Date: 1 Nov 2022
Authors: Chris Thompson, G0KLA / AC2CZ
Version: 0.1

PACSAT BBS Operations

1 Introduction

This document is an initial discussion of the basic operations of a new PACSAT Bulletin Board System (BBS). This can be used as the basis for a detailed design and to help choose the hardware needed for things like memory and compute.

This document is organized into the following sections:

- **Purpose of the BBS** – what is the primary purpose of a PACSAT BBS in an era with high speed internet access available to almost every potential ground station?
- **Principles** – a discussion of the guiding principles that we can use to design the required BBS Operations.
- **BBS Operations** – the key operations that the BBS must support and a summary of their functionality.

References are given at the end. In general the technical details of the PACSAT protocols are not discussed. Those should be laid out clearly in a more detailed design.

For the purpose of this document we assume the following definitions:
Ground Station – or station. The end user who wishes to access or send new data to the BBS.
Control Station. A station capable of sending administrator commands to the BBS.
PACSAT File – A file or message stored in the BBS. Either uploaded from the ground or created on the spacecraft.
PACSAT File Header – A set of meta data associated with a PACSAT File. See [1].
Directory – The collection of files or messages that are stored in the BBS.
BBS – The Bulletin Board System or program running on the satellite that allows the storage of PACSAT Files in a Directory and supports the query, retrieval, broadcast and management of those files.

2 Purpose

Originally the PACSATs were setup to disseminate information, especially packet messages, or “email’, to the Amateur Radio community. Several parts of the PACSAT Protocol deal with mail forwarding and automated gateways. This is now obsolete. The ease of moving information across the internet, and even the availability of high speed internet via satellite (SpaceX Starlink and others), means that other purposes should drive our design. These are likely education, research and fun, with fun being the characteristic that will sustain use by many stations.

You may want to add emergency response, but that is difficult given the small number of passes and the technical requirements of a ground station. But it should be discussed.

2.1 Education

Initially it is a technical challenge and very rewarding to receive the signals from the PACSAT. It is equally rewarding to upload a file or receive a personal message. A lot can be learned to create a reliable automated ground station. This can be a great project for Amateurs and educators everywhere.

Once those technical challenges are overcome and the ground station automatically handles each pass, it is easy to loose interest and move on. So there needs to be more.

2.2 Research

We hope that Amateur Radio will soon extend to missions around the moon and beyond. Those spacecraft will need delay tolerant protocols and would be enriched by the ability to upload and download stored information. We can test the technologies needed in Low Earth Orbit (LEO) and have them ready when the opportunity exists to support deep space missions.

The PACSAT protocols are antiquated, were designed to run through a narrow RF channel and have no forward error correction. We can experiment with updated schemes to achieve a better result with more modern approaches, while remaining as compatible as possible with the original schemes.

2.3 Fun

If you are familiar with computer games, then the most realistic or the most technically competent games are not the best. The primary measure of success is how fun they are. We should support as many features as possible that make use of the spacecraft as fun as possible. This should be above and beyond the fun of overcoming the technical challenges to setup a ground station. Here are some ideas, though there are surely many more:

- We could automatically tally DXCC (or a suitable equivalent) based on the messages that stations exchange with different DXCC counters. The “scores” could be available in a file and could be periodically broadcast to all stations. This would encourage the two way exchange of messages with as many countries as possible.
- Stations should be encouraged to collect telemetry. The “leaderboard” for contributors could be available as a file and periodically broadcast.

- Messages should be threaded into discussions using a new PACSAT File Header tag. The ground station could then group these together similar to modern email or discussion forums.
- It should be possible to like or up-vote messages and the ground station should support sorting to show those messages first. The current tally would be stored in a new PACSAT File Header tag. Sending the up-votes would require a new Broadcast Request message type or could be exchanged when a ground station logs in to perform an upload.
- Control stations should be able to “pin” messages so they appear at the top of the ground stations list. This could be supported by a new PACSAT File Header tag that can only be set by the server.
- Ground stations could be issued badges or honors in a similar way to modern discussion forums.
- We expect modern files to embed formatting and images. This can all be supported with the existing protocols if the ground station supports HTML or other markup. The images don’t even need to be on the spacecraft, though they could be.

