Communications Infrastructure for the MOST Microsatellite Project (excerpt)

Laura Halliday

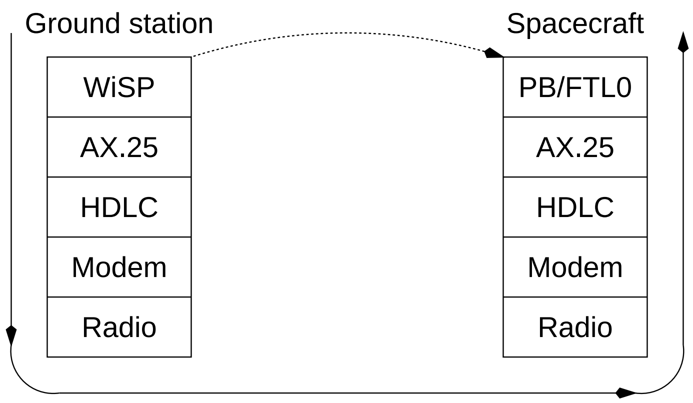


Figure 2.1: The layers of a satellite communication system

2.2 AX.25

Amateur digital communications, including amateur satellite communications, use the AX.25 protocol. This is a derivative of the standard X.25 protocol, with enhancements and adaptations for amateur radio use. As documented in its reference manual (1), AX.25 defines a vast protocol with sophisticated flow control and elaborate facilities for creating and managing connections. Amateur satellite communications use very little of this: with one exception they use one type of AX.25 packet, the UI (**U**nnumbered **I**nformation) packet, with all connection management and flow control handled by higher-level protocols. This results in exceptionally simple implementations: as part of this research the necessary subset of AX.25 was implemented in a few hours in Microsoft Visual Basic.

The exception to the use of UI frames is the FTL0 protocol for uploading files, which uses the connected-mode facilities of AX.25. FTL0 is discussed in Section 3.4.

AX.25 transmissions are organized in *frames*, where each frame is an individual datagram. The layout of an AX.25 information frame (there are other kinds, but they are not used in PACSAT communications) is shown in Figure 2.2.

Address	Control	Protocol	Information	FCS

Figure 2.2: AX.25 information frame layout

The encoding of each field will be discussed in turn.

Table 2.1: Example of AX.25 address encoding

Character	**Hex value**	**Shifted hex value**
V	`0x56`	`0xac`
A	`0x41`	`0x82`
3	`0x33`	`0x66`
S	`0x53`	`0xa6`
F	`0x46`	`0x8c`
L	`0x4c`	`0x98`
0	`0x00`	`0x60`
U	`0x55`	`0xaa`
O	`0x41`	`0x82`
S	`0x53`	`0xa6`
A	`0x41`	`0x82`
T	`0x54`	`0xa8`
5	`0x35`	`0x6a`
11	`0x0b`	`0x6c`

The *Address* field specifies the origin and destination of the message. Addresses are encoded in ASCII, shifted one bit over to allow the least-significant bit to provide control information—it is set to one on the last byte of the last callsign in the field.

An example of address encoding for a packet sent from `UOSAT5-11` to `VA3SFL-0` is presented in Table 2.1. The encoding of the SSID (**S**econdary **S**tation **ID**entifier) is illustrated in Table 2.2. The address extension bit is adapted from HDLC (Section 2.3): it is `0` if there are more addresses in the frame, and `1` if the address is the last in the frame.

The AX.25 SSID performs the same function as ports under TCP, permitting a single station to have multiple presences on the network, possibly with different functions.

The *Control* field, one byte, is always `0x03` in the UI frames that constitute the vast majority of satellite traffic.

The *Protocol* field, more correctly, the Protocol ID field, is one byte in length and specifies if a higher-level protocol is in use. In PACSAT communications the value is almost invariably `0xf0`, meaning that no higher-level protocol is in use.

Table 2.2: AX.25 SSID encoding

Bit number	**Item**
7	Command, always `0`
6–5	Reserved, always `11`
4–1	Numeric SSID
0	HDLC address extension bit

The *Information* field contains whatever data the application wishes to send.

The FCS (**F**rame **C**heck **S**equence), also called the CRC (**C**yclic **R**edundancy **C**heck), provides a 16 bit checksum to maintain data integrity. The checksum calculations are based on the CCITT standard polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$. The fading nature of the channel between the spacecraft and the ground station produces errors in bursts, and the performance of this and similar codes has been analyzed extensively (2) in such an environment. The code used by AX.25 can detect all error bursts of length 16 or less, 99.997% of all error bursts of length 17, and 99.9985% of all error bursts of length greater than 17.

2.3 HDLC

Like AX.25, HDLC (**H**igh-**L**evel **D**atalink **C**ontrol) defines an elaborate protocol. For satellite AX.25 purposes HDLC merely defines a *line code* (3). A line code alters the data stream in order to better adapt it to the transmission medium. In the present case HDLC provides three functions:

- NRZI differential encoding
- Bit stuffing
- Frame delimiting

HDLC is a stream-oriented protocol, and there is no notion of grouping individual bits to form bytes. In the discussion of HDLC the term “octet” will be used only as a grouping of 8 bits, with no implication that the grouping is significant to a higher-level protocol.

2.3.1 NRZI differential encoding

NRZI (**N**on-**R**eturn to **Z**ero **I**nverted) encoding provides differential encoding of the bit stream that is to be transmitted. Differential encoding encodes data in the *difference* between successive bits, rather than in the bits themselves. This is a convenience in some of the modulation formats used in satellite operations, like MSK (Section 2.5.4), but is mandatory in other formats like BPSK (Section 2.5.3).

In NRZI encoding a zero bit is transmitted by a change in the output, while a one bit is transmitted by no change in the output. This is illustrated in Figure 2.3. Note that the absolute value of the output is irrelevant: it is only *changes* that convey information. The two possible output waveforms are thus equivalent.

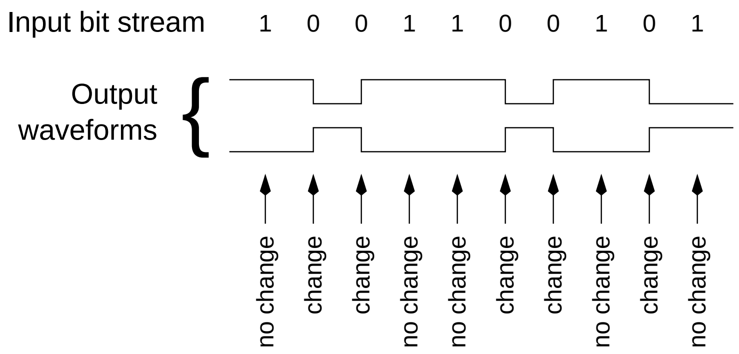


Figure 2.3: NRZI encoding

2.3.2 Bit stuffing

A weakness of NRZI over radio channels is that if there are too many one bits in a row, a DC component will be necessary for transmission—in systems whose bandwidth does not (and often cannot) extend to DC. Additionally, recovering the timing of individual bits—easy for zero bits under NRZI—can become unreliable when an extended series of one bits is transmitted. The solution adopted by HDLC and other stream-oriented data transmission schemes like USB (4) is *bit stuffing*: if too many bits of one kind appear in a row, insert a bit of the other kind on transmission, and remove it on reception. This is illustrated in Figure 2.4.

Under HDLC only five consecutive one bits may occur in outgoing data. Should a sixth one bit occur, a zero is inserted on transmission and removed on reception.

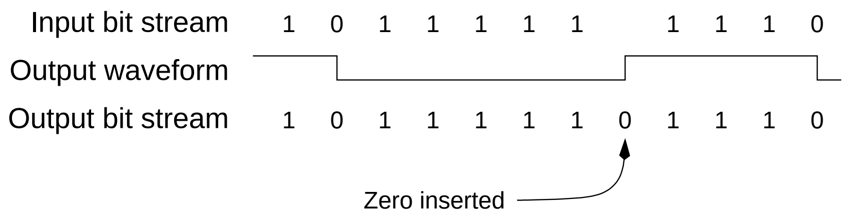


Figure 2.4: HDLC bit stuffing

2.3.3 Frame delimiting

HDLC indicates the beginning and end of a frame with a bit pattern that is not permitted in user data: the octet `0x7e`, with the otherwise-forbidden bit pattern `01111110`, which is not subject to bit-stuffing. By convention an idle HDLC channel sends consecutive frame characters, as illustrated in Figure 2.5.

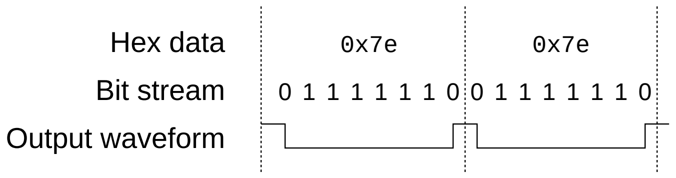


Figure 2.5: Bit patterns and waveforms on an idle HDLC channel

2.3.4 Polynomial scrambling

While not part of HDLC—it is, typically, a modem function—the application of polynomial scrambling to transmitted data is closely related to the functions provided by HDLC.

The process of scrambling the data enhances its transmission in several ways:

- An increased density of transitions further eases timing recovery.
- An increased density of transitions further reduces the low frequency bandwidth requirements of the system.
- The pseudo-random nature of the scrambled data renders the transmitted spectrum noise-like, with no spectral lines that could interfere with other services in shared spectrum allocations.

The standard 9600 baud analogue MSK modem was designed by Miller in the 1980s (5) and uses polynomial scrambling. Miller's design, often referred to by his amateur radio callsign G3RUH, uses the scrambling polynomial $1 + x^{12} + x^{17}$, where each output bit is computed from the current bit and the bits transmitted 12 and 17 bits ago, as illustrated in Figures 2.6 and 2.7.

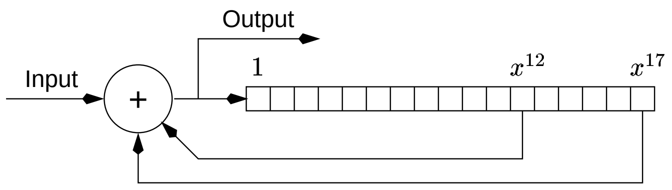


Figure 2.6: Polynomial scrambler in the G3RUH modem

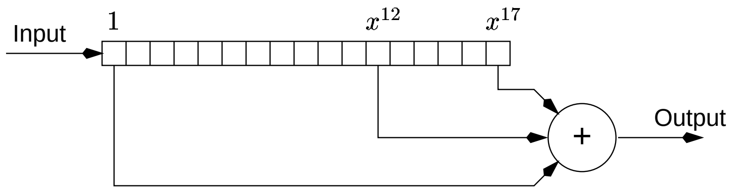


Figure 2.7: Polynomial descrambler in the G3RUH modem

The original implementation was built with discrete logic shift registers and exclusive-OR gates. Modern implementations use programmable logic (6), or perform these operations in software (7).

The scrambler and descrambler are closely related: the scrambler divides the bit sequence by the polynomial, while the descrambler multiplies by the same polynomial. In theory either circuit can serve as the scrambler with the other as the descrambler, but in practice the feedback division circuit is only used in the scrambler, because if it was used in the descrambler an erroneous received bit could circulate indefinitely. The circuit in Figure 2.7 will thus only output three corrupt bits on reception of a single erronoeus bit. This is acceptable in the AX.25 environment where packets are, in any case, rejected for a single detected bit error.

2.4 KISS

KISS (**K**eep **I**t **S**imple, **S**tupid) is a means of encapsulating AX.25 datagrams for transmission over serial lines. It was devised as an interim means for implementing new protocols in amateur digital radio systems (8).

Table 2.3: KISS command functions

Hex value	**Function**
`0x00`	**Data frame**
`0x01`	**Transmitter keying delay**
`0x02`	Persistence
`0x03`	Slot interval
`0x05`	**Full duplex**
`0xff`	**Exit KISS mode**

When radio amateurs first started experimenting with packet radio the available computers had limited capability and were not able to implement the AX.25 protocols in their own software. The solution was a separate, dedicated external device called the TNC (**T**erminal **N**ode **C**ontroller) that included protocol and modem functionality.

This approach worked, but it was inflexible: since the protocol was fixed in the TNC firmware, it was impossible to experiment with different protocols. The solution was to bypass the protocol functions of the TNC and only use the modem. The protocol devised to communicate between more-powerful host computers and the TNC, with much of its functionality now disabled, was KISS. Many modern implementations carry this progression further and perform *all* their processing in software, requiring little more than a sufficiently powerful host computer with a sound card.

KISS was modelled on the TCP/IP standard SLIP (9), with enhancements for amateur radio use.

A frame transmitted by KISS is a complete AX.25 frame (Figure 2.8) without the checksum or HDLC encoding. On transmission the TNC will compute the CRC and perform HDLC encoding. On reception it is the task of the TNC to remove the HDLC encoding and validate the checksum before making the frame available to the host.

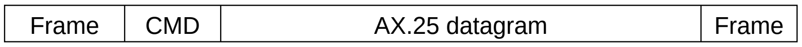


Figure 2.8: The layout of a KISS frame

The KISS command function byte is provided so that the host computer may give instructions to the TNC. The standards on this are loose, but the minimal requirements in the standard are listed in Table 2.3. Only the **bold face** commands are relevant to satellite operations; the others configure the system for terrestrial operation.

Table 2.4: KISS character usage

Function	**Hex value**	**Mnemonic**
Frame end	`0xc0`	`FEND`
Frame escape	`0xdb`	`FESC`
Transposed frame end	`0xdc`	`TFEND`
Transposed frame escape	`0xdd`	`TFESC`

KISS reserves several characters for its own purposes (Table 2.4). Key is the frame end character `FEND`, which delimits frames. Should the bit pattern corresponding to `FEND` occur in data it is replaced by the combination of the frame escape character `FESC`, followed by the transposed frame end character `TFEND`. Similarly, should `FESC` occur in the data, it is replaced by the combination of `FESC` followed by the transposed frame escape character `TFESC`.

A program that wishes to receive and process KISS transmissions must thus initially wait for an `FEND` character. If the following character is anything other than an `FEND` character it begins to accumulate a frame; otherwise, it ignores consecutive `FEND`s until a frame begins. The program handles the command function code as necessary and then proceeds to accumulate characters until the frame is complete, as signalled by a second `FEND`. During this processing it must recognize and process escape sequences. The frame is now complete: the checksum is validated and the frame is handed to the host for processing.

This type of "multi-mode" processing is routinely implemented with a *finite state machine*, which switches processing modes to follow the protocol. The finite state machine used for processing KISS frames is illustrated in Figure 2.9. Several of the programs described in Appendix C use this state machine, implemented in a routine called `Protocol()`.

2.5 Modems & Modulation

The function of a *modem*, short for **mod**ulator-**dem**odulator, is to convert digital signals into an analogue form that may be transmitted over a communications channel: in this case, a radio system.

The feature that drives many modem designs and implementations is bandwidth, and this will be discussed in some detail. Several modulation schemes presently in use on amateur satellites will be discussed and their features will be compared.

