

# SIMPLE FIELD BUS PROTOCOL

COPYRIGHT ©2003 SOFT&MEDIA, diG.C. - HEXEL Electronic lab.

Version: 1.0.0 - September 2003  
authors: Claudio Ghiotto, Paolo Marchetto

SFBP - rev. 1.0.0, Sep. 2003

written by Claudio Ghiotto

## Introduction

The Simple Field Bus Protocol was designed for communications among devices sharing a single field bus or related field buses. This protocol was also designed to be simple to implement and compact to be suitable for small embedded systems. Furthermore the authors believes that this protocol should be made available to everyone is interested either for implementation or simply for study, under the General Public License (see <http://www.gnu.org> foundation for more info); for more details on copyright statements, please read the copyright section below.

The purpose design of the capability required may be summarized as follow:

- reach a reasonable amount of devices;
- be able to address all devices at a time;
- let both asynchronous, unconnected communications (datagrams) and synchronous one (connected);
- be light for small amount of data;
- give a mechanism to let carrying fragmented stream of data using synchrnous packets;
- give the chance of multiple communications (master to master configuration);
- be fault tolerant in case of collisions;
- deal with some method to prevent at best the collisions;
- have some ability to recover failures both for packets and per each frame sent;
- detect as better as possible, with compromise of compactness, for line defaults;
- work within the range of 8 bit of data per frame (to be able to run over a RS232 for point to point links, for instance).

Furthermore some assertions was made, so that the protocol will rely on a field bus line capable to detect line faults, including collisions. Expansions can be done by adding repeaters or, better, switches; however this last topic is out of the purposes of this document.

Even though the protocol was designed having in mind the IEEE RS 485 as physical layer, efforts was spent to render the SFPB packets to be carried over others physical layers.

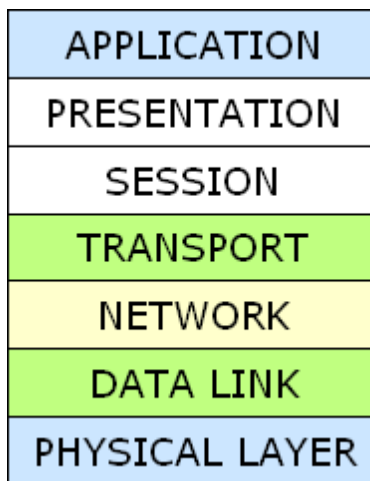
All these considerations led to a protocol that is outlined by the following features:

- up to 127 devices addressable
- 1 broadcast address
- unconnected datagram and connected packets for either fast messages or safe data transfer
- fragment bit field for stream of data bigger than a DU (DU, in this document stand for the payload Data Unit)
- compact paket headers made up of three bytes
- DU of 6 bytes
- packets made up of 11 bytes
- three type of packets: data, control and system
- CSMA/CD like collision detection
- special date-time packets
- simple packet acknowledgment system made up of a single datagram packet
- frames made up of 8 bit data, 1 stop bit, 1 parity bit, even parity checking always on
- optional failure detection at frame level and at packet level (for bus and network, avoidable on point to point links)

While we designed the SFBP, we thought about the TCP/IP protocol taking from it a guideline. So that we have two main types of packets: connected and unconnected datagrams.

## The ISO-OSI Model and SFBP

### The ISO-OSI model



**APPLICATION:** Provides network services at the application level.

**PRESENTATION:** External data representation (i.e.: byte ordering (coding), compression/decompression, protection...)

**SESSION:** Ensures data exchange between applications, e.g. Synchronization.

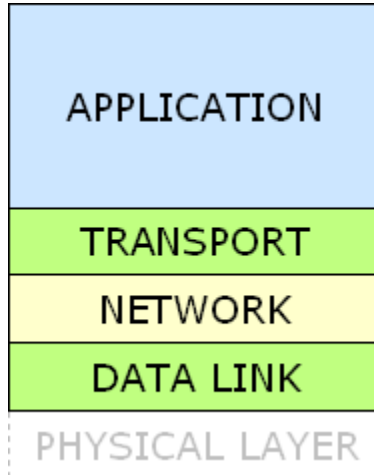
**TRANSPORT:** The transport layer is between the layers oriented to data processing (the upper ones) and the layers oriented to communication (the lower ones). Long messages are divided into packets on the sender side and they are collected on the receiver side.

**NETWORK:** Network layer is responsible of routing packets, higher layers are independent of routing.

**DATA LINK:** Data link layer controls media access (MAC: Media Access Control), data transfer, error detection and packet format.

**PHYSICAL:** The physical layer specifies the physical characteristics of the link therefore the type of the medium; such as voltage coding and modulation, for electric media, or light spectrum and intensity, for optical media.