2.4 Emergency Response

When FalconSat-3 was handed over to the Amateur Radio community Bob Bruninga, WB4APR (SK), suggested that all of the received bytes be sent to a central place so that a web interface could be provided to the BBS. Then emergency response messages could be exchanged with less need for a sophisticated ground station. At least at one end. We could consider this, or other similar use cases, but emergency response requires a ground station that is available and usable in an emergency response situation. It would somehow need to outperform the traditional nets on 40m or 20m.

We should also keep in mind that mirroring the contents of the BBS Directory on the web might take some (or all) of the “fun” out of building your own copy of the BBS Directory at your ground station. While we did collect all of the data centrally for FalconSat-3, we only displayed the telemetry centrally and never displayed the consolidated messages and Directory. But this can be discussed further.

3 Principles

There are many principles which drive the design of the BBS. A few important ones are summarized below and put in the context of the modern era (historically these were discussed in reference [2]):

3.1 Broadcast protocol vs connected mode

The BBS stores and provide files or messages. AX.25 Packet Radio[5] is very inefficient at transmitting files to many users. A traditional BBS typically uses connected mode which guarantees that a station downloads a complete file. If 100 stations need to access the *AMSAT ANS Bulletin* then it must be downloaded 100 times. A satellite is unlike a terrestrial BBS because it covers a vast area. It can therefore broadcast a file to many users at the same time. We can leverage this to make downloads much more efficient, but we need to cope with missing sections of a file due to poor reception.

3.2 Non-interactive operations during a pass

Unlike a traditional BBS, stations do not login and interact with online commands throughout a pass. Each pass of the satellite is only 10 minutes at most. The ground station user will therefore make choices in advance of a pass, marking the files they want to download or setting up rules to automatically download new files that are of interest.

This means that the ground station must hold a copy of the information that is stored on the spacecraft, or at least a copy of the Directory. We can't login and search for things. Instead we broadcast a complete copy of the Directory to stations on the ground with the ability to close holes in their listings. Then a user can query their local copy of the Directory and select which PACSAT Files to download.

3.3 Receive only stations

Many stations will only be able to receive data, perhaps using just a simple Software Defined Radio (SDR) and will not be able to transmit. The broadcast protocols can allow a rich experience for these stations assuming transmissions are planned to support them. For example, recent directory entries could be automatically broadcast over populated areas to give receive only stations details of what is on the spacecraft. Important or interesting PACSAT Files can be broadcast on a schedule defined by a BBS Control Station. This could have the popularity of Slow Scan TV from the ISS if the content is chosen carefully and is not available elsewhere.

4 BBS Operations

The BBS consists of the following components which are discussed below:

- PACSAT File Headers
- Directory
- Files
- PACSAT Broadcasts
- File Uploads
- Command and Control

4.1 PACSAT File Headers

Each message or file stored in the BBS has a PACSAT File Header [1]. This Header stores meta data needed to manage the file on the spacecraft and to help users decide what is stored in the file. When the user receives the Directory then only the PACSAT File Headers are transmitted.

The PACSAT File Header is pre-pended to the start of the file when it is uploaded or broadcast.

The header is split into three sections:

- Mandatory Fields – these fields are always present and are in a fixed order
- Extended Header – these fields are present on all “messages”. If a file is a message then all extended fields must be present and are in a fixed order.
- Optional Header – these are only included if needed. Any or all of the fields can be included and can be in any order. We can add new fields for our own purposes.

4.1.1 Important fields

The most important fields for the operation of the BBS are in the Mandatory header. One other field is discussed in reference [1] as being in the Extended Header, but is mandatory - the uploadTime field. This was made mandatory in an update to the PACSAT Protocols, see reference [8].

