FBB Forward Protocol.

----------------------

FBB software includes two forward protocoles. The first one

is standard with MBL/RLI protocole. The second one was developped

to allow a better efficiency, particularly on long links where

propagation time of data are long. The exchange of commands is

reduced to a minimum, and not acknoledged to get time. The data

transfer direction is changed every block of data, a block of data

holding up to five messages. This uses the "pipeline" effect of

long links (Nodes and digipeaters), and gain some time over short

links (HF...).

FBB protocole is very simple in its principle. It is based on

MID/BID usage. The identification is made by the F letter in the

SID (system type identifier contained in square brackets). All

command lines must start in first collumn with the 'F' character.

All command lines are ended by a return (CR) character.

Suppose I call another BBS to forward some mail. When I

connect another BBS using FBB protocole, I will receive the SID

followed by a text and the prompt (">"). If the SID contains the F

flag, I will send immediately my SID and the first proposal.

Proposals looks like :

FB P F6FBB FC1GHV FC1MVP 24657\_F6FBB 1345

F>

FB : Identifies the type of the command (proposal)

P : Type of message (P = Private, B = Bulletin).

F6FBB : Sender (from field).

FC1GHV : BBS of recipient (@field).

FC1MVP : Recipient (to field).

24657\_F6FBB : BID ou MID.

1345 : Size of message in bytes.

F> : End of proposal.

ALL the fields are necessary. This kind of command must hold

seven fields. If a field is missing upon receiving, an error

message will be send immediately followed by a disconnection.

A proposal can handle up to five FB command lines. If the

total size of messages seems to be too important, the proposal can

handle less lines. In FBB software, a parameter is defined in

INIT.SRV file to tell the maximum size of the message block. It is

set by default to 10KB for VHF use. It can be adjusted according

to the quality of the link.

Exemple of proposal :

FB P F6FBB FC1GHV.FFPC.FRA.EU FC1MVP 24657\_F6FBB 1345

FB P FC1CDC F6ABJ F6AXV 24643\_F6FBB 5346

FB B F6FBB FRA FBB 22\_456\_F6FBB 8548

F>

This proposal is limited to three FB lines, as the amount of

messages overran the 10KB limit defined for this link.

When receiving the proposal, the other BBS will reject,

accept or defer the message. This command is made by a FS line :

FS -+=

This means :

- I don't want the first message (-).

- I need the second message (+).

- I defer the third message, as I'm still receiving it.

It should interesting to defer a message if you are still

receiving it on a other channel, or if you think that the size is

to big, or for another reason. The message should be proposed

again at the next connection.

FS line MUST have as many +,-,= signs as lines in the

proposal.

When receiving the FS lines, I can send the block of

messages. Each message is made with the title on the first line,

the text, and a Ctrl Z in the last line. The is no blank line

between the messages.

Title of 2nd message

Text of 2nd message

.....

^Z

When the other BBS has received all the asked messages, it

acknoledges by sending its proposal, and the system is reversed.

If it has no message to send, it only sends a line :

FF

This line must not to be followed by a F>.

If the other hand has no message, it sends a line :

FQ

and asks for the disconnection.

Example :

---------

F6FBB FC1GHV

----------------------------------------------------------------

Connects FC1GHV

Connected

[FBB-5.11-FHM$]

Bienvenue a Poitiers, Jean-Paul.

>

[FBB-5.11-FHM$] (F6FBB has the F flag in the SID)

FB P F6FBB FC1GHV.FFPC.FRA.EU FC1MVP 24657\_F6FBB 1345

FB P FC1CDC F6ABJ F6AXV 24643\_F6FBB 5346

FB B F6FBB FRA FBB 22\_456\_F6FBB 8548

F>

FS +-+ (accepts le 1st et le 3rd).

Title 1st message

Text 1st message

......

^Z

Title 3rd message

Text 3rd message

......

^Z

FB P FC1GHV F6FBB F6FBB 2734\_FC1GHV 234

FB B FC1GHV F6FBB FC1CDC 2745\_FC1GHV 3524

F>

FS -- (Don't need them, and send immediately the proposal).

FB P FC1CDC F6ABJ F6AXV 24754\_F6FBB 345

F>

FS + (Accepts the message)

Title message

Text message

......

