

SIMPLE FIELD BUS PROTOCOL

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

Version: 2.0.0 – November 2004

Reference: SFBP v.1 by Claudio Ghiotto and Paolo Marchetto.

SFBP - rev. 2.0.0, Nov. 2004

written by Claudio Ghiotto

License

This protocol is open and is released under the terms and condition of 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.

Introduction and reference

This is a variation from the SFBP version 1, developed by Claudio Ghiotto and Paolo Marchetto on September 2003, which is a revision to implement a better checksum algorithm; in addition some errata corripge and further details has been added.

This revision has been taken to correct the too simple checksum adopted in the earlier version of this protocol. The new algorithm is much like as a CRC, with a simple mechanism which features good performances, little footprint either in memory and in the network packets, keeping it as simple as possible, and featuring a good rejection to the typical effects of noise on the intended transmission media.

Description

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 embedded systems.

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.

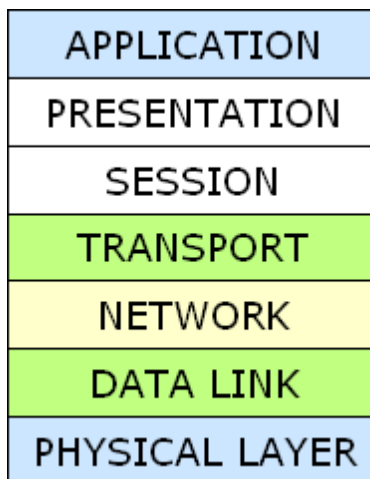
Changes still meet the original purposeses.

Characteristics

- 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: Payload Data Unit)
- compact packet 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
(made up of only 4 bytes for version 1, and 5 bytes for version 2)
- 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)

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.

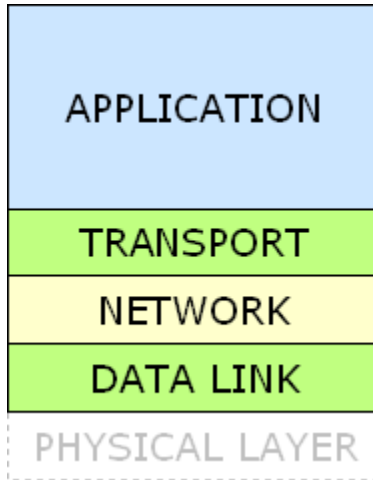
TRANSPORT: The transport layer is between the layers oriented to data processing (the upper ones) and the layers oriented to communication (the lower ones). Long messages are divided into packets on the sender side and they are collected on the receiver side.

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 significant 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.

SFBP

Simple Field Bus Protocol packet

SFBP packets are made up of 11bytes, or optionally more, streamed in frames.

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

SM	start marker	1 byte	value: fixed, 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 many bytes are valid in the 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
- 4 priority (experimental, see priority packets)
- 5 reserved
- 6 system packet

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

7 6 5
L1 - L2 - L3

T = 6

L1, L2, L3:

0	forbidden (must be 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 5 bytes (SM DA SA PI CS)

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 fixed length fragment being of a stream of data.

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

DU	data unit	6 bytes	invalid bytes should be zero padded with zeroes to the DU length (6 bytes).
CS	check sum	1 byte	value: cyclic sum of all bytes but the SM and the CS itself (see addendum Checksum algorithm)

[t] Physical layer : Delay time of 3 frames width, rounded in excess
This is not mandatory. It is suggested for physical layers such as the IEEE RS 485.

This symbol is used everywhere in this document and should be refered to this definition.

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 the CS-CD information (returned by 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 which is increased per each failed attempt multiplied 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 the packet is intended as not delivered and a new attempt will be done, up to the retransmit limit is reached.

<u>SENDER</u>		<u>RECEIVER</u>
packet	----->	packet ok?
ACK received,	<-----	yes, reply ACK
end of transmission		

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 any packet incoming from any sender but the original which sent the first packet of the stream (with the NEXT bit set), this will be true until a packet with NEXT bit set to zero is received, or receiver timeout.

Unconnected Datagrams

Unconnected datagram packets have both the ACK and NEXT bit fields set to one. 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. However should be kept in mind that datagram packets are not guaranteed to be delivered.

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 Address ID 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 the 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, as a sort of plug'n play.

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 in turn contains its address, the sender may collect addresses by all the received replies.

Address ID and MAC Address

Devices could be left with undefined address ID but marked with a MAC Address (Media Access Control Address), the address ID could then be set by a special broadcast packet carrying the new intended address ID and the MAC Address, the device which matches the MAC address could so set the given address ID.

Predictive synchronus Carrier Sense Multiple Access / Collision Detection mechanism

The SFBP provides a mechanism to optimize the multiple access to the common medium, allowing better performances, but still having the media access at a non-deterministic model.

This behaviour has been chosen because it is well suited to a fuzzy world where devices must follows events and events are not predictable by definition, so we attempt to give a smart order on random events. You can experience it by observing a highly traffic cross road: cars follows their lane however at cross border they attempt to enter in the crossing flow non stopping themselves, but reducing or increasing speed. Yes, the result seems very confused, but it should be kept in mind that drivers are humans. It is simple even though it requires to the device a small additional effort to be processed.