The following fields are the most important:

FILE_ID – Every PACSAT File is identified by a 32 bit integer. The numbers are not reused once files are purged, so in theory this needs to last the lifetime of the spacecraft. This is the

number that stations transmit when they request the broadcast of a specific file. It is also the number attached to a fragment of broadcast data so the ground station knows where to store it. It is the number that the spacecraft issues to a ground station that wants to upload a new file.

FILE_TYPE – Every PACSAT File has a file type represented by an 8 bit number and defined in reference [3]. These are primarily used by the Ground Station which can perform post processing of the file or allow a helper program to open or display it.

BODY_OFFSET – This defines how long the PACSAT File Header is and where the actual file data starts.

BODY_CHECKSUM – This is a simple checksum over the PACSAT File Data. It is primarily to detect uncorrected Single Event Upsets (SEU) in the PACSAT memory, but it also detects errors in your program when you have bugs. Which is quite useful.

HEADER_CHECKSUM – This is similar to the Body Checksum but for the Header.

UPLOAD_TIME – It is not sufficient to use FILE_ID to confirm if the full directory has been received on the ground. Some FILE_IDs may be discarded because uploads failed, or some files may have been purged. To confirm that all PACSAT File Headers have been received on the ground we use their upload times. This is discussed in the Directory section below and in reference [8].

4.1.2 Other fields

Almost all other fields, while they may seem important, are stored by the BBS and simply passed back down to the ground. Some, like CREATE_TIME, could also be set by onboard system created files but are just for information.

4.1.3 Filenames (or not)

It is worth mentioning that the FILE_NAME and FILE_EXT fields are an 8.3 DOS like format that was used to store the file name in the some of the PACSATs. In reality the file can be stored in the BBS using the FILE_ID as the filename, or as a handle on a memory pointer. We do not need a separate name. So we can ignore it in most cases. On FalconSat-3 housekeeping processes, such as Whole Orbit Data (WOD) used these fields to store the filename, but it carried no further information.

The actual name of a user message file, once the PACSAT File HEADER is removed, could be interpreted as the FILE_NAME.FILE_EXT, but there is also a USER_FILE_NAME field which is in the Optional Header. This is used by the ground station software and can be ignored by the BBS.

In detailed design we should standardize how these fields are used.

4.1.4 Optimization

There are a lot of redundant fields in the PACSAT File Header. We may want to consider a streamlined header that could save considerable bandwidth. There are 5 dates for example. All 4 bytes long. Only upload_time is important. There are 4 “callsigns” or to/from addresses covering at least 40 bytes and up to a maximum of 86 bytes (all in addition to the two 7 byte callsigns in the AX.25 header).

The design of the PACSAT File Header was to support email traffic management and to allow a file to be recreated on the ground with its original name and key attributes. It is unlikely either are still a requirement. If we remove these fields and someone still needs to send a file with its file attributes then they can ZIP it into an archive and send the archive as a binary file. Once unzipped the file is recreated as before. That could remove a lot of overhead from most file transmissions.

The protocol includes version numbering in every packet. The version number is 2 bits long and is currently version 0b00. We could perhaps create version 0b10 of the PACSAT File Header with an updated layout.

4.2 Directory

4.2.1 Directory Broadcasts

Users are not interactively connected to the BBS listing files or searching for content. Instead they will store a copy of the Directory at their ground station. In the BBS the Directory consists of PACSAT Files, each with a PACSAT File Header. The newest Directory entries can be automatically broadcast to all users or users can request entries that they are missing.

A Directory Broadcast is sent in an AX.25 Unnumbered Information (UI) Frame with Protocol Identifier (PID) of 0xBD (See reference [5] for details of UI frames). A Directory Broadcast contains a short header indicating the FILE_ID this refers to, an OFFSET, two dates (discussed below) and the bytes of the PACSAT File Header. The OFFSET (and one bit in the flag byte) allows a long PACSAT File Header to be sent across two or more packets. This is also why you need the FILE_ID at the start of each Directory Broadcast Packet. You could be receiving the second half of a Directory Header without its FILE_ID field.