^Z

FF (no more message)

FB B F6FBB TEST FRA 24654\_F6FBB 145

F>

FS + (Accepts the message)

Title message

Text message

......

^Z

FF (still no message)

FQ (No more message)

Disconnection of the link.

In this example, FBB protocole is used as the two BBS were

identified by the F flag in the SID. If F6FBB had sent the SID

[FBB-5.10-MH$] when answering FC1GHV, the protocole should be the

standard MBL/RLI.

All callsigns are only examples !

Extension to the protocole. Compressed forward FBB.

---------------------------------------------------

The protocole utilized for the transfer of ascii files compressed

is an extension to the existing protocole. The compressed forward

is validated by the presence of the letter B in the SID

[FBB-5.12-BFHM$]. The transfer of compressed files can only take

place under FBB protocole. The presence of the letter B in the SID

without the F letter will remain without effect.

The only difference as regard to the standard protocol is the

submit line. It can specify the type of data contained in the

compressed message. FA means that the transfer will be an ascii

compressed message. FB means that the message will be a binary

compressed file (this last possibility is not yet implemented).

The submission of an ascii message will be in the form :

FA P FC1CDC F6ABJ F6AXV 24754\_F6FBB 345

The submission of a binary file will be in the form :

FB P FC1CDC F6ABJ F6AXV 24754\_F6FBB 345

The transfered data are of a specific format. The transfer will be

done in binary mode. This last one is derived of the YAPP protocol

which is very reliable. All transfer is made of a header, a block

of data, an end of message and a checksum. Each transfer is

equivalent to the transfer of one message of the standard protocol

and shall not be followed by a control Z, the end of file

specifier is defined in another way. Unlike YAPP transfers, there

is no acknowledgement during the transmission of messages, the

protocole is then more simple and efficient.

Format of header for an ascii compressed message (submission FA) :

<SOH> 1 byte = 01 hex

Length of the header 1 byte = Length from the title,

including the two <NUL> characters.

Title of the message 1 to 80 bytes

<NUL> 1 byte = 00 hex

Offset 1 to 6 bytes

<NUL> 1 byte = 00 hex

Format of header for a binary compressed file (submission FB) :

<SOH> 1 byte = 01 hex

Length of the header 1 byte = Length from the filename,

including the two <NUL> characters.

Name of the file 1 to 80 bytes

<NUL> 1 byte = 00 hex

Offset 1 to 6 bytes

<NUL> 1 byte = 00 hex

As to follow the french regulation, the title of the message or

the file name are transmitted in readable ascii, not compressed.

The offset is also transmitted in ascii and specifies the offset

at which the data should be inserted in the file (in case of a

fragmented file). In the version 5.12, this parameter is not

utilized and is always equal to zero.

A data block contains from one to 256 bytes. It begins by two

bytes which specify the format.

Data block format :

<STX> 1 byte = 02 hex

Number of data 1 byte = 00 to ff hex.

if length is 256 bytes, the value is 00.

Data bytes 1 to 256 bytes

The last data block is followed by the end of file specifier and

the checksum.

End of file specifier format :

<EOT> 1 byte = 04 hex

Checksum 1 byte = 00 to ff hex

The checksum is equal to the sum of all the data bytes of the

transmitted file, modulo 256 (8 bits) and then two's complemented.

The checking of the checksum is very simple :

The sum of the datas from the file and the checksum received

modulo 256 shall be equal to zero.

In case of a checksum error, the message or the file is not taken

to account and the system issues a disconnect request after having

sent the comment :

\*\*\* Erreur checksum

Ascii values of the characters (1 byte) used in the protocole :

<NUL> = 00 hex

<SOH> = 01 hex

<STX> = 02 hex

<EOT> = 04 hex

Most of ideas for this binary transmission are issued from YAPP

protocole. Thanks to WA7MBL.

Extension to the protocole. Compressed forward FBB Version 1.

-------------------------------------------------------------

The protocole utilized for the transfer of ascii files compressed

is an extension to the existing protocole. The compressed forward

is validated by the presence of the letters B1 in the SID

[FBB-5.15-B1FHLM$]. The transfer of compressed files may only take

place under FBB protocole. The presence of the letter B in the SID

without the F letter will remain without effect.