2.7 Notes and References

1. *AX.25 Link Access Protocol for Amateur Packet Radio, Version 2.2*. Newington: ARRL and Tucson: TAPR. 1997.
   Internet: `http://www.tapr.org/tapr/pdf/AX25.2.2.pdf`

2. S. Wicker, *Error Control Systems for Digital Communication and Storage*. Upper Saddle River: Prentice Hall, 1995.

3. T. McDermott, *Wireless Digital Communications: Design and Theory*. Tucson: TAPR. 1996.

4. *Universal Serial Bus Specification*, Revision 1.1, 23 September 1998.
   Internet: `http://www.usb.org/developers/data/usbspec.zip`

5. J. Miller, "9600 Baud Packet Radio Modem Design", paper presented at ARRL 7th Computer Networking Conference, 1988.
   Internet: `http://www.amsat.org/amsat/articles/g3ruh/a109.txt`

6. *Spirit–2 Technical Reference Manual and Interfacing Instructions*, 3rd edition. Tampa: PacComm Packet Radio Systems Inc. 1995.

7. T. Sailer, "DSP Modems", paper presented at 11. Internationale Packet-Radio Tagung, Darmstadt, Germany, 1995. In German.
   Internet: `http://www.ife.ee.ethz.ch/˜sailer/ham/ham.html`.

8. M. Chepponis and P. Karn, "The KISS TNC: a Simple Host-to-TNC Communications Protocol", paper presented at ARRL 6th Computer Networking Conference, 1987.
   Internet: `http://people.qualcomm.com/karn/papers/kiss.html`

9. J. Romkey, *A Non-standard for Transmission of IP Datagrams over Serial Lines: SLIP*. RFC–1055, June 1988.
   Internet: `http://www.ietf.org/rfc/rfc1055.txt`

10. E. Lee and D. Messerschmitt, *Digital Communication*, 2nd edition. Boston: Kluwer Academic Publishers, 1994.

11. S. Haykin, *Communication Systems*, 3rd edition. New York: John Wiley & Sons. 1994

12. F. Kostedt and J. Kemerling, *Practical GMSK Data Transmission*, Winston-Salem: MX·COM, 1998.
    Internet: `http://www.mxcom.com/app_notes/apgmskr2.pdf`

13. S. Pasupathy, “Minimum shift keying—a spectrally efficient modulation”, *IEEE Communications Magazine*, 17:4, 1979.

14. A. Jones and and J. Gardiner, “Generation of GMSK using direct digital synthesis” in *IEE Colloquium on Implementations of Novel Hardware for Radio Systems*, 1992.

15. J. Miller, “The Shape of Bits to Come”, paper presented at ARRL 10th Computer Networking Conference, 1991.
Internet: `http://www.amsat.org/amsat/articles/g3ruh/a108.txt`

16. K. Murota and K. Hirade, “GMSK Modulation for Digital Mobile Radio Telephony”, *IEEE Trans. on Communications*, vol. COM–29:7, July 1981.

17. R. de Buda, “Coherent demodulation of frequency shift keying with low deviation ratio” in *IEEE Trans. on Communications*, vol. COM–20:6, June 1972.

18. M. Hodgart and J. Schoonees, “A Robust MSK Demodulator Through DSP” in *Proceedings of the 1992 South African Symposiom on Commmunications and Signal Processing*, 1992.

Communications Infrastructure for the MOST Microsatellite Project (excerpt)

Laura Halliday

Chapter 3

File transfer with PB and FTL0

3.1 Introduction

The unique file transfer protocols used by microsatellites are an attempt to make optimal use of an expensive resource: the satellite downlink.

The first digital amateur satellites used an adaptation of a terrestrial file transfer protocol. The key adaptation was addition of facilities to permit files to be transferred—in either direction—over several passes of the satellite. While this worked, it only allowed one ground station to access the satellite at one time. Other ground stations could hear the downlink transmissions from the satellite, but could not make use of them.

The solution was a *broadcast protocol.* The assumption, based on experience with early digital satellites and terrestrial networks, was that files were of general interest. By broadcasting a single copy of a file to all interested ground stations, system throughput and efficiency could be drastically increased.

The broadcast protocol may be characterized as a selective request repeat protocol. This is a protocol where ground stations initiate a file transfer, and then request retransmission of needed portions until the file is complete. The broadcast protocol also permits receive-only operation, where a passive station may receive files by merely listening to broadcasts initiated by others.

The satellite environment facilitates using different frequencies, in different amateur radio bands, for communications between ground stations and the satellites. This eliminates channel contention, a major consumer of channel capacity in terrestrial networks: terrestrial stations that wish to transfer files must share the channel with the files themselves.

This chapter will discuss the broadcast protocols in detail, examining the interaction between ground stations and satellites.

3.2 Terminology

The **P**ACSAT **B**roadcast Protocol, PB, is the protocol used for downloads on current satellites. **F**ile **T**ransfer **L**evel **0**, FTL0, was once the protocol for uploads and downloads, but is now only used for uploads.

In all discussions of the PACSAT broadcast protocol times are stored as 32 bit Unix times, the number of seconds since 0000 UTC 1 January 1970. All multi-byte quantities are stored least-significant byte first, "little-endian" byte order, as used by Intel microprocessors.

3.3 Downloading files with PB

The **P**ACSAT **B**roadcast protocol is the protocol currently in use for file transfer on the amateur satellites listed in Table 3.1 (1).

Table 3.1: Amateur satellites using PB

Satellite	**Downlink**
AO–16	1200 baud BPSK
UO–22	9600 baud MSK
KO–23	9600 baud MSK
KO–25	9600 baud MSK
TO–31	9600 baud MSK
UO–36	9600 and 38400 baud MSK

3.3.1 PACSAT File system

In order to support a large number of users with potentially disparate computer systems, the satellites use their own *PACSAT file system.* This is a very simple flat numbered file system. There is a single directory and the files are identified by a serial number, assigned by the file server at upload. This has proven adequate in experience, and is simple to implement with the limited computer resources available on the satellites. By adding additional information to files (PACSAT file headers, Section 3.3.2) it is possible for ground stations to select files of interest for download.

3.3.2 PACSAT file headers

To facilitate processing by both the satellite and ground stations, files have additional information added at creation time: the *PACSAT file header*.

The PACSAT file header provides information about the file: when it was uploaded, by whom, the type of data it contains, when it should be automatically deleted, and so forth. The header information is stored as a series of keys and values at the beginning of the file, before the user information. When a file is uploaded this information is added by the uploading software. When the file is downloaded this information is removed before making the file available to the user.

The PACSAT file header begins with a special marker `0xaa55`. It ends with an item whose header ID and length are 0, producing the bit pattern `0x000000`. In all header items the identification code is a 16 bit integer, and the length is an 8 bit unsigned integer. The header items listed in Table 3.2 are mandatory (2).

A sample PACSAT file header was obtained from AO–16, illustrated in Figure 3.1 and decoded in Table 3.3. The file is a satellite callsign log file generated on 8 December 1999. Note that this PACSAT file header does not conform to the PACSAT file header documentation (3), which states that header items must be provided in ascending numeric order. The position of item `0x12`, Upload Time (originally optional, now mandatory), violates this aspect of the standard.

Table 3.2: Mandatory PACSAT file header items

Code	**Length**	**Description**
`0x01`	4	File number
`0x02`	8	Text file name
`0x03`	3	Text file extension
`0x04`	4	File size in bytes
`0x05`	4	File creation time
`0x06`	4	Last modification time
`0x07`	1	Set if a **S**ingle-**E**vent **U**pset has occurred in the file
`0x08`	1	Type of data stored in the file
`0x09`	2	Body checksum: 16-bit XOR of all bytes in the file body
`0x0a`	2	Header checksum: 16-bit XOR of all bytes in the file header
`0x0b`	2	Location of user data in the file

```
aa 55 01 00 04 e0 21 00 00 02 00 08 43 4c 39 39
31 32 30 38 03 00 03 20 20 20 04 00 04 4e 0b 00
00 05 00 04 d5 b0 4d 38 06 00 04 d6 b0 4d 38 12
00 04 d6 b0 4d 38 07 00 01 00 08 00 01 d9 09 00
02 d4 99 0a 00 02 b6 0d 0b 00 02 50 00 00 00 00
23 30 54 4d 53 41 54 2d 31 00 20 20 20 20 50 59
33 50 51 00 00 4b 42 32 57 51 4d 00 56 45 32 4c
41 00 00 56 41 33 53 46 4c 00 4e 33 45 56 51 00
00 4b 38 54 4c 00 00 00 4e 56 38 46 00 00 00 4a
4a 33 59 55 4a 00 56 4b 33 4b 4f 53 00 56 4b 33
4a 44 47 00 4c 57 31 44 58 50 00 58 51 35 42 52
43 00 43 45 33 53 53 42 00 43 45 35 4e 47 00 00
41 44 34 45 42 00 00 4b 45 34 5a 58 57 00 4b 45
34 4b 4f 4c 00 4b 4e 34 57 5a 00 00 56 45 33 46
52 48 00 4e 38 58 4b 5a 00 00 56 45 33 42 43 47
00 57 42 38 48 52 4f 00 57 30 53 4c 00 00 00 4b
```

Figure 3.1: Sample PACSAT file header

Table 3.3: Decoded PACSAT file header

Raw value	Description	Engineering units
`0xaa55`	Header marker	
`0x0001 04 000021e0`	File number	
`0x0002 08 434c393931323038`	File name	“CL991208”
`0x0003 03 202020`	File extension	“ ”
`0x0004 04 00000be4`	File size	3028 bytes
`0x0005 04 384db0d5`	Creation time	031357 UTC 8 December 1999
`0x0006 04 384db0d6`	Last modification time	031358 UTC 8 December 1999
`0x0012 04 384db0d5`	Upload time	031358 UTC 8 December 1999
`0x0007 01 00`	SEU flag	None detected
`0x0008 01 d9`	File type	217 (callsign log)
`0x0009 02 99d4`	Body checksum	
`0x000a 02 0db6`	Header checksum	
`0x000b 02 0050`	Body offset	80 bytes
`0x000000`	End marker	

Table 3.4: PACSAT broadcast protocol frame header

Item	**Length**	**Description**
Flags	1	See Table 3.5.
File number	4	PACSAT file number
Offset	4	Location of first information byte
Time old	4	See text
Time new	4	See text

3.3.3 Broadcasts

The basic unit of communication between the satellite and ground stations is the *broadcast frame*. The satellite broadcasts to all ground stations within its footprint, and stations may use or ignore what they hear at their discretion. The information the satellite broadcasts is in response to requests transmitted by ground stations. The satellite broadcasts several kinds of information, which will be examined in turn.

3.3.4 Directory broadcasts

The first kind of broadcast from the satellite is a *directory broadcast.* This informs ground stations what files are available for download. A directory broadcast (4) consists of a frame header followed by a PACSAT file header (Section 3.3.2). It is transmitted as an AX.25 UI frame (Section 2.2) with a protocol id of `0xbd` and a destination address of `QST-1`.[1]

The frame header contains the information described in Table 3.4. The *flags* field contains several bit fields, as described in Table 3.5. The *time old* and *time new* fields identify possible different versions of the file. Given the upload time for the file, there are no other files other than the one specified by this particular file number with *time old* $\leq$ *upload time* $\leq$ *time new*.

The *offset* field may not have been implemented in the implementations studied: this value was zero in every directory broadcast received.

A sample directory broadcast was recorded from AO–16. The raw data are presented in Figure 3.2, and are decoded in Table 3.6. After the header information we note the `0xaa55` bit pattern denoting the beginning of a PACSAT file header.

Table 3.5: PACSAT broadcast *flags* field

Bits	**Description**
0–1	Frame type. `00` is a PFH broadcast frame. All others reserved.
2–3	Version identifier. Currently always `00`.
4	Always zero: server generated frame.
5	Set if the last byte of the frame is the last byte of the PFH.
6	Set if this is the newest file on the server.
7	Reserved.

```
20 67 AE 00 00 00 00 00 00 E0 7E 3C 38 EE CF 3D
38 AA 55 01 00 04 67 AE 00 00 02 00 08 42 4C 39
39 31 31 32 34 03 00 03 20 20 20 04 00 04 E0 06
00 00 05 00 04 49 2D 3B 38 06 00 04 E1 7E 3C 38
12 00 04 E0 7E 3C 38 07 00 01 00 08 00 01 CA 09
00 02 FB 44 0A 00 02 84 0C 0B 00 02 50 00 00 00
00 9D 3D
```

Figure 3.2: Sample PACSAT directory broadcast

Table 3.6: Decoded PACSAT directory broadcast

Item	**Raw value**	**Engineering units**
Flags	`0x20`	See Table 3.7
File number	`0x0000ae67`	
Offset	`0x00000000`	Not implemented?
Time old	`0x383c7ee0`	001216 UTC 25 November 1999
Time new	`0x383dcfee`	001022 UTC 26 November 1999

Table 3.7: Decoded PACSAT broadcast flags field

Bits	Description	Engineering units
`00`	Frame type	PFH broadcast frame
`00`	Version identifier	0
`0`	Frame origin	Server generated frame
`1`	Last byte in PFH?	Yes
`0`	Newest file on server?	No
`0`	Reserved	

The *flags* value in Table 3.6 decodes to the information shown in Table 3.7.

3.3.5 File broadcasts

When a ground station requests a file for download it is transmitted in a series of *file broadcasts.* These consist of a series of pieces of the file, with control information so that ground stations may reassemble them. Like all broadcasts in the protocol, these transmissions may be received by any interested ground station—even a passive one that makes no requests at all.

File broadcasts are addressed to `QST-1`, and use AX.25 protocol `0xbb`.

A file broadcast packet has minimal control information, followed by the file data. The elements of a file broadcast packet are listed in Table 3.8. A number of options are defined for the Flags field, but the only one used in present implementations is the bit specifying the piece location within the file by byte count, rather than by block count. The Flags field was thus always `0x02` in all received packets.

Sample packets were obtained from AO–16 broadcasts. Figure 3.3 illustrates the broadcast of the first piece of a file, and it is decoded in Table 3.9. Note that the first bytes following the broadcast header are `0xaa55`, the PACSAT file header. Figure 3.4 illustrates the broadcast of a piece from the interior of a file, and it is decoded in Table 3.10.

[1] *QST* is a signal used in the amateur radio service for information of interest to all radio amateurs.