4.2.2 t_old and t_new

The Directory Broadcast header contains two dates called t_old and t_new which allow the Ground Station to determine if it has a complete directory listing. The two dates guarantee:

There are no files other than this one where t_old <= UPLOAD_TIME <= t_new

The BBS does not need to store these old and new times, it can calculate them when each Directory Broadcast is transmitted. The Directory needs to be sorted by UPLOAD_TIME and each file needs a unique UPLOAD_TIME. When a Directory Broadcast is sent:

t_old is set to 1 second after the upload time of the previous file

t_new is set to 1 second before the time of the next file

This is discussed in reference [8].

4.2.3 Directory Requests

A ground station can use the t_old and t_new dates to calculate a Directory Broadcast Request for any “holes” that it has in its Directory. These are identified as gaps in the date ranges. A complete directory will have no gaps between the t_old and t_new dates of the Directory Headers.

If there are gaps then the ground station transmits an AX.25 UI Frame with a PID of 0xBD and a short header indicating that this is a directory fill request. The header is followed by a list of date pairs indicating the holes in its directory.

Upon receiving the Directory Broadcast Request the BBS will transmit all of the Directory Headers that have an UPLOAD_TIME between the date pairs.

Once the directory broadcast is complete, if the ground station still has holes in the directory, then it can transmit a new Directory Broadcast Request for the remaining holes.

4.3 Files

4.3.1 File Broadcasts

Files are identified by their FILE_ID. Interesting files can be broadcast automatically or files can be requested by a ground station.

A File Broadcast is an AX.25 Unnumbered Information (UI) Frame with PID of 0xBB. It contains a short header indicating the FILE_ID this refers to, the FILE_TYPE, an OFFSET and a set of bytes from the file. The OFFSET indicates where these bytes should be inserted into the file and is the exact number that is used in a file seek() operation before they are written to storage.

The ground station saves all received file broadcasts to disk, even broadcasts meant for other people. If files are partially received then the ground station operator can later decide if they want to receive the rest of the file, which can be requested with File Requests.

4.3.2 File Requests

A ground station can send a File Broadcast Request to download new files or for any “holes” that it has in a file. A hole in a file is represented by a byte offset and length, indicating the bytes that are missing. The BBS does not need to know how to calculate this and it is easy to process. For those interested, the ground station uses the algorithm for IP Packet re-assembly (RFC 815) to calculate the holes. See reference [7].

The ground station transmits an AX.25 UI Frame with a PID of 0xBB and a short header indicating it is a File Broadcast Request. It contains the FILE_ID and either a flag indicating the the whole file should be transmitted or a flag indicating that that a list of holes should be transmitted. This is then followed by the holes list.

The BBS works through the File Broadcast Request transmitting the file or the requested parts of the file in chunks. Once the operation is complete, if the ground station still has holes in the file, then it can transmit a new File Broadcast Request with the remaining holes.

4.4 PACSAT Broadcast

4.4.1 PB Status

Ground stations monitor the down-link frequency of the spacecraft until they can hear its signal. Specifically they look for the status of the PACSAT Broadcast or PB before they transmit a Broadcast Request. The BBS must transmit this frequently. Confusingly it contains information encoded in the “destination” callsigns name, as well as a list of the callsigns that are currently “on the PB”. e.g.:

PB AC2CZ G0KLA/D	*- sent to the callsign PBLIST*
PB Empty.	*- sent to the callsign PBLIST*
PB Closed.	*- sent to the callsign PBSHUT*

The first means that AC2CZ and G0KLA are on the PB and G0KLA is receiving a directory broadcast. The second indicates that no one is on the PB and the last that the PB is shut and not available. If the PB is full and no more stations can be added then the status is sent to the callsign PBFULL.