The differences as regard to the binary protocol version are:

- A variable number of fields in the submit line, but at least 7

(as in previous version).

- A new set of answers :

+ or Y : Yes, message accepted

- or N : No, message already received

= or L : Later, already receiving this message

H : Message is accepted but will be held

R : Message is rejected

E : There is an error in the line

!Offset: Yes, message accepted from (Offset).

Most of these answer do not need explanation or were already used

in previous version. + and Y, - and N, = and L are equivalent but

are still used for compatibility.

!Offset asks the remote BBS to start transfer from Offset.

For instance, YL!3350RH (or +L!3350RH) means that :

- 1st message is accepted

- 2nd message is delayed

- 3rd message will be sent from offset 3350 (in compressed file)

- 4th message is refused

- 5th message is accepted but will be held

The submission of an ascii message will be in the form :

FA P FC1CDC F6ABJ F6AXV 24754\_F6FBB 345

The submission of a binary file will be in the form :

FB P FC1CDC F6ABJ F6AXV 24754\_F6FBB 345

The transfered data are of a specific format. The transfer will be

done in binary mode. This last one is derived of the YAPP protocol

which is very reliable. All transfer is made of a header, a block

of data, an end of message and a checksum. Each transfer is

equivalent to the transfer of one message of the standard protocol

and shall not be followed by a control Z, the end of file

specifier is defined in another way. Unlike YAPP transfers, there

is no acknowledgement during the transmission of messages, the

protocole is then more simple and efficient.

Format of header for an ascii compressed message (submission FA) :

<SOH> 1 byte = 01 hex

Length of the header 1 byte = Length from the title,

including the two <NUL> characters.

Title of the message 1 to 80 bytes

<NUL> 1 byte = 00 hex

Offset 1 to 6 bytes

<NUL> 1 byte = 00 hex

data blocs ...

Format of header for a binary compressed file (submission FB) :

<SOH> 1 byte = 01 hex

Length of the header 1 byte = Length from the filename,

including the two <NUL> characters.

Name of the file 1 to 80 bytes

<NUL> 1 byte = 00 hex

Offset 1 to 6 bytes

<NUL> 1 byte = 00 hex

data blocs ...

As to follow the french regulation, the title of the message or

the file name are transmitted in readable ascii, not compressed.

The offset is also transmitted in ascii and specifies the offset

in the file from which the data will be sent.

A data block contains from one to 256 bytes. It begins by two

bytes which specify the format.

Data block format :

<STX> 1 byte = 02 hex

Number of data 1 byte = 00 to ff hex.

if length is 256 bytes, the value is 00.

Data bytes 1 to 256 bytes

The first block data first contains the CRC16 of the full binary

file, then the size of the full uncompressed file, and then the

binary from offset 0 or specified offset if !Offset was asked.

The last data block is followed by the end of file specifier and

the checksum of the data sent.

End of file specifier format :

<EOT> 1 byte = 04 hex

Checksum 1 byte = 00 to ff hex

The checksum is equal to the sum of all the data bytes of the

transmitted file, modulo 256 (8 bits) and then two's complemented.

The checking of the checksum is very simple :

The sum of the datas from the file and the checksum received

modulo 256 shall be equal to zero.

In case of a checksum error, the message or the file is not taken

to account and the system issues a disconnect request after having

sent the comment :

\*\*\* Erreur checksum

The CRC16 is computed for the full binary file including the length

of the uncompressed file (4 bytes in top of file).

In case of resume, it will be the only mean to be sure that the part

of the already received file matches with the new one.

The LZHUF\_1 program used with option "e1" generates a binary compressed

file with the following format :

CRC16 : 2bytes

Length: 4 bytes

Datas : rest of the file

In case of forwarding with a BBS using version 0, only the part from

offset 2 will be sent

In case of forwarding with a BBS using version 1, the 6 top bytes will

be always sent, then seek to Offset+6, then send data.

Ascii values of the characters (1 byte) used in the protocole :

<NUL> = 00 hex

<SOH> = 01 hex

<STX> = 02 hex

<EOT> = 04 hex

Comments will be welcome.

F6FBB @ F6FBB.FMLR.FRA.EU