1 General

1.1 Revision Descriptions

None - first revision.

2 Physical Interface

2.1 Structure

Each SVIFT unit has two identical interface connectors, named “A” and “B”. Units are interconnected by connecting any of these (A or B) connectors to any of the connectors of the next unit - thus forming a daisy chained bus.

Such a bus segment can be connected to a supervisor or controller at the free A or B connector at any of its ends.

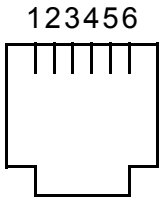
It is also possible to connect both ends of a SVIFT bus segment to independent supervisors/controllers. In this way high reliability redundant buses can be built, where no single fault can cause loss of the total functionality.

2.2 Electrical

2.2.1 Connectors and Cables

All equipment uses RJ12 jacks, connected according to the following figure:

Pin	Name	Function
1	Pow	+Power (input/output)
2	TD	Transmit Data (output)
3	Gnd	Signal Ground
4	RD	Receive Data (input)
5	-	through connected
6	-	through connected



All cables are 2*2 twisted pairs shielded 0.14 - 0.23 mm^2, using 1-2 as one pair and.3-4 as the other. All cables have pins 2 and 4 “crossed”, i.e. has pin 2 at one end connected to pin at the other 4 and vice versa. Pins 1 and 3 are always “straight” connected, i.e. matching pin numbers connected.

2.2.2 Circuit Implementation

The SVIFT bus is intended for operation in a single signal reference potential plane (SRPP). The Gnd pins of both interface connectors are therefore connected to the shield connector in each unit. The shield connector is also directly connected to the mechanical structure of the unit when its material is conductive.

All units must implement the data transfer functions of the RD and TD pins, see the “Signalling Method” chapter for further definition. They must also implement the RD pin voltage detection necessary to determine whether the interface is connected to another unit or not. See the “RD Voltage Detect” chapter.

Each unit must interconnect the Pow pins of both interfaces. It may also supply power to the Pow pin, use the Pow pin for its own supply or both. Its data sheet must specify which of these total four powering options it supports. Further specifications is given in the “Pow Pin Specification” chapter.

2.2.3 Signalling Method

Voltage Levels:	According to CCITT V28
Signalling Method:	Asynchronous Serial, full duplex
Signalling Speed:	9600 Baud
Byte Format:	1 start bit, 8 data bits, 1 stop bit. (Bit order and logical levels according to CCITT V28).
Flow Control	None

2.2.4 RD Voltage Detect

The RD pins of both interface jacks must implement circuitry to detect loss of input signal. The condition for loss of input signal is $-V_{th} < V_{RD} < V_{th}$, where V_{th} = 3 +/-0.5V. This detection is presented as two bits of a ROFLB object named IFLAGS, having bit names A_INV and B_INV.

2.2.5 Pow Pin Specifications

A unit may be able to supply power to the Pow pins of the interface connectors. If so, the voltage supplied output must be in the range 9-12V at all currents specified. This feature and much current can be supplied must be stated in its data sheet. Any unit implementing this feature must have some means for limiting the current supplied to the Pow pins to maximum I_{th}. It must also have an approved fuse with rating I_{th} series connected with this supply. The value if I_{th} is 250mA.

A unit may use the Pow pin for its power supply. If so, its data sheet must specify this feature and the worst case current consumption. In order to use this function, the unit must be able to operate at an incoming voltage at the Pow pin of minimum 6.0V.

3 Protocol

3.1 General

The SVIFT framing protocol has the following features:

- It provides two levels of message tagging - by adding “expansion bytes”. This enables a stateless design in complex systems where the SVIFT response messages need routing back to various requesting units.
- It allows out of band (or rather out of frame) signalling for other purposes, this by use of header HI-bit=0. Typically used for Test/Debug/Load terminal functionality.

The expansion bytes - for tagging - are for historic reasons named “Master” and “Star Controller” expansion bytes. These names today have no defined interpretation other than: if they are both used in a hierarchical manner, the “Master” expansion bytes should be used for the higher level.

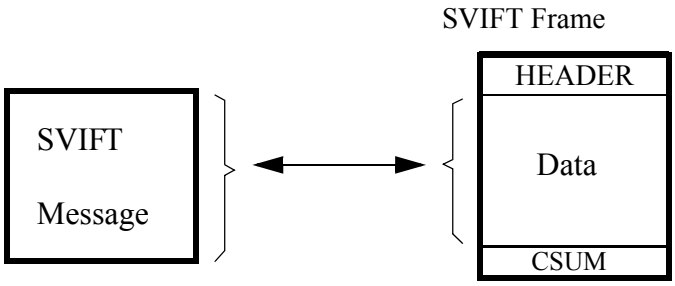
3.2 Format, size

SVIFT messages are contained in frames. These frames always uses a header field at the beginning, and a checksum byte at the end of each message.