Each time the BBS receives a Broadcast request it is added to the PB. The PB acknowledges the station by sending a UI packet with PID 0xF0. This tells the ground station it can stop sending its request. The ground station will also stop if it sees that it is now on the PB or is receiving the requested content. The response when the station is added to the PB is, e.g.:

OK AC2CZ

The BBS will reject Broadcast requests from stations that are already on the PB or who request invalid files. It will send a message like these in a UI frame:

NO -1 AC2CZ

NO -2 AC2CZ

Where -1 indicates a “temporary error” and the station should try later, -2 indicates a file is not available. The full list of errors is not well defined and should be part of our design.

Interestingly the PACSAT Protocols say that these “OK” and “NO” messages are for manual ground stations and should not be used by automated ground stations. But it is not clear how an automated station would otherwise easily obtain this information.

4.4.2 PB Operation

The BBS then services the stations on the PB round robin, sending the next packet for each one in turn. The packets don’t go to the stations directly. They are sent to destination QST-1 as AX.25 UI frames with a PID of 0xBB for Files and 0xBD for Directory Headers. All stations can receive them. So in the first PB example above G0KLA is also receiving the file that AC2CZ requested and AC2CZ is receiving the Directory Headers. As is any other station listening in. Any duplicate data is ignored on the ground.

If a station is on the PB for a time exceeding the timeout (typically 10 mins to match a pass length) then it is removed. If the request for a station is complete then it is also removed.

There is also a BBS mechanism to cancel a file request but this is not implemented in PacsatGround. There is currently no provision to cancel a directory request.

4.5 File Uploads

If would be very inefficient to upload files using AX.25 UI frames. While they have an advantage when broadcasting to many stations, they offer no advantage when only one station (the BBS) is receiving the file. AX.25 connected mode already contains the logic to ensure that all data is successfully sent and acknowledged. File Uploads therefore follow the File Transfer Level 0 (FTL0) Protocol, which sits on top of AX.25 Connected mode. See reference [5] for full details of AX.25 connected mode. See reference [4] for a full discussion of FTL0, but note that the original design for FTL0 envisaged Uploads, Downloads and queries of the BBS. Little of that functionality was actually used. It is restricted to the upload of files.

4.5.1 Uplink Status

Similarly to the PB, which controls and publishes which stations are receiving broadcast requests, FTL0 periodically publishes the availability of the uplink with something like the following:

Open ABCD:

Open ABC G0KLA D:

The first message means that the uplink is available and there are no stations connected to any of the four receivers. The second means that G0KLA is connected to the receiver on frequency C. More than one station can connect on each frequency, but clearly you have the best chance of connecting on a clear frequency. The ground station can choose to interpret and display these messages as it sees fit. If it is in control of the uplink frequency then it can try to pick a clear channel. Note however that this gives no indication how busy each receiver is handling broadcast requests.

These messages were not well standardized on past PACSATs and are confusing. We could discuss the best way to send this information.

4.5.2 Uplink Process

The ground station connects to the BBS to initiate an uplink by sending a normal AX.25 SAMB frame to the BBS callsign. This is a separate callsign to the one used for Broadcast Requests. The BBS negotiates the connection in the normal way and then the two stations exchange connected mode I-frames containing data, with a normal AX.25 Disconnect at the end. See reference [5] for a complete discussion of AX.25.

The data stream is considered continuous and not bounded by the start and end of the I-frames. Instead we process FTL0 frames which consist of a header, a defined type, a length and a chunk of data specified by that length. The data may run across I-frames.

The FTL0 conversation proceeds as follows to upload a file:

1. LOGIN_RESP: Once connected the server sends an FTL0 LOGIN_RESP packet with 5 information bytes giving the login time and an information byte.
2. UPLOAD_CMD: The ground station then sends the FTL0 UPLOAD_CMD packet with a FILE_ID and a length. If this is a new file then the FILE_ID is zero.
3. UL_GO_RESP: If the server has room for the file it sends an FTL0 UL_GO_RESP packet. If this is a new file it allocates a new FILE_ID. If it was an existing file, which the ground station has partially uploaded, then it includes the byte offset of the next byte it expects to receive.