Table 3.8: Data elements in a file broadcast packet

Item	**Length**	**Description**
Flags	1	Bit field. Always `0x02`.
File id	4	PACSAT file number
File type	1	PACSAT file type
Offset	2	Offset of this piece in the file: low-order 16 bits
Offset MSB	1	High-order 8 bits of offset

```
02 7E AE 00 00 C9 00 00 00 AA 55 01 00 04 7E AE
00 00 02 00 08 41 4C 39 39 31 31 32 39 03 00 03
20 20 20 04 00 04 C1 03 00 00 05 00 04 64 C5 41
38 06 00 04 5B FC 41 38 12 00 04 5A FC 41 38 07
00 01 00 08 00 01 C9 09 00 02 EB AD 0A 00 02 88
0D 0B 00 02 50 00 00 00 00 12 19 5F C5 41 38 00
00 00 A0 67 00 00 31 00 00 00 1C 17 00 00 EE 9D
29 2F 12 19 6E C5 41 38 00 00 00 CF 9F 00 00 30
00 00 00 1B 17 00 00 C2 C0 D3 34 12 19 72 C5 41
38 00 00 00 A6 A1 00 00 30 00 00 00 1B 17 00 00
04 AB 2B 35 12 19 75 C5 41 38 00 00 00 CF A2 00
00 30 00 00 00 1B 17 00 00 69 6F 63 35 12 19 79
C5 41 38 00 00 00 68 A6 00 00 30 00 00 00 1B 17
00 00 C3 B4 37 36 11 19 7F C5 41 38 00 00 00 5A
AB 00 00 30 00 00 00 1B 17 00 00 B6 02 7C 37 11
19 84 C5 41 38 00 00 00 5B AB 00 00 31 EE 27
```

Figure 3.3: Sample PACSAT file broadcast: beginning of file

Table 3.9: Decoded file broadcast packet: beginning of file

Item	**Raw data**	**Engineering units**
Flags	`0x02`	
File id	`0x0000ae7e`	
File type	`0xc9`	201 (activity log)
Offset	`0x0000`	File offset 0
Offset MSB	`0x00`	(first piece)

```
02 7E AE 00 00 C9 E8 01 00 19 FF DF 41 38 C1 8E
00 8A AC 00 00 00 00 00 00 0C 00 00 00 62 0A 00
00 04 15 05 E0 41 38 C1 8E 03 04 00 00 00 00 00
00 00 00 00 00 00 03 10 1B E0 41 38 C2 8E 03 43
58 36 44 44 00 00 0C 19 22 E0 41 38 C2 8E 00 8A
AC 00 00 00 00 00 00 0C 00 00 00 62 0A 00 00 04
15 23 E0 41 38 C2 8E 03 04 00 00 00 00 00 00 00
00 00 00 00 03 10 3C E0 41 38 C3 8E 03 43 58 36
44 44 00 00 0C 19 59 E0 41 38 C3 8E 00 8A AC 00
00 00 00 00 00 0C 00 00 00 62 0A 00 00 04 15 5C
E0 41 38 C3 8E 03 04 00 00 00 00 00 00 00 00 00
00 00 03 10 77 E0 41 38 C4 8E 03 43 58 36 44 44
00 00 0C 19 78 E0 41 38 C4 8E 00 8A AC 00 00 00
00 00 00 0C 00 00 00 62 0A 00 00 04 15 81 E0 41
38 C4 8E 03 04 00 00 00 00 00 00 00 00 00 00 00
09 11 7B E4 41 38 00 00 00 2E 17 00 00 9C 48
```

Figure 3.4: Sample PACSAT file broadcast: file interior

Table 3.10: Decoded file broadcast packet: file interior

Item	**Raw data**	**Engineering units**
Flags	`0x02`	
File id	`0x0000ae7e`	
File type	`0xc9`	201 (activity log)
Offset	`0x01e8`	File offset 488
Offset MSB	`0x00`	

Table 3.11: Destination callsign usage in PB

Callsign	**Meaning**
`BBSTAT`	General status information.
`PBLIST`	Unlisted calls are invited to make requests.
`PBFULL`	Queue full: do not transmit.
`PBSHUT`	Satellite closed.

3.3.6 Status transmissions

Satellites broadcast status information so that ground stations may decide when to download files. The content and meaning of status messages is documented as part of the broadcast protocol.

Status broadcasts advise if the satellite is available for file transfers. They advise if the broadcast queue is full or empty, and if it is not full, who is in it. They also advise who is using the satellite uplink frequencies, particularly important if the satellite has more than one uplink receiver. All these transmissions are AX.25 UI frames, addressed to different destination callsigns for different broadcast functions, as summarized in Table 3.11. The status broadcasts are human-readable, but are typically interpreted by groundstation software.

The first status message a ground station must receive is the satellite status. This advises that the satellite is open for business:

```
Open ABCD:
```

The code `ABCD` shows that the satellite[2] has four uplink frequencies, and that none of them are in use. To assist ground stations in selecting uplink frequencies, the satellite advises which frequency a ground station is using for FTL0 file transfers:

```
Open A CD: N0ALJ
```

Once the `Open` message has been received a ground station may transmit a request to the satellite. The satellite acknowledges the request with the only non-broadcast message in the protocol:

[2]The example is based on data received from AO–16.

```
OK VA3SFL
```

If the satellite cannot fulfil the request it issues an error message instead:

```
NO -2 VA3SFL
```

A common error is a request for a non-existent file. The broadcast queue may be full.

While file transfers are in progress the satellite advises ground stations of the broadcasts it has prepared. The format is a list of the callsigns for which a broadcast is queued. The `\D` notation indicates a directory broadcast for the station:

```
PB: N0ALJ N5ZNL VA3SFL\D W4SM\D
PB: W4SM N5ZNL VA3SFL
```

If the broadcast queue is empty the satellite advises ground stations of the fact:

```
PB: Empty
```

The satellite may be closed for uploads:

```
Shut: ABCD
```

An additional useful transmission is the `B:` message:

```
B: 1691718618
```

This is a running count of bytes transmitted by the spacecraft file server. While it is not possible to evaluate bit error rate under AX.25, it is still possible to compare the values in the `B:` message to the byte count received by the ground station and thus evaluate link quality. WiSP presents this information as a percentage efficiency.

Table 3.12: File fill request format

Component	**Item**	**Size**
Header	flags	1 byte
	file id	4 bytes
	block size	2 bytes
Hole list	offset	2 bytes
(repeated as necessary)	offset msb	1 byte
	length	2 bytes

Table 3.13: File fill request *flags* format

Bit number	**Item**
0–1	`00`: start sending file
	`01`: frame contains a hole list
2–3	Version, currently `00`
4	Reserved, must be `1`
5–7	Reserved, must be `000`

3.3.7 Fill requests

Ground stations transmit *fill requests* to advise the satellite of their wishes. A ground station may request a file fill or a directory fill. The formats for both requests are the same.

A station that wishes to make a request of the server transmits the request as an AX.25 UI frame, using protocol id `0xbb` for file requests and `0xbd` for directory requests, the same as the broadcast messages from the satellite. The format of a file fill request is illustrated in Table 3.12. The format of a directory fill request is illustrated in Table 3.14. Note the similarity of the two formats.

The block size parameter is fixed at 244 bytes, resulting in a total AX.25 frame length of 255 bytes, the maximum length that current satellite AX.25 implementations can handle. The hole list is the gaps in the file that the ground station wishes to fill.

The *flags* field in a file fill request frame is defined in Table 3.13.

Table 3.14: Directory fill request format

Component	**Item**	**Size**
Header	flags	1 byte
	file id	4 bytes
	block size	2 bytes
Hole list	start time	4 bytes
(repeated as necessary)	end time	4 bytes

Table 3.15: Directory fill request flags format

Bit number	**Item**
0–1	`00`: Directory fill request
2–3	Version, currently `00`
4	always `1` indicating a client-generated frame
5–7	Reserved, must be `000`

In the case of a directory fill the "holes" are time intervals in which the ground station has no information on files.

The *flags* field in a directory fill request frame is defined in Table 3.15.

3.3.8 An example download sequence

To illustrate how the different components of the PACSAT broadcast protocol interact, here is a sample download sequence, from initial acquisition of the satellite to the file being complete at the ground station.

In this sample the ground station uses the callsign `VA3SFL`, while the satellite callsign is `MST-SAT`. The satellite broadcast callsign is by convention `MSTSAT-11`, while the BBS (**B**ulletin **B**oard **S**ystem—a holdover from early satellites) callsign is then `MSTSAT-12`. By convention, the ground station SSID is 0, resulting in `VA3SFL-0` as the complete ground station callsign.

The ground station is initially idle. It waits for a status indication from the satellite.

The ground station receives a message:

> From `MSTSAT-12` to `BBSTAT-0`, “`Open:  AB:`”

The satellite is now available, and the ground station will now typically request a directory fill. The satellite acknowledges the request:

> From `MSTSAT-11` to `VA3SFL-0`, “`OK VA3SFL`”.

The file server adds a directory broadcast to the broadcast queue, and adds the information to the broadcast message:

> From `MSTSAT-11` to `PBLIST-0`, “`PB: VA3SFL\D`”.

The PB message is likely to contain the callsigns of other stations accessing the satellite.

The ground station receives the directory update, applies download rules (or acts on human instructions) and starts to download file `A123`. It issues a request for the satellite to start transmitting the file, which causes the satellite to issue an acknowledgement:

> From `MSTSAT-11` to `VA3SFL-0`, “`OK: VA3SFL`”.

The satellite then queues the broadcast of the first piece of the file and updates the broadcast message:

> From `MSTSAT-11` to `PBLIST-0`, “`PB: VA3SFL`”.

Again,other stations may be in the broadcast queue. The satellite will proceed to broadcast information queued in response to requests by other ground stations. The ground station will receive these transmissions and add them to its own files.

When the broadcast request is at the head of the queue the satellite transmits a piece of the file as several frames, each from `MSTSAT-11` to `QST-0`. Other ground stations in the satellite’s footprint receive the pieces of the file.

The ground station notes from the PACSAT file header that the file is of length 2048 bytes and reserves storage accordingly. After its turn in the broadcast queue has expired it notes that it has received bytes 1 to 512 and 1024 to 1512. The ground station issues a fill request for file `A123`, bytes 513 to 1023, and 1513 to 2048. The satellite doesn’t hear the request due to another station issuing a request at the same time. The ground station times out and repeats the request. The satellite responds:

From `MSTSAT-11` to `VA3SFL-0`, “`OK: VA3SFL`”.

The satellite then adds a broadcast of the pieces of file `A123` to the broadcast queue, and updates the broadcast queue message:

From `MSTSAT-11` to `PBLIST-0`, “`PB: VA3SFL`”.

When the broadcast reaches the head of the queue the satellite broadcasts the requested pieces. The ground station notes that the file is now complete, saves it for further processing, and moves on to the next file of interest.

Note that this exchange is public. A receive-only station could have learned of the file and received it by merely listening to the messages from `MSTSAT-11` to `QST-0`.

3.4 Uploading files with FTL0

While FTL0 is obsolete for downloading files, it is still the current protocol for uploading files. This section will thus only discuss the use of FTL0 for uploads.

Unlike PB, which uses AX.25 UI packets, FTL0 uses AX.25 in connected mode. There are thus procedures for logging in, logging out, and the notion of a session. FTL0 itself is simple, but it uses the facilities of a complex underlying protocol. This is conceptually similar to the FTP protocol in the TCP/IP protocol suite.

3.4.1 Packet format

FTL0 defines a standard format for packets. They consist of a command code, a data length, and data, as applicable, in the format described in Tables 3.16 and 3.17. The original FTL0 specification documents packet types relating to downloading files, but these are obsolete and are not documented here.

3.6 Notes and references

1. Source: AMSAT News Service. List current as of December 1999. UO–36 was still undergoing testing and on-orbit checkout.
   Internet: `http://www.amsat.org/amsat/news/ans.html`

2. J. Ward and H. Price, "PACSAT File Header Definition", paper presented at ARRL 9th Computer Networking Conference 1990.
   Internet: `ftp://ftp.amsat.org/amsat/satinfo/pacsat/pacdoc.zip`

3. J. Ward and H. Price, "Regular Broadcasting of PACSAT File Headers", presented at the ARRL 9th Computer Networking Conference 1990. Despite the title, this is the documentation for the broadcast protocol in its present form.
   Internet: `ftp://ftp.amsat.org/amsat/satinfo/pacsat/pacdoc.zip`

Communications Infrastructure for the MOST Microsatellite Project (excerpt)

Laura Halliday

Table 4.7: UoSAT–3 whole-orbit data format header items

Item	**Length**	**Interpretation**
Start time	4 bytes	32 bit Unix time
End time	4 bytes	32 bit Unix time
Sample period	2 bytes	sample period in seconds
Number of channels	1 byte	0 to 255

4.3 Whole orbit data

While telemetry beacons can transmit a great deal of information, they may only be received while the satellite is within radio range of a ground station. When details of satellite functions are required for an extended period the satellite gathers its measurements into *whole orbit data*.

In this section we will examine several formats for whole orbit data: UoSAT–3 format, extended UoSAT–3 format, PACSAT format, and UO–11 format. All will be illustrated with samples obtained from satellites.

4.3.1 UoSAT–3 whole-orbit data format

The whole-orbit data format used by many microsatellites today originated with the UoSAT–3 project, launched as UOSAT–OSCAR–14 on 22 February 1990. The system gathered telemetry observations and saved them in files for later download. The format of the files is straightforward and consists of three sections (8):

- File header, including start and end times and sample period.
- List of telemetry channels.
- The telemetry observations themselves.

The file header consists of the items in Table 4.7. Like all items in the UoSAT telemetry systems, multi-byte quantities are stored least-significant byte first. “32 bit Unix time” is the number of seconds since 0000 UTC 1 January 1970.

The header is then followed by the list of channel numbers, each stored as an unsigned byte.

Table 4.8: Decoded header information in a sample UO–22 whole-orbit data file

Parameter	**Raw value**	**Engineering units**
Start time	`0x383dcd85`	000005 UTC 26 November 1999
End time	`0x383e7622`	115930 UTC 26 November 1999
Sample period	`0x001e`	30 seconds
Number of channels	`0x13`	19 channels

The data samples follow the header information. Each is stored as an unsigned 16-bit integer. The underlying data type is actually an unsigned 12-bit quantity; the remaining four bits are used for control functions in the telemetry beacon, but are unused in whole orbit data. The values are recorded in the same order that was stated in the file header.