The following limitations must be obeyed:

Maximum Frame Size:	40 bytes
Maximum Message Size:	32 bytes

The encapsulation is illustrated by the following figure:



3.2.1 The header field

3.2.1.1 Header First Byte (HFB)

The header field is minimum one byte in size and has the following bit pattern:

	7	6	5	4	3	2	1	0
HFB	HI	MT	ME	SCE	FRLEN			

The following subfields are defined in byte 1:

HI Header Indicator, defined as follows:
1 - The Frame contains a SVIFT message
0 - Test character, used for Test/Debug/Load terminal

MT Message Type:
0 - Currently unused
1 - Indicates presence of a PROT byte (always used wit SVIFT)

ME Master Expansion bit:
0 - No Master Expansion Byte is contained in this frame.
1 - One or more optional Master Expansion Byte(s) (MEB) follow. See expansion byte and header byte order descriptions below.

SCE Star Controller Expansion bit:
0 - No Star Controller Expansion Bytes is contained in this frame.
1 - One or more optional Star Controller Expansion Byte(s) (SCEB) follow. See expansion byte and header byte order descriptions below.

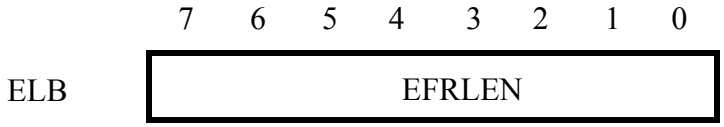
FRLEN FRame LENgth. The binary value of this subfield is the byte count, including all bytes except the HFB byte itself. The maximum value of the FRLEN subfield is 15, longer frames use FRLEN = 0 and a “extra length byte” (ELB), inserted immediately following the HFB. See ELB definition below.

A response must contain the same values of the MT-, ME- and SCE- subfields as the corresponding request.

3.2.1.2 Extra length byte (ELB)

The 5-bit FRLEN subfield of header byte 1 can only represent values up to maximum 15. Longer frames therefore use FRLEN=0 and an ELB, inserted immediately after the HFB. If any header expansion bytes (MEB(s) and/or SCEB(s)) are present, they appear after the ELB. The format of the extra length byte is as fol-

lows:



The following ELB subfield is defined:

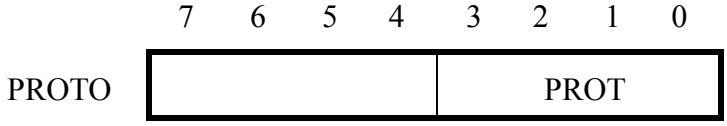
EFRLEN Extra FRame LENgth. The binary value of this subfield is the byte count of the frame, including all bytes except the HFB.

The ELB byte is only present in the frame if the length so requires.

The total size of a frame is always EFRLEN (if ELB present) + FRLEN + 1.

3.2.1.3 Header protocol byte (PROTO)

When the HFB MT-bit = 1 and also the ME-bit= 1, it indicates that the first master expansion byte is a PROTO byte. The interpretation of the PROTO byte is:

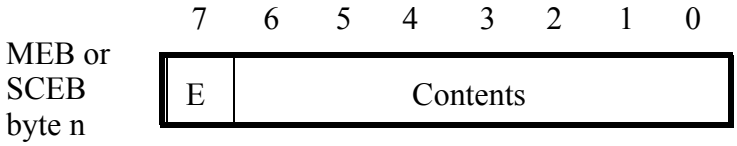


PROT PROTocol of the message. The following values are defined:
1 - The SVIFT protocol.

The PROTO byte is otherwise treated as a normal MEB - see description below.

3.2.1.4 Header expansion byte(s) (MEB and /or SCEB)

The master may also, if it so requires, add any number of optional header expansion byte(s), to the header. When adding such HEB(s) to a request, the response will also contain identical HEB(s). The format of the optional header byte is shown in the following figure:



The following subfields are defined in all expansion bytes (MEB(s) or SCEB(s):

E Expansion bit:
0 - Last header expansion byte of this kind (MEB or SCEB).
1 - One more header expansion byte of this kind (MEB or SCEB) fol-

lows.

Contents

MEB: These expansion bytes are inserted by and also interpreted only by the master. The format, size as well as coding, is “master defined”. Typical use of HEB(s) is to include application information and/or source address information.

SCEB: These expansion bytes are inserted by and also interpreted by star controllers only. They are used for information passing between star controllers. Since the format of these bytes is not defined by the protocol, the user must guarantee the same interpretation is used in all units.

Units that are not supposed to interpret the contents of each type of header extension byte (i.e. other than masters concerning MEB(s) and other than star controllers concerning SCEB(s)) may not change the presence or contents of header extension bytes. This applies to response messages, which should copy this data from the request, as well as chain controller passing of messages.

3.2.1.5 Header byte order

The following byte order is used within the header, transmitted in order top to bottom:

Type	Description	Use
HFB	Header First Byte	mandatory
ELB	Extra Length Byte	mandatory if frame size > 16
MEB	Master Extension Byte	mandatory with SVIFT
...	(more possible MEB’s)	optional
SCEB	SC Extension Byte	optional
...	(more possible SCEB’s)	optional

3.2.2 The data field

The data field contains the whole of a message.

3.2.3 The CSUM field

The one byte CSUM field contains the inverted (1’s complement) sum of all bytes of the frame, only excluding the CSUM byte itself.