## The SFBP model against the ISO-OSI model



**APPLICATION:** SFBP partially implements the application layer by the two packet types “Control packet” and “Data packet”. In addition it does not defines the Presentation nor the Session layers that are implicitly left to the Application.

The Type of packet bits of the Packet Information section are related to the Application layer level, and in addition the NEXT bit of the Packet Information section may play a role also at the Session layer level. Byte ordering is left to the application, it simply defines the BIG ENDIAN order, where the most significative byte is the last one.

**TRANSPORT:** The ACK packets allow connected packets and mechanism for retransmission on fault of connected packets. The NEXT bit gives a rough mechanism to deal with large messages. However a full implementation of transmission of large messages is left at the application level.

**NETWORK:** ACK packets and System packets works at Network layer level, infact they can be used also for routing purposes.

**DATA LINK:** This layer is covered by packets format and the CSMA/CD similar mechanism and the timing rules.

**PHYSICAL:** SFBP does not specify the physical layer, even though it was designed on top of a IEEE RS 485, it can run over any other physical layer such as IEEE RS 232, optic, radio and so forth.

# Description

## Simple Field Bus Protocol packet

SFBP packets are made up of 11 bytes streamed in frames as the following description:

**SM DA SA PI DU DU DU DU DU CS [ t ]**

SM	start marker	1 byte	value: 254
DA	destination address	1 byte	value: 0...127 (0x00...0x7f)
SA	sender address	1 byte	value: 0...127 (0x00...0x7f)
PI	packet information	1 byte:	

7 6 5 4 3 2 1 0  
L L L A N T T T

L DU valid length, this field is 3 bits long, defines how long bytes are valid in the payload data unit

A ACK bit field \*

N NEXT bit field \*

T Type of packet, this field is 3 bits long, defines the type of packet:

- 0 echo
- 1 control packet
- 2 data packet
- 3 time set
- 4 time get
- 5 reserved
- 6 system packet

system packets (with L set to any non-zero):

7 6 5  
L1 - L2 - L3

T = 6

L1 1 if checksum (DA+SA+(PI & 0x7f) is odd, 0 if checksum is even

L2, L3:

- 0 forbidden (never used)
- 1 Reset, device must reset communication subsystem
- 2 reserved
- 3 reserved

Remarks:

System packets never carries the DU, so they behave as ACK packets which are limited to the first 4 bytes (SM DA SA PI)

this field must never be combined with any non-zero DU valid length to prevent be confused with STARTMARKER.

7 reserved

\* the combination of ACK and NEXT bit fields set to one mean a datagram packet, only connected packets may use NEXT.

NEXT is set when packet is a fragment in a stream of data

ACK is set when packet is sent as acknowledgment (see ACK Packets below)

DU	payload data unit	6 bytes	invalid bytes should be zero padded to the DU length (6 bytes)
CS	check sum	1 byte	value: cyclic sum of all bytes but the SM
[ t ]	delay time of 3 frames with, rounded in excess		

## ACK packet (acknowledge packet)

ACK packet are made up of 4 bytes as the followin description:

### **SM DA SA PI [ t ]**

SM, DA, SA and PI are the same as in the normal packet but PI which contains the ACK bit field set, and the DU valid length set to a checksum of the whole ack packet but the SM byte and the LLL bit fields of PI.

PI of a ACK packet:

L	check sum parity checker: set if sum of DA+SA+(PI & 0x7f) is odd
L	zero
L	zero
A	ack bit set
N	next bit, zero
T	type set to 0

## System packets

System packets are made up of 4 bytes, they are sent for urgent system reasons, they must be processed even while waiting for NEXT packet in a stream of data.

### **SM DA SA PI [ t ]**

SM, DA, SA and PI are the same as in the normal packet but PI contains both ACK and NEXT bit fields set (which mean datagram packet), the MSB leads the parity checker and the lower two L bit fields leads the type of system packet (zero is forbidden) and the T field contains the value 6.

System packets may be sent via broadcast or directed to a targeted address.

PI of a system packet:

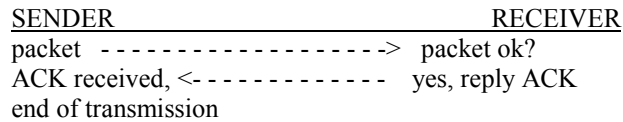
L	check sum parity checker: set if sum of DA+SA+(PI & 0x7f) is odd
L, L	system statement, zero is forbidden, other possible values: 1 reset communication subsystem 2 reserved 3 reserved
A	ack bit set
N	next bit set
T	type set to 6

## Connected packets

Connected packets have the ACK bit field not set and NEXT bit field set if packet is part of a fragmentation of a stream of data and a next packet should be expected.