4.3.2 Sample UoSAT–3 format whole orbit data file

A sample whole-orbit data file was downloaded from UO–22. We may read off the header information in from the first few bytes of the file in Figure 4.5 to produce Table 4.8.

```
85 cd 3d 38 22 76 3e 38 1e 00 13 00 08 10 1a 01
0b 03 06 21 31 11 3c 27 2f 37 15 22 2a 2b 04 00
07 07 05 00 05 00 ad 0b 92 06 aa 02 b8 02 98 03
80 00 a2 0c c4 04 7b 06 0c 09 c0 06 d7 02 75 06
50 07 90 09 04 00 fc 06 05 00 05 00 b7 0b 95 06
aa 02 b7 02 98 03 80 00 a2 0c c9 04 c5 06 61 09
d4 06 d7 02 71 06 36 07 c3 09 04 00 f2 06 05 00
05 00 a7 0b 95 06 aa 02 b7 02 98 03 80 00 a2 0c
```

Figure 4.5: Header information in a UO–22 whole-orbit data file

The list of channels follows, as illustrated in Figure 4.6 and as listed in Table 4.9.

```
85 cd 3d 38 22 76 3e 38 1e 00 13 00 08 10 1a 01
0b 03 06 21 31 11 3c 27 2f 37 15 22 2a 2b 04 00
07 07 05 00 05 00 ad 0b 92 06 aa 02 b8 02 98 03
80 00 a2 0c c4 04 7b 06 0c 09 c0 06 d7 02 75 06
50 07 90 09 04 00 fc 06 05 00 05 00 b7 0b 95 06
aa 02 b7 02 98 03 80 00 a2 0c c9 04 c5 06 61 09
d4 06 d7 02 71 06 36 07 c3 09 04 00 f2 06 05 00
05 00 a7 0b 95 06 aa 02 b7 02 98 03 80 00 a2 0c
```

Figure 4.6: Channel list in a sample UO–22 whole-orbit data file

Table 4.9: Decoded channel list in a UO–22 whole orbit data file

Raw value	Decimal	Channel Description
`0x00`	0	Array current +X
`0x08`	8	Array current -X
`0x10`	16	Array current +Y
`0x1a`	26	Array current -Y
`0x01`	1	Array voltage
`0x0b`	11	Battery current
`0x03`	3	14 volt bus current
`0x06`	6	Battery temperature
`0x21`	33	Transmitter 0 forward power
`0x31`	49	Transmitter 0 reverse power
`0x11`	17	Battery voltage
`0x3c`	60	OBC186 CPU current
`0x27`	39	Magnetometer 1 X value
`0x2f`	47	Magnetometer 1 Y value
`0x37`	55	Magnetometer 1 Z value
`0x15`	21	Transmitter 1 temperature
`0x22`	34	Receiver 0 received signal strength
`0x2a`	42	Receiver 1 received signal strength
`0x2b`	43	Receiver 1 discriminator voltage

We may then read off the first telemetry sample in the file, as shown in Figure 4.7 and decoded in Table 4.10. Note that not all values decode to "natural" units: some, like the magnetometer values and received signal strength, merely decode to the voltage measured by a sensor.

```
85 cd 3d 38 22 76 3e 38 1e 00 13 00 08 10 1a 01
0b 03 06 21 31 11 3c 27 2f 37 15 22 2a 2b 04 00
07 07 05 00 05 00 ad 0b 92 06 aa 02 b8 02 98 03
80 00 a2 0c c4 04 7b 06 0c 09 c0 06 d7 02 75 06
50 07 90 09 04 00 fc 06 05 00 05 00 b7 0b 95 06
aa 02 b7 02 98 03 80 00 a2 0c c9 04 c5 06 61 09
d4 06 d7 02 71 06 36 07 c3 09 04 00 f2 06 05 00
05 00 a7 0b 95 06 aa 02 b7 02 98 03 80 00 a2 0c
```

Figure 4.7: First sample in a UO–22 whole-orbit data file

4.3.3 Extended UoSAT whole orbit data format

The extended whole orbit data format is derived from the UoSAT–3 format. Like the UoSAT–3 format, the extended format (currently used on TO–31) consists of a header, a list of channels and the telemetry observations. The difference is the expansion of several values to 16 bit quantities, to allow for more than 256 channels of telemetry. Additionally, each telemetry observation is explicitly timestamped, unlike the older format that requires observation times to be inferred.

Attempts to obtain documentation on the format failed, so the information that follows was obtained by reverse-engineering telemetry files downloaded from TO–31.

Analysis showed that the file header contains the following information:

- 7 bytes of unknown function. The value was `0x8134010001be00` in all available files.
- The satellite name, a null-terminated ASCII string padded with `0x00` to 12 bytes.
- A constant byte `0x01`.
- The file description, a null-terminated ASCII string padded with `0x00` to 30 bytes.
- The time of the beginning of the data collection.
- A 16 bit constant `0x0000`.
- The time of the end of the data collection.

Table 4.10: Decoded telemetry observation in a UO–22 whole orbit data file

Channel Description	**Raw value**	**Engineering units**
Array current +X	`0x0004`	7.630487 mA
Array current -X	`0x0707`	532.8304 mA
Array current +Y	`0x0005`	0.5139264 mA
Array current -Y	`0x0005`	7.293801 mA
Array voltage	`0x0bad`	40.62266 V
Battery current	`0x0692`	682.3492 mA
14 volt bus current	`0x02aa`	558.8579 mA
Battery temperature	`0x02b8`	45.61132 C
Transmitter 0 forward power	`0x0398`	1.899261 W
Transmitter 0 reverse power	`0x0080`	0.15648 W
Battery voltage	`0x0ca2`	13.60162 V
OBC186 CPU current	`0x04c4`	117.6798 mA
Magnetometer 1 X value	`0x067b`	2.028127 V
Magnetometer 1 Y value	`0x090c`	2.83131 V
Magnetometer 1 Z value	`0x06c0`	2.11248 V
Transmitter 1 temperature	`0x02d7`	43.37684 C
Receiver 0 received signal strength	`0x0675`	2.020792 V
Receiver 1 received signal strength	`0x0750`	2.28852 V
Receiver 1 discriminator voltage	`0x0990`	2.99268 V

Table 4.11: Decoded header information in a TO–31 whole-orbit data file

Parameter	**Value**	**Engineering units**
Satellite name	`0x544d5341...`	“TMSAT-1”
File description	`0x486f7573...`	“Housekeeping WOD”
Start time	`0x38411942`	120002 UTC 28 November 1999
End time	`0x3841c1e2`	235930 UTC 28 November 1999
Sampling interval	`0x001e`	30 seconds
Number of channels	`0x0014`	20 channels

- A 16 bit constant `0x0000`.
- The sampling interval, a 16 bit integer, in seconds.
- A 32 bit constant `0x00000000`.
- The number of telemetry channels, a 16 bit integer.

The large number of constants in the file header suggests that the format has many other possible functions and parameters which are not currently being used.

A sample telemetry file was downloaded from TO–31. We may read off the header values from Figure 4.8 to produce Table 4.11.

```
81 34 01 00 01 be 00 54 4d 53 41 54 2d 31 00 00
00 00 00 01 48 6f 75 73 65 6b 65 65 70 69 6e 67
20 57 4f 44 00 00 00 00 00 00 00 00 00 00 00 00
00 00 42 19 41 38 00 00 e2 c1 41 38 00 00 1e 00
00 00 00 00 14 00 02 00 11 00 00 02 02 00 0b 00
00 02 02 00 0d 00 00 02 02 00 01 00 00 02 02 00
13 00 00 02 02 00 0e 00 00 02 02 00 26 00 00 02
02 00 04 00 00 02 02 00 14 00 00 02 02 00 08 00
00 02 02 00 1a 00 00 02 02 00 29 00 00 02 02 00
38 00 00 02 02 00 22 00 00 02 02 00 2a 00 00 02
02 00 32 00 00 02 02 00 1c 00 00 02 02 00 0f 00
00 02 02 00 17 00 00 02 02 00 07 00 00 02 43 19
41 38 00 00 01 0d 8f 07 2c 04 13 0c 2e 05 23 00
0b 06 11 05 2d 05 1d 00 94 01 02 02 6e 00 9a 05
d7 07 49 07 e6 03 bd 08 19 07 2d 06 61 19 41 38
00 00 ff 0c 9b 07 2a 04 a4 0b 1c 05 2a 00 c8 04
```

Figure 4.8: Header information in a TO–31 whole-orbit data file

We may then read off the list of channels from Figure 4.9 to produce the list in Table 4.12. Each entry in the list is accompanied by two sets of flags, `0x0200` and `0x0002`, whose function was not determined.

```
81 34 01 00 01 be 00 54 4d 53 41 54 2d 31 00 00
00 00 00 01 48 6f 75 73 65 6b 65 65 70 69 6e 67
20 57 4f 44 00 00 00 00 00 00 00 00 00 00 00 00
00 00 42 19 41 38 00 00 e2 c1 41 38 00 00 1e 00
00 00 00 00 14 00 02 00 11 00 00 02 02 00 0b 00
00 02 02 00 0d 00 00 02 02 00 01 00 00 02 02 00
13 00 00 02 02 00 0e 00 00 02 02 00 26 00 00 02
02 00 04 00 00 02 02 00 14 00 00 02 02 00 08 00
00 02 02 00 1a 00 00 02 02 00 29 00 00 02 02 00
38 00 00 02 02 00 22 00 00 02 02 00 2a 00 00 02
02 00 32 00 00 02 02 00 1c 00 00 02 02 00 0f 00
00 02 02 00 17 00 00 02 02 00 07 00 00 02 43 19
41 38 00 00 01 0d 8f 07 2c 04 13 0c 2e 05 23 00
0b 06 11 05 2d 05 1d 00 94 01 02 02 6e 00 9a 05
d7 07 49 07 e6 03 bd 08 19 07 2d 06 61 19 41 38
00 00 ff 0c 9b 07 2a 04 a4 0b 1c 05 2a 00 c8 04
```

Figure 4.9: List of channels in a TO–31 whole-orbit data file

The first sample from Figure 4.10 is then decoded to produce Table 4.13.

```
81 34 01 00 01 be 00 54 4d 53 41 54 2d 31 00 00
00 00 00 01 48 6f 75 73 65 6b 65 65 70 69 6e 67
20 57 4f 44 00 00 00 00 00 00 00 00 00 00 00 00
00 00 42 19 41 38 00 00 e2 c1 41 38 00 00 1e 00
00 00 00 00 14 00 02 00 11 00 00 02 02 00 0b 00
00 02 02 00 0d 00 00 02 02 00 01 00 00 02 02 00
13 00 00 02 02 00 0e 00 00 02 02 00 26 00 00 02
02 00 04 00 00 02 02 00 14 00 00 02 02 00 08 00
00 02 02 00 1a 00 00 02 02 00 29 00 00 02 02 00
38 00 00 02 02 00 22 00 00 02 02 00 2a 00 00 02
02 00 32 00 00 02 02 00 1c 00 00 02 02 00 0f 00
00 02 02 00 17 00 00 02 02 00 07 00 00 02 43 19
41 38 00 00 01 0d 8f 07 2c 04 13 0c 2e 05 23 00
0b 06 11 05 2d 05 1d 00 94 01 02 02 6e 00 9a 05
d7 07 49 07 e6 03 bd 08 19 07 2d 06 61 19 41 38
00 00 ff 0c 9b 07 2a 04 a4 0b 1c 05 2a 00 c8 04
```

Figure 4.10: First telemetry sample in a TO–31 whole-orbit data file

Table 4.12: Decoded channel list in a TO–31 whole orbit data file

Raw value	**Decimal**	**Channel Description**
`0x0200 1100 0002`	17	Battery Voltage
`0x0200 0b00 0002`	11	Battery Current
`0x0200 0d00 0002`	13	Battery Temp
`0x0200 0100 0002`	1	Array Voltage
`0x0200 1300 0002`	19	PCM Input Curr
`0x0200 0e00 0002`	14	+14V Line Curr
`0x0200 2600 0002`	38	+5V Line Curr
`0x0200 0400 0002`	4	-X Panel Temp
`0x0200 1400 0002`	20	-Y Panel Temp
`0x0200 0800 0002`	8	Array Curr -X
`0x0200 1a00 0002`	26	Array Curr -Y
`0x0200 2900 0002`	41	Tx0 Forward (W)
`0x0200 3800 0002`	56	Tx0 Reverse (V)
`0x0200 2200 0002`	34	Rx0 RRSI (dBm)
`0x0200 2a00 0002`	42	Rx1 RRSI (dBm)
`0x0200 3200 0002`	50	Rx2 RRSI (dBm)
`0x0200 1c00 0002`	28	Tx0 Temp
`0x0200 0f00 0002`	15	NavMag0 Xdir
`0x0200 1700 0002`	23	NavMag0 Ydir
`0x0200 0700 0002`	7	NavMag0 Zdir

Table 4.13: Decoded telemetry observation in a TO–31 whole orbit data file

Channel Description	**Raw value**	**Engineering units**
Timestamp	`0x38411943`	120003 UTC 28 November 1999
Filler	`0x0000`	
Battery Voltage	`0x0d01`	13.95365 V
Battery Current	`0x078f`	232.0696 mA
Battery Temperature	`0x042c`	16.43892 C
Array Voltage	`0x0c13`	41.82845 V
PCM Input Currrent	`0x052e`	423.6486 mA
+14V Line Current	`0x0023`	14.21373 mA
+5V Line Current	`0x060b`	955.6439 mA
-X Panel Temperature	`0x0511`	-0.5762487 C
-Y Panel Temperature	`0x052d`	-2.656706 C
Array Current -X	`0x001d`	12.9023 mA
Array Current -Y	`0x0194`	110.4042 mA
Tx0 Forward Power	`0x0202`	2.0042 W
Tx0 Reverse Power	`0x006e`	0.134376 W
Rx0 RRSI	`0x059a`	-97.08976 dBm
Rx1 RRSI	`0x07d7`	-110.8361 dBm
Rx2 RRSI	`0x0749`	-116.5249 dBm
Tx0 Temperature	`0x03e6`	21.64006 C
NavMag0 Xdir	`0x08bd`	6.758615 V
NavMag0 Ydir	`0x0719`	8.391602 V
NavMag0 Zdir	`0x062d`	15.95264 V

Table 4.14: Channel numbers in AO–16 WOD survey

Channel number	**Channel description**
`0x26`	-X array current
`0x27`	+X array current
`0x28`	-Y array current
`0x29`	+Y array current
`0x2B`	+Z array current
`0x2D`	BCR (**B**attery **C**harge **R**egulator) input current

4.3.4 AO–16/PACSAT whole orbit data

AO–16's on-board systems predate the now commonly used UoSAT telemetry formats. Since the satellite has worked very well (it ran 1910 days—over 5 *years*—until the onboard computer system went into safe mode on 12 December 1999), the command team have opted to leave it as is and not update the telemetry system (9).

AO–16 uses a unique system for gathering and disseminating whole-orbit data. Unlike newer satellites that gather whole-orbit data continuously, AO–16 only gathers data when instructed to do so by a command station, and then only does it for one orbit. It doesn't place the data in the satellite file system, but instead broadcasts the data on the downlink in a series of special messages. The format of the downlink messages was easily determined by inspection.

There are two kinds of downlink messages involved in the dissemination of whole-orbit data on AO–16. The first message is a simple text listing of telemetry channels, an AX.25 UI frame addressed to `WODCH-0`:

```
    WOD:262728292B2D
```

This messages states that a whole-orbit data survey is being broadcast on the downlink, and that the survey includes the telemetry channels decoded in Table 4.14.

The second message transmits the actual telemetry data in AX.25 UI frames addressed to `WOD-0`. Observations are sent in order with a timestamp for each observation. Each timestamp is a 32 bit Unix time and is stored least-significant byte first. A sample WOD packet is illustrated in Figure 4.11.

```
AC AE 02 38 01 6C 01 00 15 66 B6 AE 02 38 00 64
14 00 18 72 C0 AE 02 38 04 5B 34 00 16 7B CA AE
02 38 05 41 50 01 14 84 D4 AE 02 38 03 0D 6B 04
15 6E DE AE 02 38 1F 01 6B 02 16 80 E8 AE 02 38
55 01 51 01 16 8B F2 AE 02 38 6C 00 26 04 16 7B
FC AE 02 38 6D 00 01 00 19 77 06 AF 02 38 6E 04
00 1B 17 81 10 AF 02 38 5E 03 05 50 19 8B 1A AF
02 38 30 03 02 78 18 85 24 AF 02 38 01 00 01 84
1C 73 2E AF 02 38 02 36 04 6E 1D 8C 38 AF 02 38
00 5A 02 47 1C 89 42 AF 02 38 03 6D 04 21 1E 85
4C AF 02 38 02 7B 06 00 1C 75 56 AF 02 38 07 72
36 00 1C 90 60 AF 02 38 06 46 5E 02 1C 90 6A AF
02 38 01 11 73 02 1D 75 74 AF 02 38 11 00 74 00
1E 77 7E AF 02 38 49 01 60 01 1D 8C 88 AF 02 38
6A 01 41 02 19 8E 92 AF 02 38 81 01 16 01 1C 7E
9C AF 02 38 84 02 01 15 1A 7B
```

Figure 4.11: AO–16 whole-orbit data transmission

Table 4.15: Decoded AO–16 WOD sample

Channel description	**Raw value**	**Engineering units**
Timestamp	`0x3802AEAC`	0344444 UTC 12 October 1999
-X array current	`0x01`	-7.5 mA
+X array current	`0x6c`	217.6 mA
-Y array current	`0x01`	-1.7 mA
+Y array current	`0x00`	129.5 mA
+Z array current	`0x15`	26.4 mA
BCR input current	`0x66`	438.1 mA

Each packet in this survey consists of a sequence of 15 observations of 10 bytes each, 4 bytes for the timestamp, followed by 6 bytes of telemetry data. The first telemetry observation in the sample packet is:

```
AC AE 02 38 01 6C 01 00 15 66
```

We may decode this observation, producing the values in Table 4.15.