4. UL_ERROR_RESP: If the server does not have room for the file or if there is any other error (perhaps the partially uploaded file has been purged or the number was not valid) then it sends an FTL0 UL_ERROR_RESP packet with an appropriate error message.
5. DATA: The ground station then sends a series of FTL0 DATA packets containing the bytes of the file, beginning at the offset number supplied by the server. There is no need for the server to “reply” to each packet with another FTL0 packet because this is taken care of by the AX.25 connected mode. Each I-frame receives an RR Frame in response confirming or asking for a resend, until each packet is received or we time out.
6. DATA_END: After transmitting the last DATA packet the ground station sends an FTL0 DATA_END packet.
7. UL_ACK_RESP: The server checks the integrity of the uploaded file including recalculating the check sums in the PACSAT Header. If it passes then a UL_ACK_RESP FTL0 packet is sent.
8. UL_NAK_RESP: The server can terminate the upload at any time by sending an FTL0 UL_NAK_RESP frame. This is also sent after the DATA_END packet if something went wrong. It includes a suitable error message.
9. Time out: If the link fails during the upload then the server retains the offset of the last byte received and disconnects the ground station.
10. The client can choose to upload another file while it is connected or it can then close the AX.25 connected mode link in the normal way.

This upload process is explained in detail in Reference [4] with a state machine design in Appendix B. Ignore all parts of this document except the upload of files.

4.6 Command and Control

Previous PACSATs allow control stations to set many BBS parameters from the ground. Some examples include:

- How many stations are allowed on the PB?
- What telemetry is collected and stored in files for downloaded?
- How long are files retained?

- What files are automatically broadcast?
- How often is the Directory automatically broadcast?

PACSATs allowed the re-upload and re-initialization of the BBS software. This allowed changes and updates to be made after launch. Clearly this mechanism would need to be robust.

4.7 Summary of UI frame types

Frames sent by the server from the broadcast callsign:

Type	AX.25 Frame	AX.25 PID	To Callsign
PB Status	UI	F0	PBLIST
PB is Full	UI	F0	PBFULL or PBCTRL
PB is Closed	UI	F0	PBSHUT
Directory Broadcast	UI	BD	QST-1
File Broadcast	UI	BB	QST-1
Broadcast OK / NO	UI	BB	Ground Station callsign
Various with intent indicated by destination callsign	UI	F0	TLMI-1 TLMI-2 LSTAT TIME-1

5 References

1. PACSAT File Header Definition, Jeff Ward G0/K8KA, Harold E. Price NK6K, https://www.g0kla.com/pacsat/fhead.txt
2. PACSAT Protocol Suite – An Overview, Jeff Ward G0/K8KA, Harold E. Price NK6K , https://www.g0kla.com/pacsat/intro.txt
3. Historical PACSAT File Types, https://www.g0kla.com/pacsat/ftypes.txt
4. PACSAT Protocol: File Transfer Level 0, Jeff Ward G0/K8KA, Harold E. Price NK6K, https://www.g0kla.com/pacsat/ftl0.txt and its update: https://www.g0kla.com/pacsat/ftl0u1.txt
5. AX.25 Link Access Protocol for Amateur Packet Radio, Version 2.2, July 1998, William A. Beech, NJ7P, Douglas E. Nielsen, N7LEM, Jack Taylor, N7OO, https://www.tapr.org/pdf/AX25.2.2.pdf

6. FX.25, https://en.wikipedia.org/wiki/FX.25_Forward_Error_Correction
7. IP DATAGRAM REASSEMBLY ALGORITHMS RFC 815, David D. Clark, 1982, herehttps://tools.ietf.org/html/rfc815
8. PACSAT Broadcast Protocol Update 2, eff Ward G0/K8KA, Harold E. Price NK6K, https://www.g0kla.com/pacsat/bdcastu2.txt