Each connected packet is sent to the remote peer, then the sender set up a timeout and waits until the receiver will return an ACK packet (see ACK packets) or timeout.

Because this protocol rely on information about the lower level of the OSI model, the physical line status, detecting collisions and faults, to keep it simple no further reply is expected after an ACK packet and more the sender will stop immediately if a collision or fault is detected awaiting for a time that is increased by each fault attempt by a random constant set for each device before to attempt again the transmission; this up to the retransmit limit is reached. (see diagram). If the acknowledge is not received within the timeout, then packet is intended as not delivered and a new attempt will be done, up to the retransmit limit is reached.



Fragmented packet of a data stream.

Because other devices may attempt to send packet to a recipient peer; while receiving packet with NEXT bit set, the receiver will set up a status to do not accept packet from any sender but the original who sent packet with NEXT bit set, this will be till a packet with NEXT bit is not set or receiver times out.

## Unconnected Datagrams

Unconnected datagram packets have both the ACK and NEXT bit fields set. They are sent over the network without expecting any reply by the remote peer. This type of packet are not whole unsafe because the device can get information about the state of the line, if any fault will occur then a new attempt is made up to the sendretry limit is reached.

Unconnected datagrams are useful for fast data transfers and pulse or events such as I/O or status notifications.

## Broadcast packets

Broadcast packets are received by all the devices on the shared line. These packets can be only datagrams (unconnected) and are addressed to the destination zero (DA = 0). None ack is expected by any peer.

Broadcast packets are useful for status notification addressed to all devices, for synchronizations, for address discovery, routing and so forth.

Because a broadcast packet is same of a regular packet but the destination address, they may lead information as control packets or data packets, more because Sender address is included in the packet it may be used to run-time notify a device added to the line.

## Address discovery using connected packets

Sending a packet to any one of the 127 possible addresses may be used in special occasions for address discovery and autodetection. Because each device who recognize itself by the destination address reply with an ACK packet which contains its address, the sender may collect addresses by all the received replies.

Of course this should be realized under a special implementation which do not attempt to send the packet on non acknowledgments.

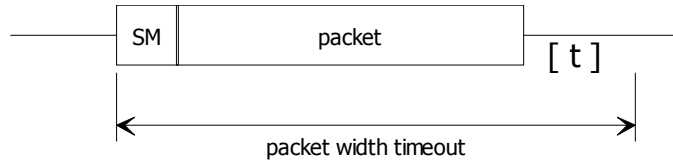
## Timeouts

Frame receive timeout is set to 2 frames, rounded in excess, this guarantee that an unsync packet will time out if a fake start marker is erroneously detected among DU data.

Packet width timeouts after all sent frames are timed out plus 3 frames time width, this is given by  $11*tf+3tf$  where  $tf$  is a frame timeout; a new packet can't be sent by any device connected to the shared line before a packet width is timed out.

## Communications among peers

Before sending a packet over the network a unit will check if last Packet Width Timeout has expired. This is done by continuously listening the network and setting up the timeout each time a start marker is detected (see diagram).



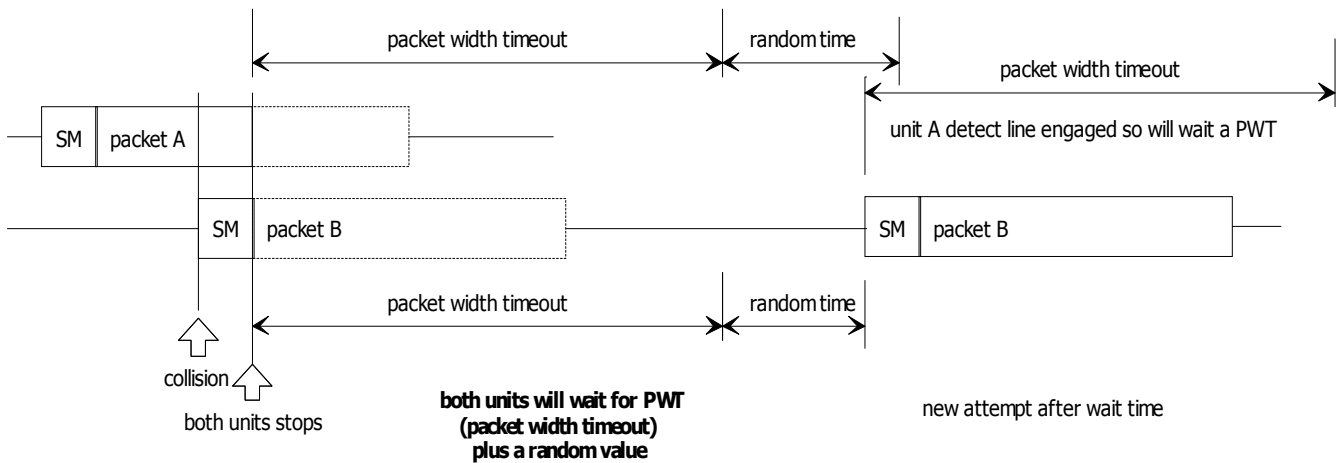
This will reduce the event of collision with other units. However two units still may attempt to send a packet over the network at the same time, in such a case a collision will occur causing a distortion to the message and in turn causing the echo of packet to be not valid.