4.5 Notes and references

1. M. Davidoff, *The Satellite Experimenter's Handbook*, 2nd edition. Newington: ARRL, 1990.

2. M. Davidoff, *The Radio Amateur's Satellite Handbook*, Newington: ARRL. 1998.

3. A. Mironov, RS–12/13 command station, distributed by P. Gowen, AMSAT-UK.
   Internet: `http://oeonline.com/˜kf8bex/rs12-tlm.txt`

4. T. G. Jeans and C. P. Traynar, "The Primary UOSAT Spacecraft Computer" in *The Radio and Electronic Engineer*, 52:8/9, August/September 1982.

5. L. S. A. Mansi and R. A. Clarke, "The UOSAT Telemetry System" in *The Radio and Electronic Engineer*, 52:8/9, August/September 1982.

6. `u2tm`, distributed by C. Wallis, AMSAT–UK.
   Internet: `http://www.users.zetnet.co.uk/clivew/u2tm.zip`

7. *Telemetry on the UoSAT–3 PACSAT Communications Experiment*, UoSAT Spacecraft Data Sheet, University of Surrey. 1990?
   Internet: `http://www.ee.surrey.ac.uk/CSER/UOSAT/amateur/TLMFormat.txt`

8. *WOD on the UOSAT–3 PACSAT Communications Experiment*, UoSAT Spacecraft Data Sheet, University of Surrey. 1990?
   Internet: `http://www.ee.surrey.ac.uk/CSER/UOSAT/amateur/WODFormat.txt`

9. J. White, AMSAT–NA, personal communication.

10. C. Wallis, *OSCAR–11 WOD*, AMSAT–UK.
    Internet: `http://www.users.zetnet.co.uk/clivew/u2awod.zip`

Communications Infrastructure for the MOST Microsatellite Project (excerpt)

Laura Halliday

7.4 Telemetry analysis

Telemetry was downloaded from satellites and analyzed. The results illustrate many of the issues involved in the design, construction and operation of a satellite.

The whole orbit data results presented in this section were downloaded in encoded format from TO–31 and UO–22 in the formats described in Sections 4.3.1 and 4.3.3. They were decoded with the display program `vbtlm` described in Appendix C.2, producing comma-delimited text files which were then imported into Microsoft Excel to produce the plots presented here.

A whole orbit data survey was obtained from AO–16 in the format described in Section 4.3.4 and is presented here. Other analyses presented here cover task messages and log files.

7.4.1 Whole orbit data: TO–31

The results presented in Figures 7.6 to 7.9 were produced by decoding a TO–31 whole-orbit data file that covers satellite activity from 1200 UTC to 2359 UTC on 28 November 1999.

Figure 7.6 clearly shows the thermal environment of the spacecraft. The alternation of sun and eclipse is clearly visible in the temperature excursions of the solar panels, which heat to 25C in the sun, then cool to -35C during eclipses. This cycle occurs every orbit, approximately every 100 minutes for TO–31. Inside the spacecraft the batteries and electronics experience a comfortable room-temperature environment with only modest temperature variations.

TO–31 derives its electrical power from solar cells which charge the on-board batteries. Figure 7.7 shows a clear alternation of charge and discharge cycles, charging when the spacecraft is in the sun, and discharging during eclipses. On exit from eclipse the top priority of the power system is to recharge the batteries, accounting for the initial surge over 2 amperes. Once the batteries are recharged the current falls to a 200 mA maintenance level.

The 500mA spikes at 1805 and 1924 UTC correlate with magnetic anomalies at the same time, as illustrated in Figure 7.9. However, the relationship may be entirely coincidental: other spikes in other telemetry files do not correspond with magnetic activity.

The solar cells on TO–31 generate electricity when the spacecraft is in the sun, and cease to do so during eclipses. This is clear from Figure 7.8, which shows that the voltage from the solar arrays rises to over 40 volts in the sun, but drops to the battery voltage (nominally 12 volts) during eclipse.

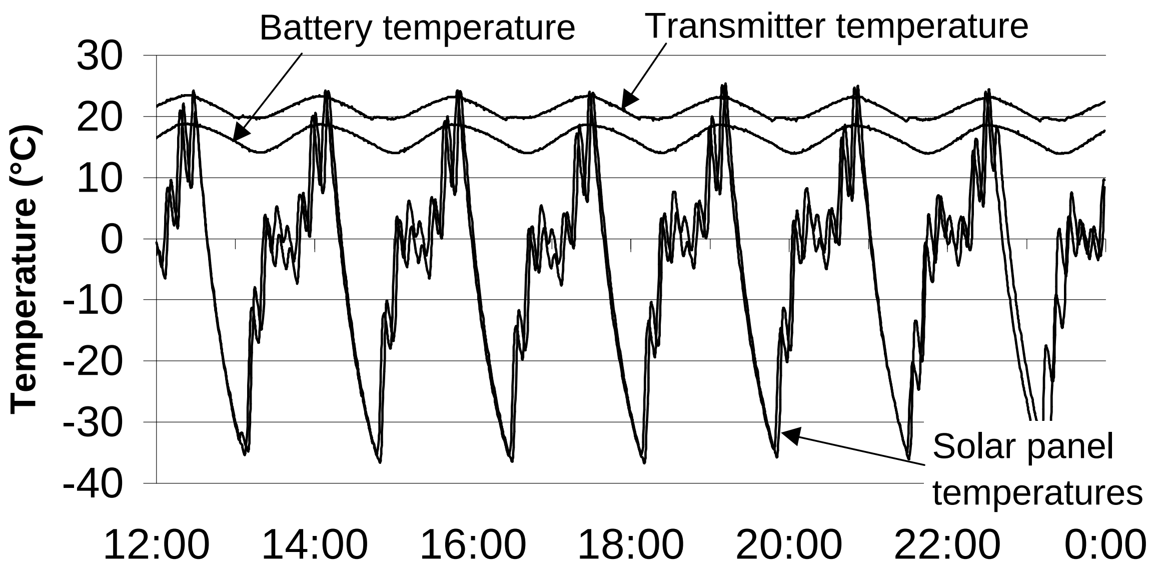


Figure 7.6: TO–31 temperatures: 1200–2359 UTC 28 November 1999

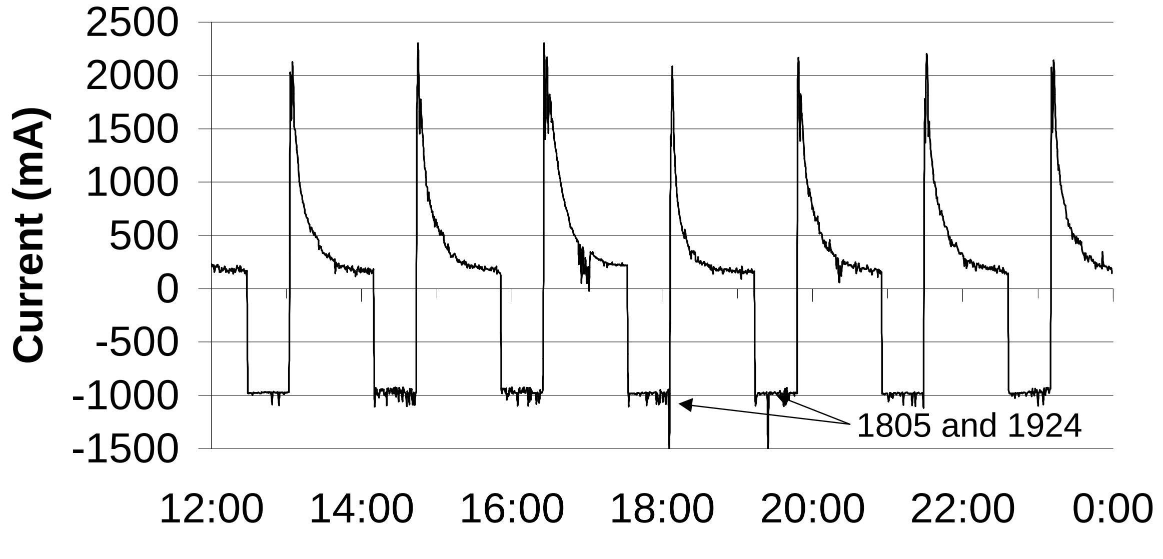


Figure 7.7: TO–31 battery current: 1200–2359 UTC 28 November 1999

The battery voltage shows small variations during this period, reflecting the charge/discharge cycle of the spacecraft.

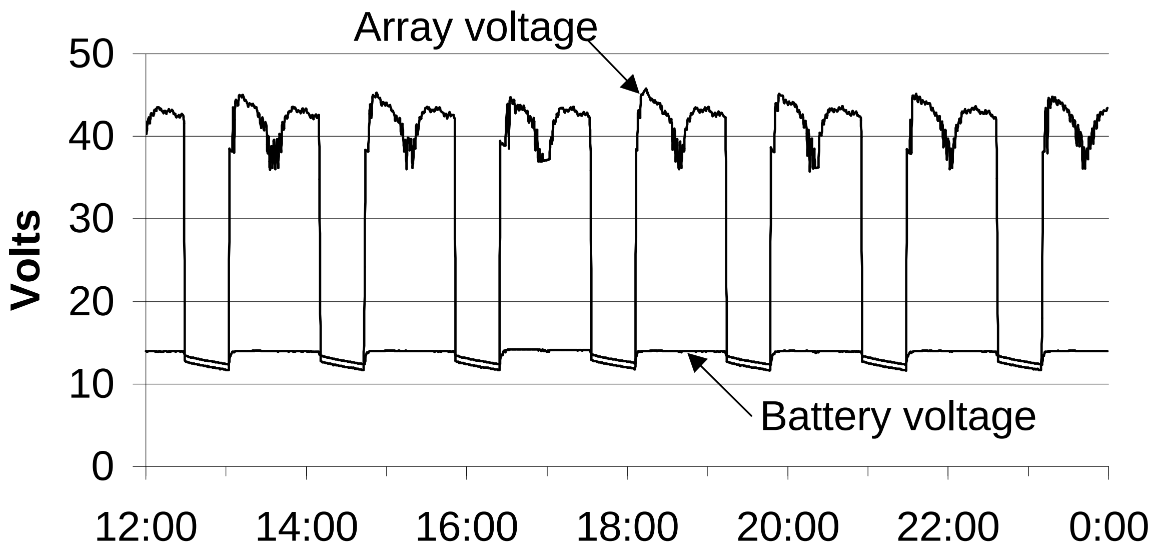


Figure 7.8: TO–31 voltages: 1200–2359 UTC 28 November 1999

Like all the satellites built by the University of Surrey and related programs, TO–31 is stabilized with a gravity-gradient boom. The boom defines the spacecraft Z axis, and is the axis of the spacecraft spin. This is clear from the readings of the spacecraft magnetometers presented in Figure 7.9. The rapid variation on the X and Y magnetometers is due to the spacecraft spinning every 10 minutes. The slow variation on the Z magnetometer is caused by the orbit of the spacecraft altering the spacecraft's orientation to the Earth's magnetic field. There is an additional slower variation caused by the Earth's axial spin altering the relationship of the North and South Magnetic Poles with the spacecraft orbit. The large spikes in the magnetometer readings correlate with the large spikes in the power system seen in Figure 7.7, but may still be coincidental. Even if they are the result of a single phenomenon, it is unknown which is the cause and which is the effect.

7.4.2 Whole orbit data: UO–22

The analysis of UO–22 whole orbit data is facilitated by the unbroken availability of data for an extended period, from June to December 1999. By gathering data for an extended period it was hoped that seasonal changes could be detected in the spacecraft, with the possibility of detection of long-term changes as well. Long-term changes were observed in the spacecraft thermal environment

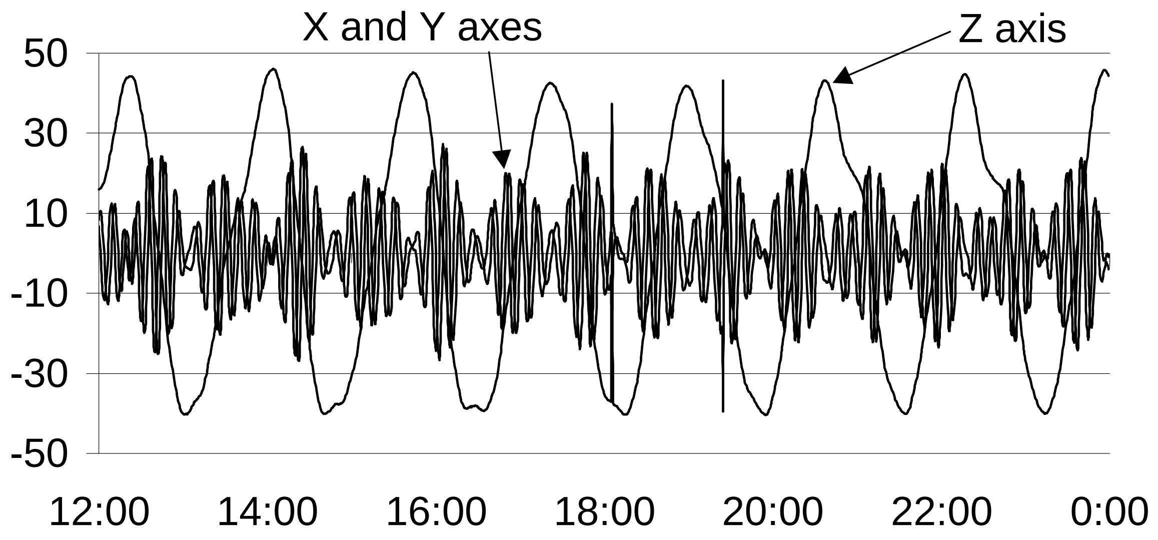


Figure 7.9: TO–31 magnetometers: 1200–2359 UTC 28 November 1999

which were initially believed to be seasonal, but have now been confirmed to be the result of orbital perturbations (3).

The thermal environment of UO–22 is illustrated in Figures 7.10 to 7.13 and shows temperature variations related to the cycle of sun and eclipses. The solar array temperatures are the subject of a detailed analysis conducted once per month; the internal temperatures are recorded continuously.

It is clear that the satellite is running hot, especially in November. Additional data were analyzed to isolate the cause.

The data in Figure 7.11 suggest an increasing trend, but the exact cause of the trend is unknown: the rise is much faster than the temperature change from July to November 1999.

