128-bit keys and 64-bit data blocks.

Encryption uses a padded CBC mode (Cipher block chaining).

(See http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

During mutual authentication, both peers agree on a session key. The authentication mechanism is not the scope of this documentation.

The 8-byte initialization vector (IV) for CBC starts at a value determined during authentication and is kept free running afterwards. This will hinder replay-attacks.

Note that both peers must keep track of two IVs, one for Tx and one for Rx. Each IV is eight bytes.

The plaintext is padded up to an eight byte boundary, adhering to the following scheme:

PID	**PAYLOAD**	**XX...XX**	**PADCNT**	**CRCH**	**CRCL**

PID	PID describing what‘s inside the payload data. Same values as used in transport layer. Setting PType to an encryption mode recursively is not defined. Bits 0..1 are always zero, as there is no further addressing added.
PAYLOAD	The plaintext data.
XX...XX	As many padding bytes as necessary to make the whole block size a multiple of 8. The padding bytes could be filled with random data or with a copy of **PADCNT**.
PADCNT	One byte pad count (Number of padding bytes present.)
CRCH/CRCL	16-bit CRC over everything, from **PType** to **CRCL**. Described on Page 13.

The ciphertext is placed in the **TLData** field of the Transport Layer.

AES encryption -- PType = 0x03

AES is a symmetric cipher using 128, 192 or 256-bit keys and 128-bit data blocks. The key size has no influence on the data encapsulation described here.

Encryption uses a padded CBC mode (Cipher block chaining).

(See http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

During mutual authentication, both peers agree on a session key. The authentication mechanism is not the scope of this documentation.

The 16-byte initialization vector (IV) for CBC starts at a value determined during authentication and is kept free running afterwards. This will hinder replay-attacks.

Note that both peers must keep track of two IVs, one for Tx and one for Rx. Each IV is sixteen bytes.

The plaintext is padded up to a sixteen byte boundary, adhering to the following scheme:

PID	**PAYLOAD**	**PADCNT...PADCNT**	**CRCH**	**CRCL**

PID	PID describing what‘s inside the payload data. Same values as used in transport layer. Setting PType to an encryption mode recursively is not defined. Bits 0..1 are always zero, as there is no further addressing added.
PAYLOAD	The plaintext data.
PADCNT	Pad count: This byte is repeated **PADCNT** times.
CRCH/CRCL	16-bit CRC over everything, from **PType** to **CRCL**. Described on Page 13.

The ciphertext is placed in the **TLData** field of the Transport Layer.

4. Data payload layer

For unencrypted transport, the data payload is simply the contents of the **`TLData`** field of the Transport layer.

In encrypted modes, the data payload is placed into the **`PAYLOAD`** field of the ciphertext container.

PAC (PlainAsciiCommand) payload format -- PType = 0x00

PAC (PlainAsciiCommand) payload format

Your generic ASCII command encoding, e.g. "cread 24". Several commands may be put into one frame by chaining them with CR, LF or CRLF.

On answer, a command may reply with several CRLF terminated lines of text. The end of the response data is marked with an EOF byte (Code 26).

If several commands are sent in one frame, the answers are also replied within one frame, separated by EOF characters.

Note that the transport layer implicitly makes sure that a replay of the last command frame will respond with a replay of the last response frame, without actually executing the command again.

LTD (LengthTagData) payload format -- PType = 0x01

In this format, the data consists of zero or more tagged data blocks:

LEN	**TAG1**	**(TAG2)**	**Data**	more...
\----------/ = LEN				

LEN	Length of data block. Length counts the bytes following it, from **TAG1** to the last **Data** byte. If the length byte is zero, it is interpreted to be 256.
TAG1	Command group tag.
(TAG2)	Command tag. (Optional, depends on **TAG1**)
Data	Optional data bytes, may be empty.

Binary data is always transferred low byte first.

LTD16 (LengthTagData16) payload format -- PType = 0x21

This format is almost identical to LTD, but using 16-bit values for Length and Tag items. As always, the least significant byte is transmitted first.

LEN	**TAG1**	**(TAG2)**	**Data**	more...
\----------/ = LEN				

LEN	Length of data block. Length counts the bytes following it, from **TAG1** to the last **Data** byte. A length value less than two is not valid.
TAG1	Command group tag.
(TAG2)	Command tag. (Optional, depends on **TAG1**)
Data	Optional data bytes, may be empty.

Binary data is always transferred low byte first.

5. CRC calculation

BiS uses a 16bit CRC in several places. The CRC is calculated according to CRC16-CCITT.

The start value is 0xFFFF. The CRC is always transmitted high byte first. The CRC is thus the only place in the BiS protocol where high-byte-first transmission is used.

On transmit, <CRCH><CRCL> are assumed as zero for the CRC calculation, thus two zero bytes are always appended to the CRC calculation.

On receive, <CRCH><CRCL> are fed through the CRC as any other bytes, the result will be zero if the calculation was correct. (This is the reason for using high-byte-first mode for the CRC).