The mechanism is based on a time long as the time required by the packet length (thereafter called "Packet width time") plus a small amount of "hole" time. This time is synchronized each time a STARTMARKER frame is received, even if the following frame leads a mismatching address.

This technique led to a very efficient way to send random packets over heavy traffic network, limiting the collisions. Tests had proven that a normal CSMA/CD versus the PS-CSMA/CD gives a reduction of collisions of about 30% under heavy traffic network conditions while it does not show significative differences under low traffic conditions.

Because collisions downgrade the performance of a network as its traffic raise, this explain the increased performances mentioned above only at high traffic rates.

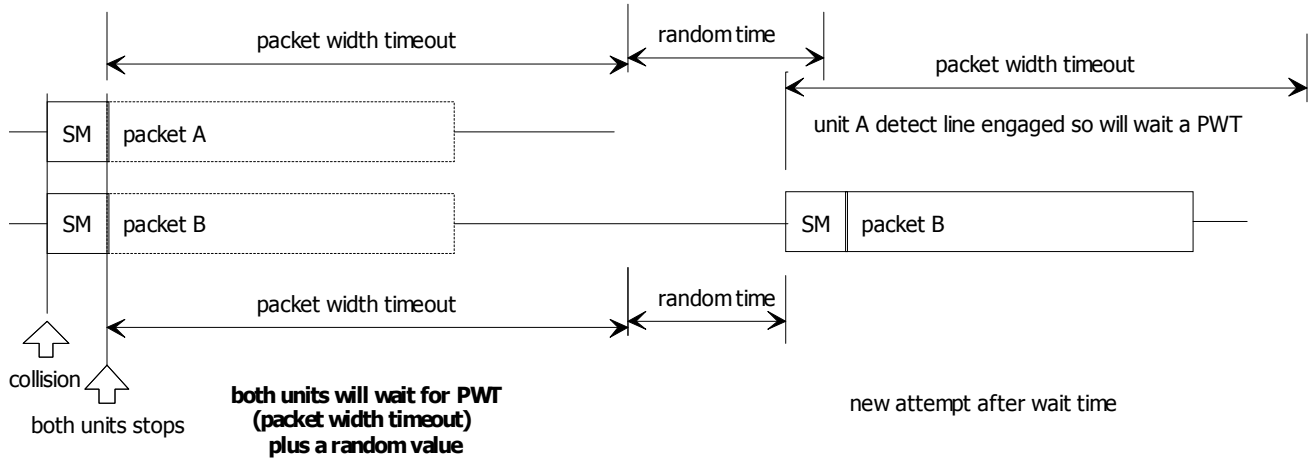
Because this is an open source protocol, you are welcome to submit more test results sending the report to info@smartcontroller.org.

Timeouts

The "Frame receive" timeout is set to 2 frames, rounded in excess. This guarantee that an un-synchronized packet will timeout if a fake start marker is erroneously detected among DU data.

"Packet width" timeouts after that all the sent frames are timed out, plus the time reuiged to send 3 frames (hole time), 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.

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.

Cyclic checksum algorithm

by Claudio Ghiotto

Lot of algorithms are available to make a cyclic redundant checksum, but almost everyone is too heavy to be implemented in small devices, so has been adopted the following algorithm which is easy and small to implement, because the small amount of operation involved is fast, and has proven to be quite strong against the typical noise met on line despite to its simplicity. The basics of its work rely on the need to process the data with a polynomial expression so that applying a noise, which offsets data, cannot mislead to a fake valid checksum result. For a shift operation actually multiply or divide the number by two, adding then the new data simply makes a polynomial, however a simple shift lead to lost the most or the least significative information, so the number is "rotated" to left by one bit, this preserve the whole information and satisfy the need to diffuse to all the expression a variation applied to one of their members, such as the case of a noise, dramatically reducing the probability of a fake "valid" checksum.

Follows a C sample which implements such algorithm:

principle:

```
unsigned char cs = 23, d;  
  
cs = (( cs << 1 ) | ( cs >> 7 )) + d;
```

full sample:

```
unsigned char computeCheckSum(unsigned char *d)  
{  
    // d      NULL terminated string of data to send  
  
    unsigned char cs = 23, i;  
    for(i = 0; d[i]; i++) doCS(&cs, d[i]);  
    return cs;  
}  
  
void inline doCS(unsigned char *cs, unsigned char d)  
{  
    *cs = ((*cs << 1) | (*cs >> 7)) + d;  
}
```

Its representation is the following:

$$cs = 23$$

$$cs = \left| \left(\left| 2cs \right|_{2^8} + cs / 2^7 \right) + d \right|_{2^8}$$

Where cs is the checksum and d is the data processed by the algorithm. The cs is initialized by a prime number (23 has been chosen) before beginning to process all data that will form the final checksum.

In System and Ack packets the members DA, SA, PI are processed to compute the checksum, which is then appended to the packet.

In Datagram and Connected packets the DA, SA, PI and the 6 DU members are processed to compute the checksum, which is then appended to the packet.