UO–22 derives electrical power from solar arrays, which generate over 40 volts in the sun, but drop to the voltage of the battery, nominally 12 volts, in eclipses. This is clear in Figures 7.14 and 7.15. The eclipse period is clear, and its length may be estimated from the telemetry data: 26 minutes in July, but only 18 minutes at the end of November. The satellite has less time to cool each orbit, and thus runs hotter. The temperature variation may be estimated: to a first approximation, the energy input to the satellite is proportional to the time the spacecraft spends in the sun. From July to November this has increased from 74 minutes per orbit to 82 minutes per orbit, an increase of 10.8%. Absolute temperature varies as the fourth root of energy input (the Stefan-Boltzman Equation) (4), and the fourth root of 1.108 is approximately 1.026, a 2.6% increase in absolute temperature. This

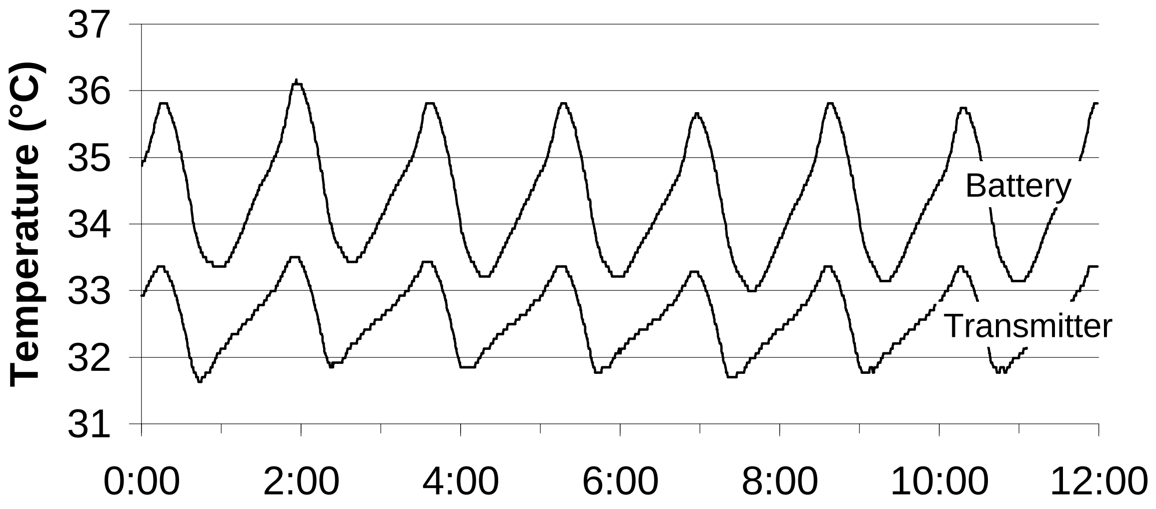


Figure 7.10: UO–22 internal temperatures: 0000–1159 UTC 6 July 1999

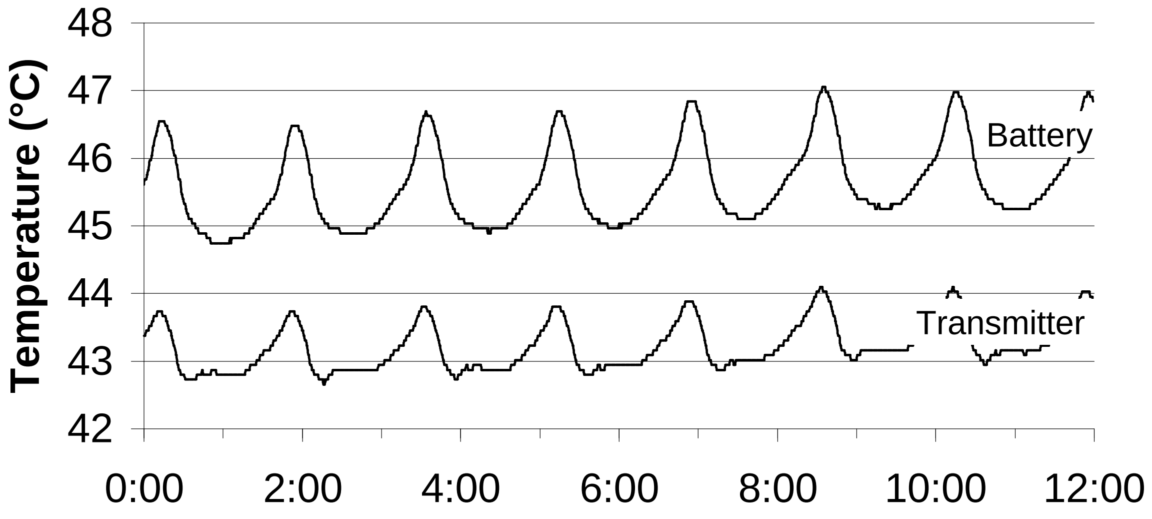


Figure 7.11: UO–22 internal temperatures: 0000–1159 UTC 26 November 1999

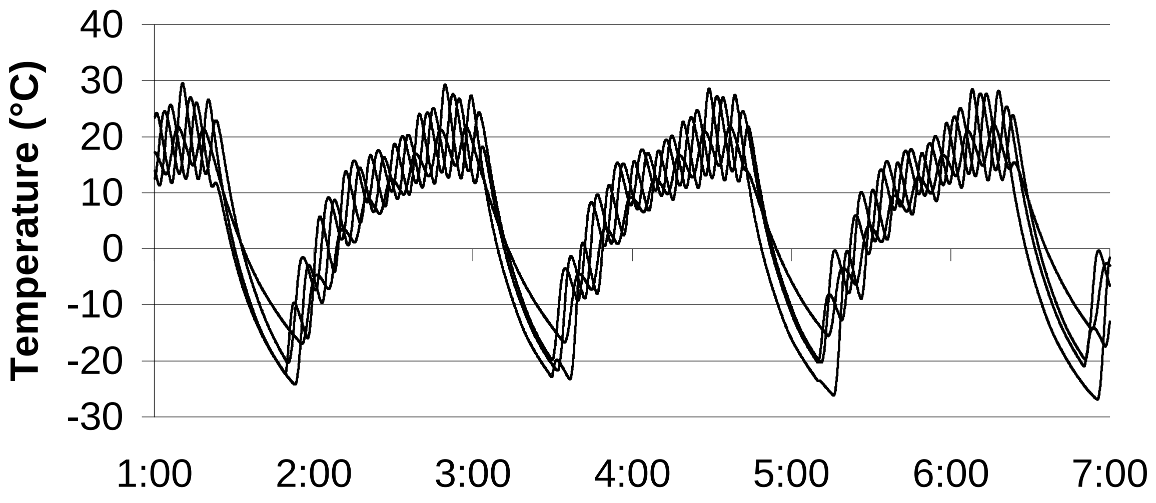


Figure 7.12: UO–22 solar array temperatures: 0100–0659 UTC 15 July 1999

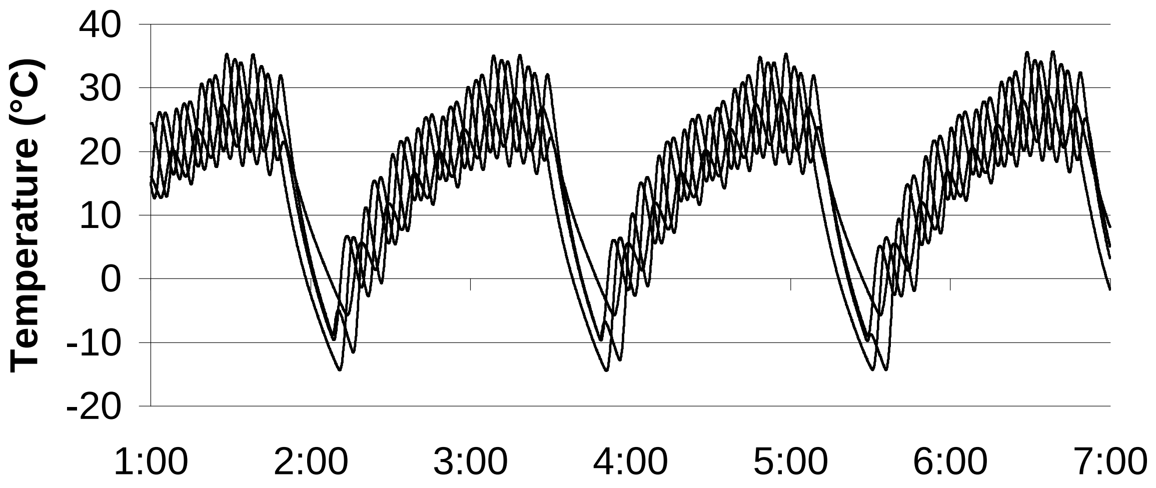


Figure 7.13: UO–22 solar array temperatures: 0100–0659 UTC 15 November 1999

corresponds to a temperature rise of approximately 8 degrees, close to the observed temperature increase. Such seasonal changes are to be expected in satellite operations, and must be allowed for in the thermal design.

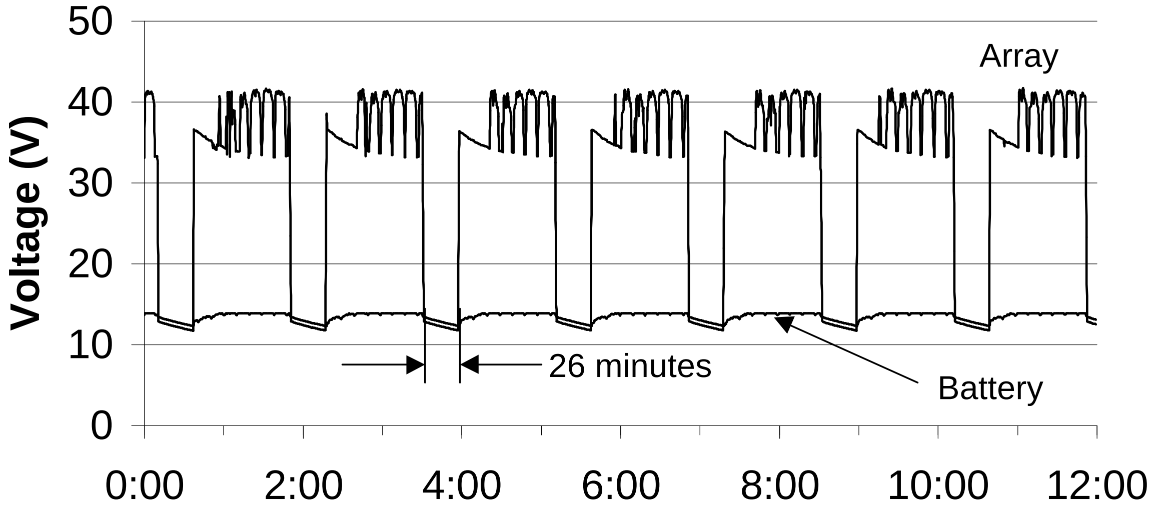


Figure 7.14: UO–22 solar array and battery voltages: 0000–1159 UTC 6 July 1999

December whole orbit data show a further increase in temperature to 47 C, with the eclipse period down to 15 minutes.

Other activities of the satellite may be observed by analysis of whole orbit data. Figures 7.16 and 7.17 show a clear cycle of charging while in the sun and discharging during eclipses. While there are no gross differences between the seasons, it is probable that more detailed analysis of the data would show a drop in charging efficiency with higher battery temperature.

Like TO–31, UO–22 shows spikes in battery current. The magnetometer data in Figures 7.18 and 7.19 do not correlate with the spikes, which are thus not related to magetorquer activity. The data show the 10 minute spin of the spacecraft, the spacecraft orbit around the Earth's magnetic field, and the varying orientation of the spacecraft with respect to the magnetic poles as the Earth rotates. Like TO–31, UO–22 is stabilized by a gravity gradient boom. The boom defines the spacecraft Z axis, which is the axis of the spacecraft spin.

A final example of telemetry analysis arises from the surveys of solar array current. Detailed inspection of the data (Figure 7.20) shows that one of the solar panels, -Y, is different. It consistently produces less current, and shows different thermal characteristics.

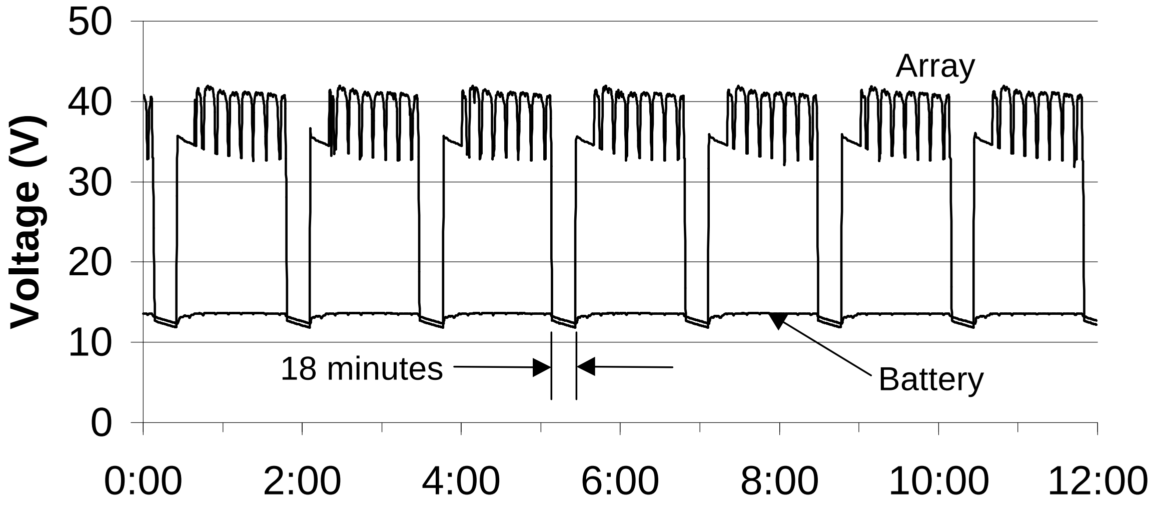


Figure 7.15: UO–22 solar array and battery voltages: 0000–1159 UTC 26 November 1999