This way a sender can detect if the line has been collided by an other packet, if this case happen the sender should stop to send frames and should wait a whole packet width time plus a random coefficient increased by each attempt done to induce a variation required to avoid synchronization between partners in attempt to send packets that would collide each time.

Because a collision may occur randomly, it could happen just in the middle of a packet sending (may be due to a new device "hot-connected"). Because receiver cannot recover this situation due to the fact that the last received frame subjected to the collision was unreliable and very likely will produce a checksum error; it is required that a sender who detect a collision immediately stops to send frames and will re-transmit the last packet later.

How long later? Below a diagram shows a possible scenario:

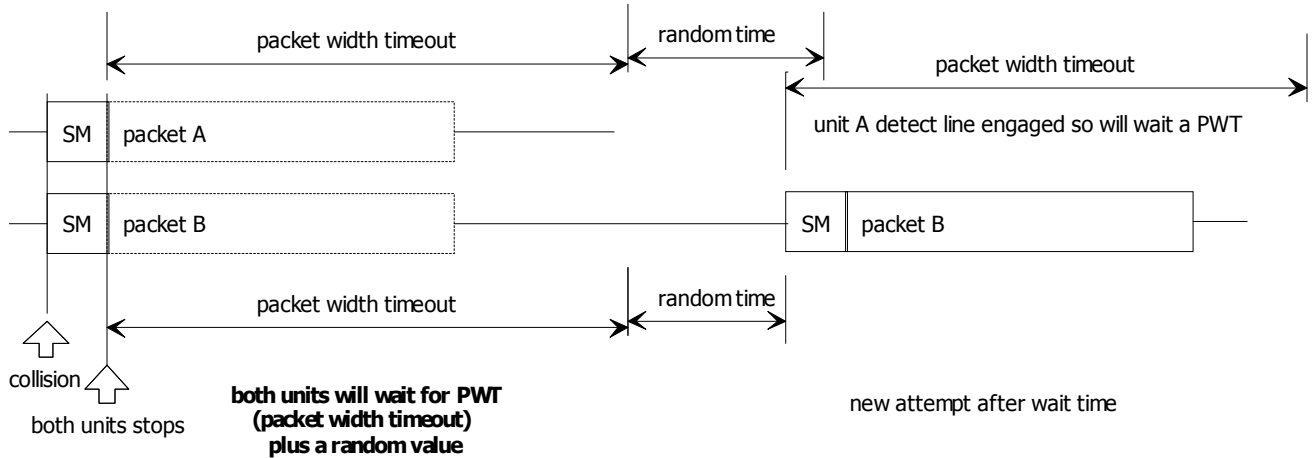
### HOW COLLISIONS ARE DEALLED



This scenario should be rare because packet B starts too late related to packet A; unit B should detect the beginning of the packet A so that the unit B should have been set a PWT.

So a more realistic scenario could be the following:

## HOW COLLISIONS ARE DEALLED case 2



In this case both A and B units attempt to send a packet because at the time they try to send they believe that the line is not engaged. This will cause a distortion of the signals on the line and both will not match the actual data on the line against the data sent falling in a collision detection.

## Checksum Algorithm

The checksum algorithm is a simple sum of all data sent but the SM (STARTMARKER) and the checksum itself. Following the algorithm for connected and unconnected packets:

$$cs = \left| \sum d \right|_{2^8}$$

where: cs is the checksum, d are the members DA, SA, PI, and the 6 DU members of the packet. Following the algorithm for System and Ack packets:

$$cs = \left| DA \right|_{2^8} + \left| SA \right|_{2^8} + \left| PI \& 2^7-1 \right|_{2^8}$$

$$PI = PI + 2^7 * (cs \& 1)$$

Checksum is the DA, SA and the PI but the MSB, the resultant cs is used to set the MSB of PI as parity bit, the receiver must match the same parity by processing the same algorithm.