Note that for the outer framing, the CRC calculation is always done on the data without any escaping.

6. Debugging / Out of band data

Since the sequence 0x94 0x94 is not valid in BiS, it can be used for debugging purposes. Some implementations send debug data by prefixing each debug character with 0x94 0x94.

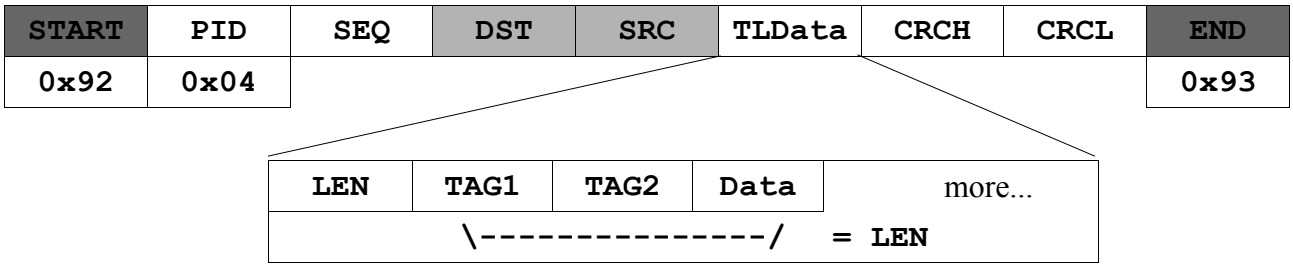
This will not disturb a transmission if any character following 0x94 0x94 is thrown away (Or displayed to a debug window) by the receiver.

This is a rather slow channel, but might prove helpful anyway.

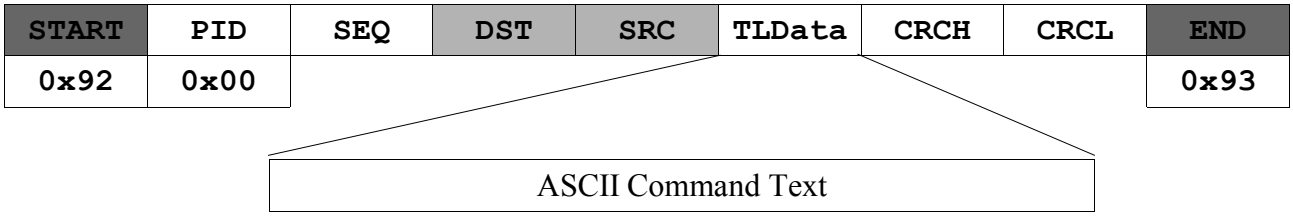
7. Layering

Following are diagrams of the documented network layer structures. In the examples, the values for **PID** are given with the **Amode** field set to zero. Also, the slave side start character, 0x92, is shown.

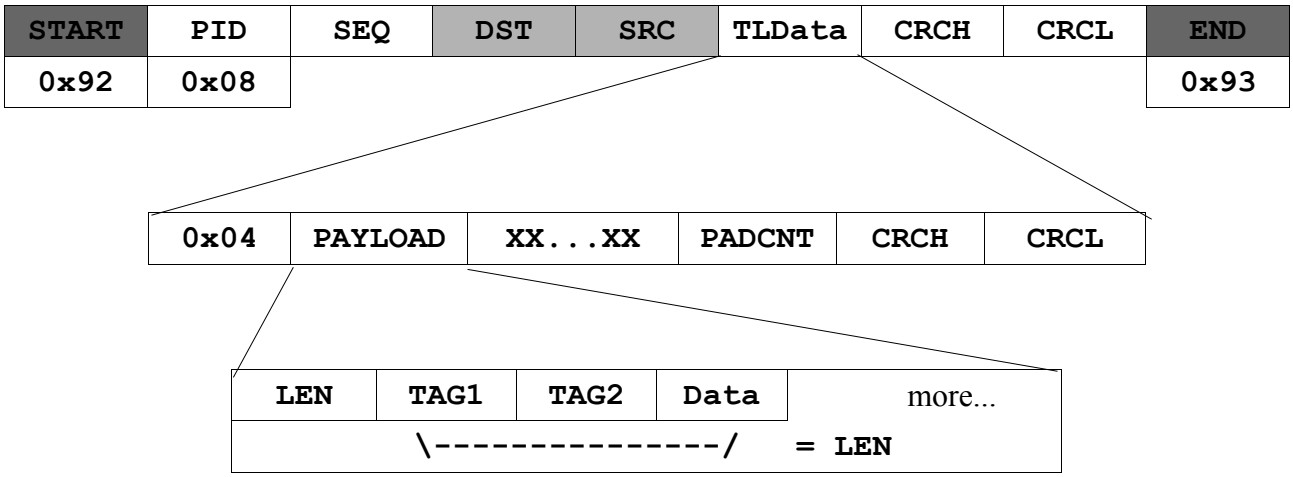
BiS-LTD



BiS-PAC



BiS-TEA-LTD



BiS-TEA-PAC