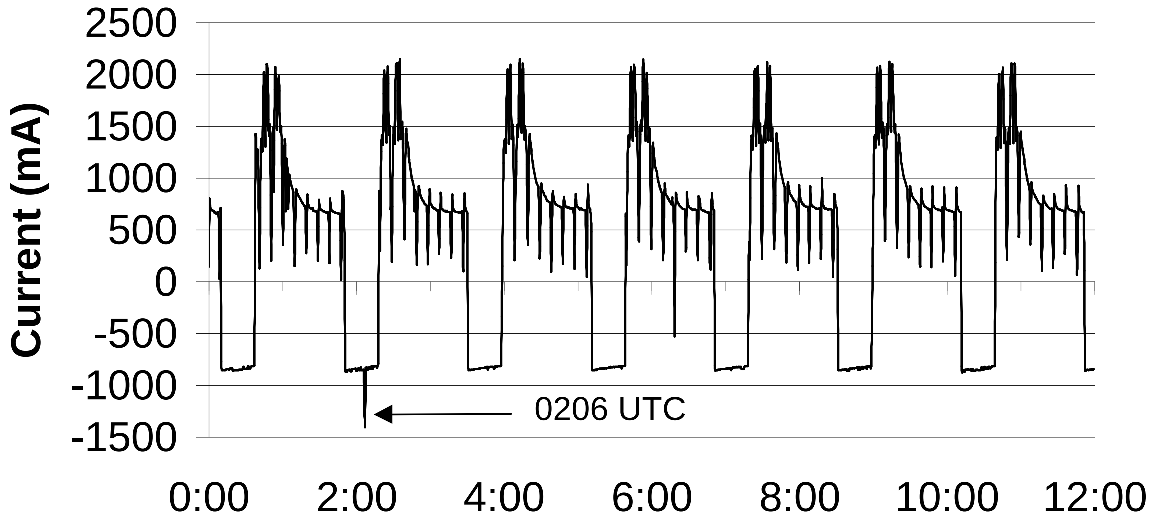


Figure 7.16: UO–22 battery current: 0000–1159 UTC 6 July 1999

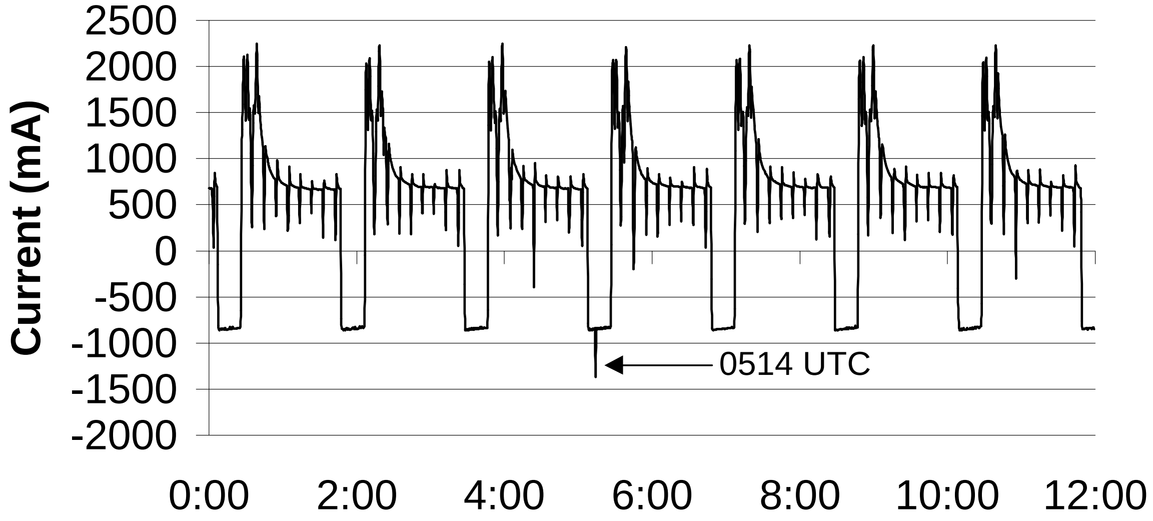


Figure 7.17: UO–22 battery current: 0000–1159 UTC 26 November 1999

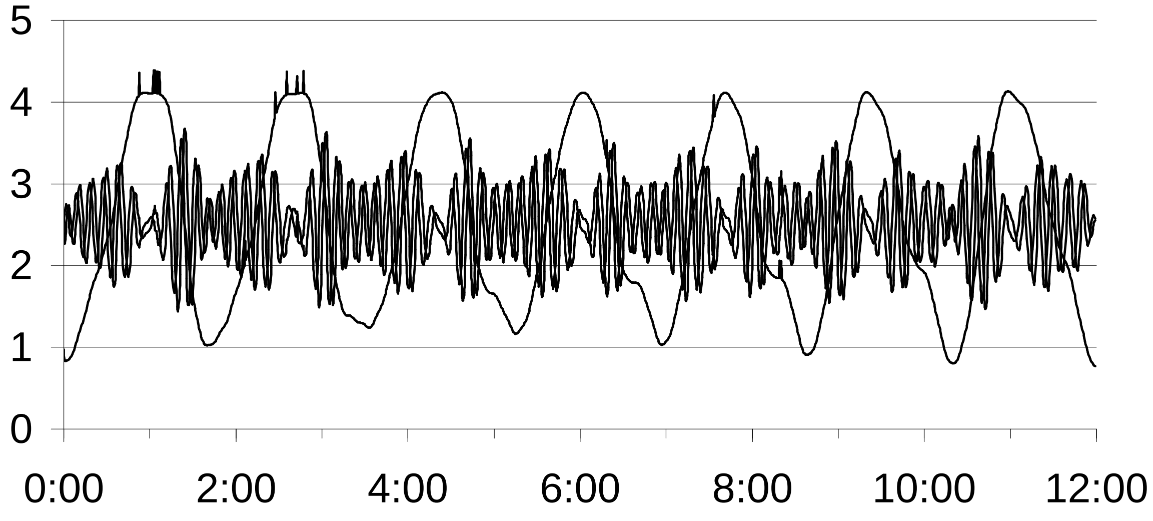


Figure 7.18: UO–22 magnetometers: 0000–1159 UTC 6 July 1999

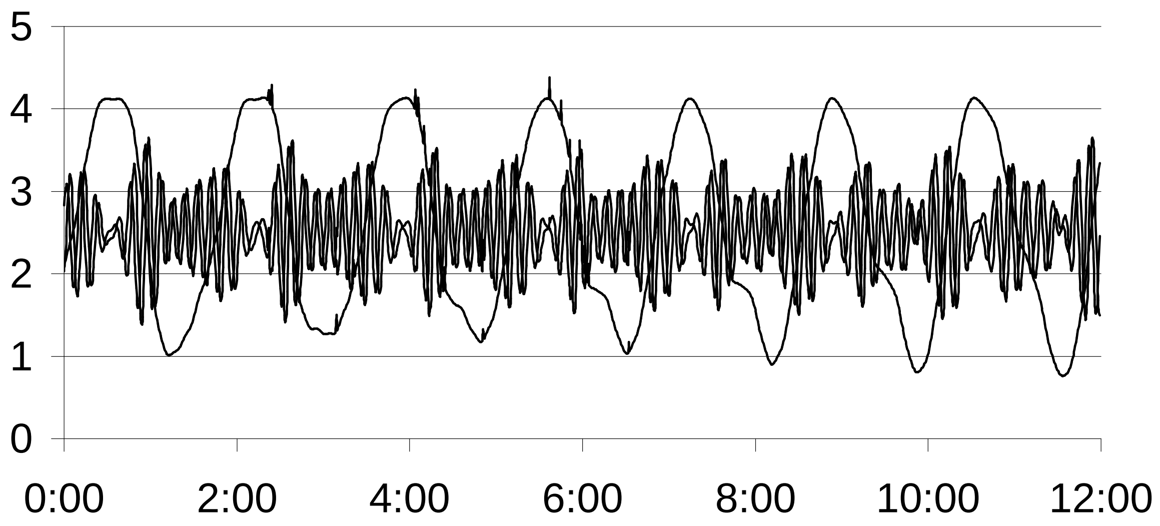


Figure 7.19: UO–22 magnetometers: 0000–1159 UTC 26 November 1999

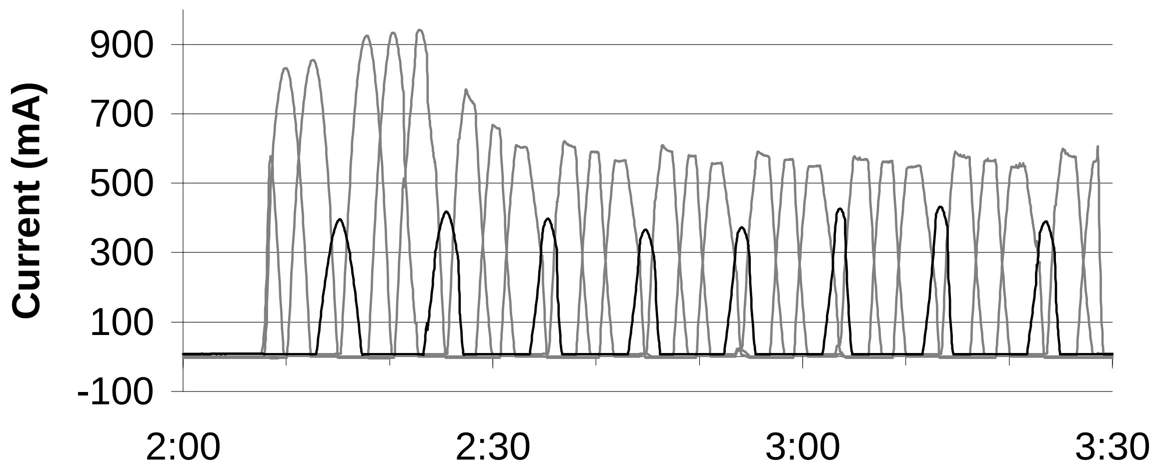
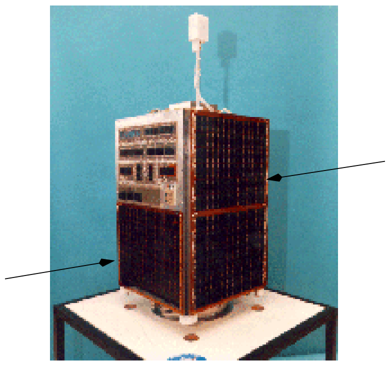


Figure 7.20: UO–22 solar array current: 0200–0330 UTC 15 July 1999

The initial hypothesis was that this panel was damaged, but inspection of a prelaunch photograph (*right*, obtained from SSTL's web site) shows that the -Y array is smaller, with the rest of that face of the satellite devoted to experiments.



Further investigation identified the experiment on UO–22 as an evaluation of solar cell technology, evaluating samples of several different GaAs, InP and Si solar cells (5).

7.4.3 Whole orbit data: AO–16

AO–16 uses an unusual system for gathering whole orbit data, as discussed in Section 4.3.4. A whole-orbit data survey was received and processed at VA3SFL on 12 October 1999. It illustrates the function of the power system and the satellite's simple stabilization system.

The plot of current from the X and Y axis solar arrays, Figure 7.21, shows the spin of the spacecraft as each array sweeps past the sun. Published references describe the use of the 2 meter antenna as a radiometer to spin the spacecraft (6). The data from the X and Y axis solar arrays show a rotation period of aproximately 3 minutes. This correlates with observed ground station performance, where the spacecraft signal strength is subject to deep fades every 2 to 3 minutes.

AO–16 achieves Z axis stabilization with magnets that align the spacecraft with the Earth's magnetic field. While the data in Figure 7.22 are incomplete, they are highly suggestive of the spacecraft rotating twice around the Z axis each orbit: every 50 minutes for an orbital period of 100 minutes.

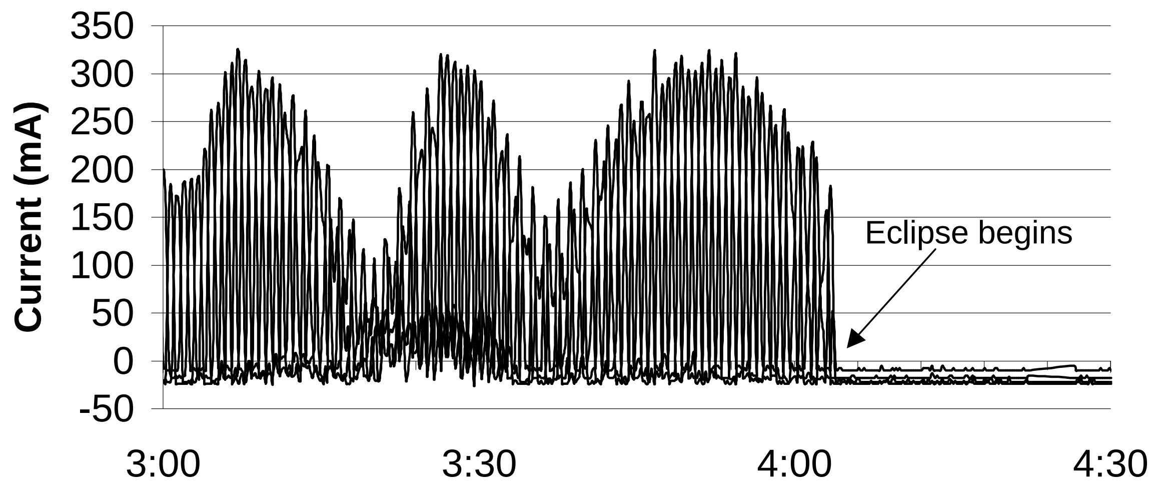


Figure 7.21: AO–16 X and Y axis solar array current: 0300–0430 UTC 12 October 1999

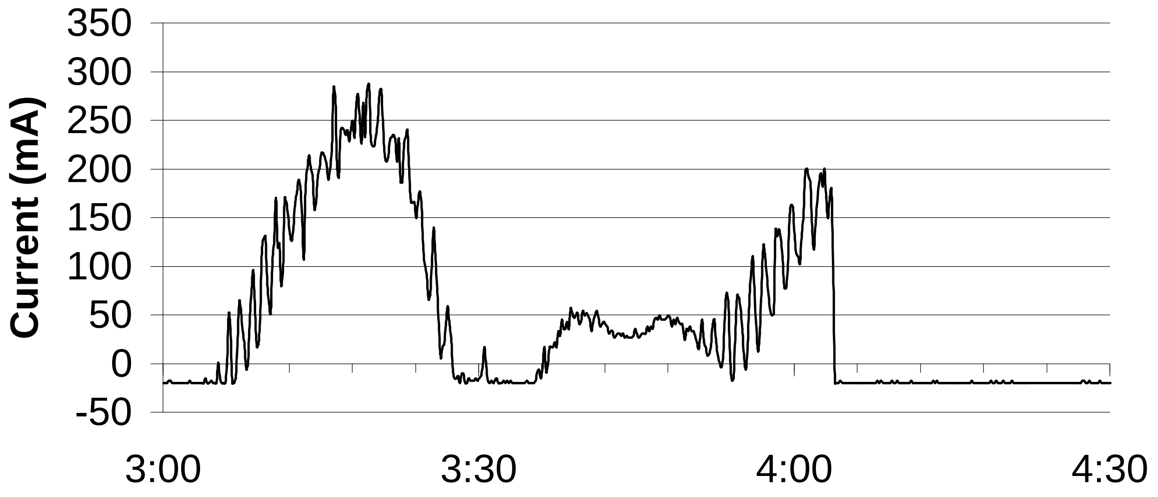


Figure 7.22: AO–16 Z axis solar array current: 0300–0430 UTC 12 October 1999

7.5 Other data analysis

Other data gathered during satellite passes were analyzed. The results from analysis of selected log files and task messages are presented.

7.5.1 Log files

Applications on board satellites create log files of their activities. Sample activity and callsign log files were downloaded from UO–22, decoded with software supplied by SSTL, and their contents analyzed.

The activity log file records FTL0 file activity and server housekeeping. An example sequence shows VK2XGJ, Wollongong NSW, Australia, uploading a file:

```
09:44:30  LOGIN     VK2XGJ-0  0 013033
09:44:31  UPLOAD    VK2XGJ-0        f#6d346 off:0 l#11413
09:45:14  UL DONE   VK2XGJ-0        11413 bytes 0 seconds
                                    cksum Ack'd
                                    Av upload rate 265bps
09:45:15  LOGOUT    VK2XGJ-0    013033 user disconnect
```

Another entry in the activity log shows CX5AE, Montevideo, Uruguay starting to upload a file, stopping (possibly due to LOS, **L**oss **O**f **S**ignal), timing out, and being automatically logged out by the server.

```
00:55:44  LOGIN     CX5AE-0  0 012773
00:55:46  UPLOAD    CX5AE-0        f#6d2a8 off:0 l#1958
00:57:12  BLOWOFF   CX5AE-0        Inactivity timeout
00:57:38  LOGOUT    CX5AE-0    012773 server disconnect
```

The callsign log file records the callsign of every station making a request of the file server. Callsign files for 18 and 19 July 1999 were downloaded and analyzed. As expected in Figure 7.23, industrial countries were heavily over-represented with 20% of the countries (7 out of 34) accounting for 67% of the callsigns (145 out of 217). By number of callsigns the largest single user was the U.S.A.

with 49 callsigns, followed by France with 22 callsigns and Germany with 18 callsigns. During the sample period the satellite was accessed by 10 Canadian callsigns.

Asia is almost entirely represented by Japanese callsigns. The only African callsigns were from South Africa.

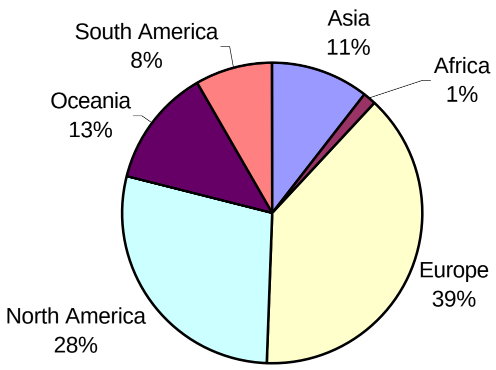


Figure 7.23: UO–22 callsigns by continent

7.5.2 Task messages

In addition to file transfer and telemetry, satellites transmit *task messages*, transmitting information relevant to various subsystems (7).

The first task message is produced by the main housekeeping task. On AO–16 it is called `PHT`:

```
PHT: uptime is 1910/19:36:23. Time is Sun Dec 12 01:19:57 1999
```

Note the remarkable value for uptime: AO–16 ran for over 5 years before a software crash shortly after this message was received.

A similar message received from UO–22 provides similar information. The short up time is due to reloading UO–22's software in mid-November 1999.

```
Fri Dec 17 14:32:10 1999 Up: 27/22:8 EDAC=103 F:146768
   L:146016 d:1 [0].
```

Of particular interest is the `EDAC` number, a count of the number of error events detected by the error-correcting spacecraft memory. The `F` and `L` parameters specify the amount of free memory in SCOS and the size of the largest block of free memory, respectively.

Another task message is from the task scheduler, informing ground stations of the next scheduled operation:

```
Sked 1.5 File1:6f59a Next:Sat Dec 18 00:00:00 1999
```

The OAK attitude control system transmits valuable information:

```
OAK E:1 M:0 C:1 R:64 P:-25 Y:1133 YR:599 OX:195
   OY:-196 OZ:66 TX:0 TY:0 TZ:0
```

Of particular interest are the `R`, `P` and `Y` values, roll, pitch and yaw, respectively. The satellite spins, so the yaw value is always changing. The `YR` yaw rate parameter specifies the rate of this change. The angular parameters are in .01 degree increments, showing sub-1 degree pointing accuracy from the attitude control system which is based on a gravity-gradient boom and magnetorquers.

Not all task messages are as well documented. The following task message was received from KO–25 and is probably from the **E**arth **I**maging **S**ystem, but the significance of the individual values is unknown:

```
EIS:<385a70d9> [s4] Timer:0 TxEnable:1
```

7.6 Performance analysis

The downlink performance of UO–22 and TO–31 was analyzed. Particular emphasis was placed on protocol overhead and effective throughput, and the results are presented here.

7.6.1 Environmental issues

The environment in which the ground station operates was analyzed.

The present ground station antennas are on the roof of the UTIAS building in Downsview, Ontario. The area is flat, part of southern Ontario on the north shore of Lake Ontario. There are thus no geographic obstacles. While there are deciduous trees, they subtend an angle of less than 5 degrees as seen from the ground station antenna site, and had no discernable effect on ground station performance. In both summer and winter 435 MHz signals from satellites were acquired within seconds of theoretical AOS (**A**cqusition **O**f **S**ignal), and lost within seconds of theoretical LOS. No seasonal effects were noted between July and December.

The only detectable obstruction was the satellite television antenna on the roof of the UTIAS building. This attenuated signals for a satellite azimuth from approximately 200 to 240 degrees, at elevation angles below 15 degrees. This is a small part of the sky, and the loss of data is minimal. Such an environment would not affect MOST, since passes in the southwest have mutual visibility with the proposed ground station in Vancouver.

7.6.2 Data throughput

Data were gathered during a representative week, 22 to 26 July 1999, to evaluate access time, net data throughput, and the amount of data transferred. The ground station operations were nominal, as were the satellite operations.

The first measurement was access time. As shown in Figure 7.24, the access time is uniform over the study period. Minor variations are mainly due to orbital geometry and average out over longer periods.

The amount of data transferred was then computed. Protocol information was analyzed with an instrumented version of `decode` (Appendix C) that classified packets by their function and tallied the number of bytes devoted to file transfer. This compensated for the multiplexed nature of the

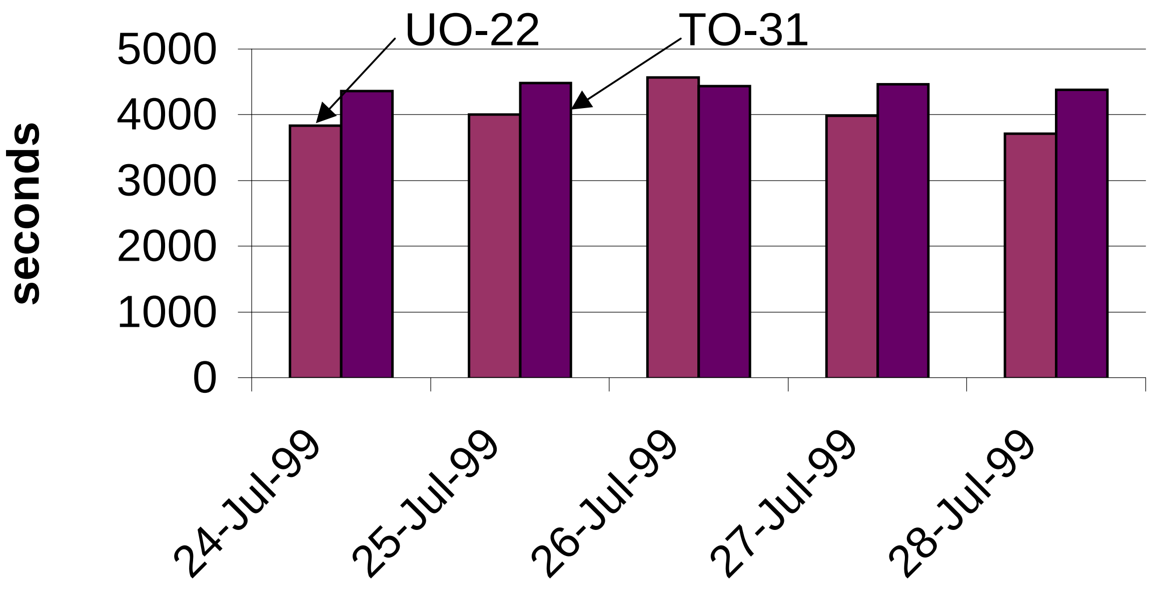


Figure 7.24: UO–22 and TO–31 access time, 22–26 July 1999

downlink—rather than counting the bytes received at VA3SFL, the count was of all bytes transmitted by the satellite, more closely approximating the environment anticipated for MOST.

The final analysis combines the results of Figures 7.24 and 7.25 to determine the net number of bytes per second transferred from the satellite, as presented in Figure 7.26. TO–31 has consistently shown higher throughput, believed to be partially the result of a higher duty cycle on its downlink, and partially the result of lower protocol overhead (Section 7.6.3). In November 1999 this duty cycle was increased, with many passes showing throughput exceeding 1000 bytes per second. Reliability concerns with the ground station (Section 7.3.4) have precluded further analysis of this development.

7.6.3 Protocol overhead

Protocol overhead was analyzed with an instrumented version of `decode` (Appendix C) that classified packets by their function and tallied the number of bytes. All data received on the downlink for the week of 20 July 1999 were used for the analysis.

Differences were observed, as reported in Figures 7.27 and 7.28. UO–22 transfers large numbers of small files, and thus must devote more of its downlink to directory information. TO–31 transfers small numbers of large files, and can thus devote more downlink capacity to actually transferring files.

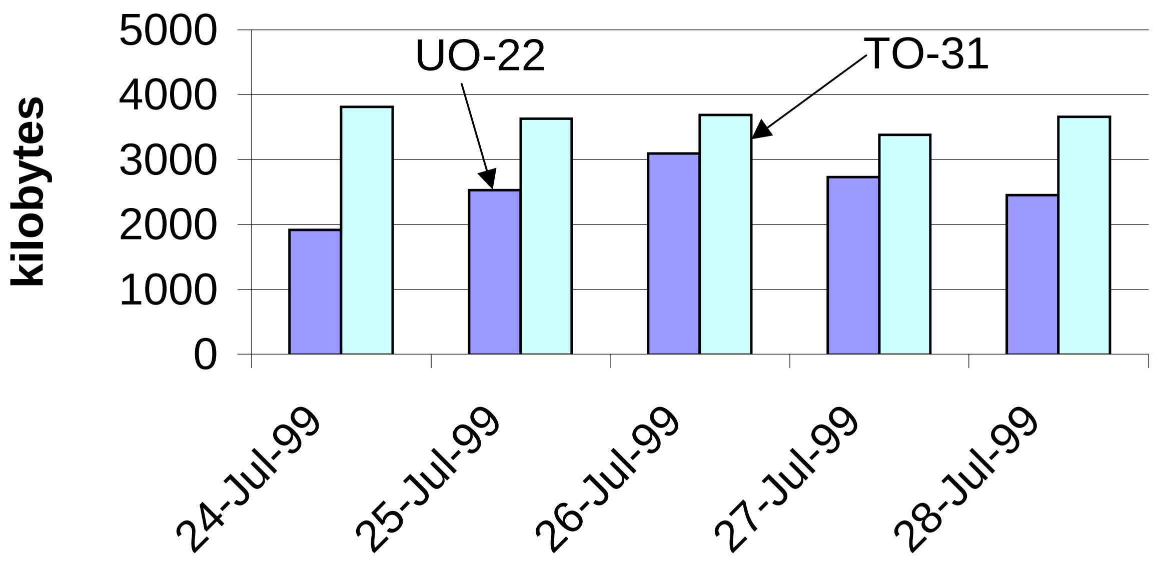


Figure 7.25: UO–22 and TO–31 file transfer, 22–26 July 1999

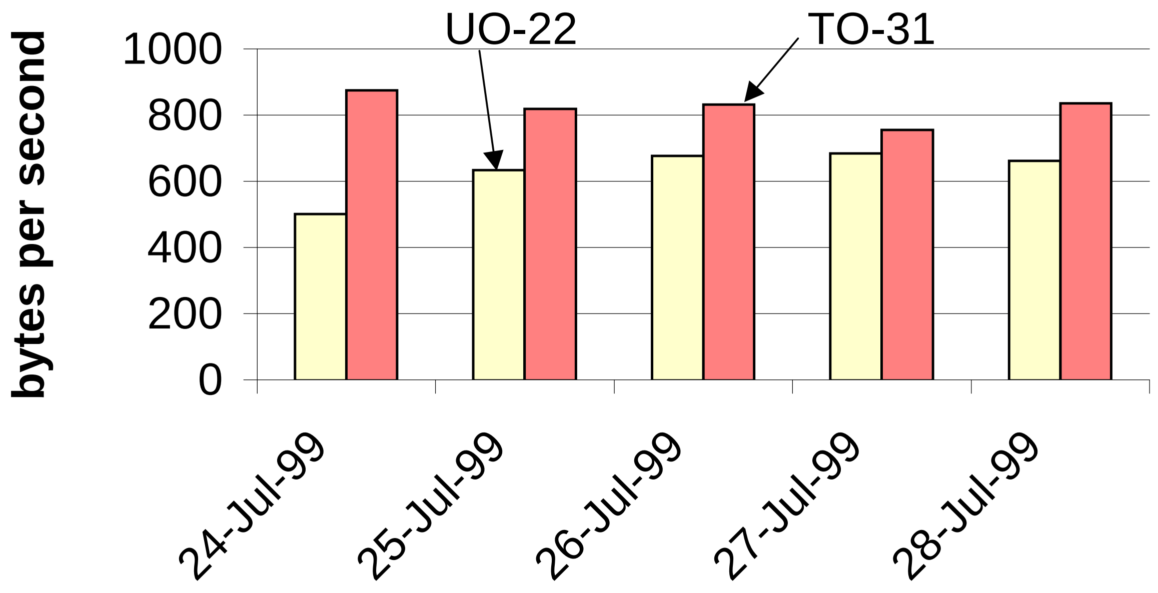


Figure 7.26: UO–22 and TO–31 throughput, 22–26 July 1999

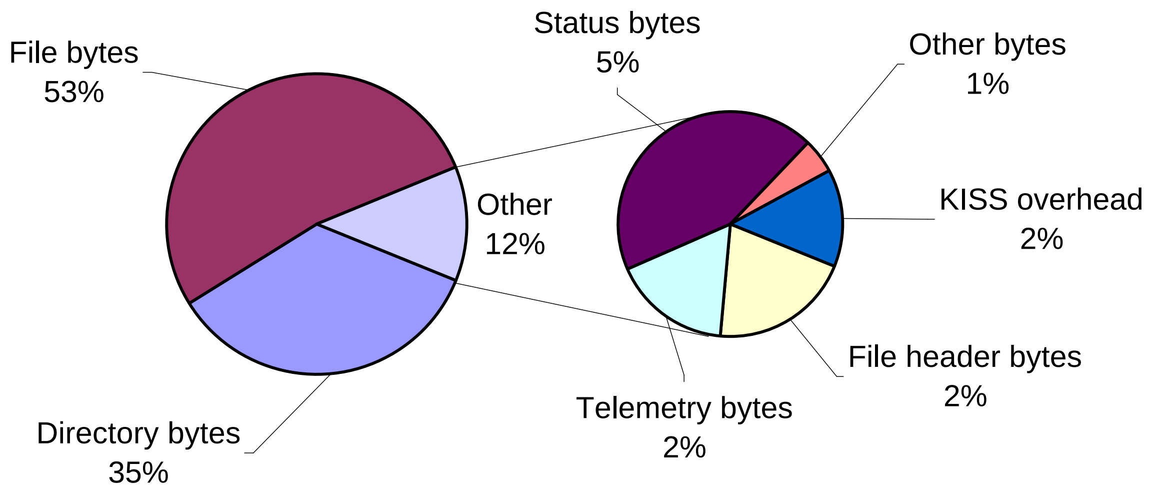


Figure 7.27: UO–22 protocol overhead analysis: July 1999

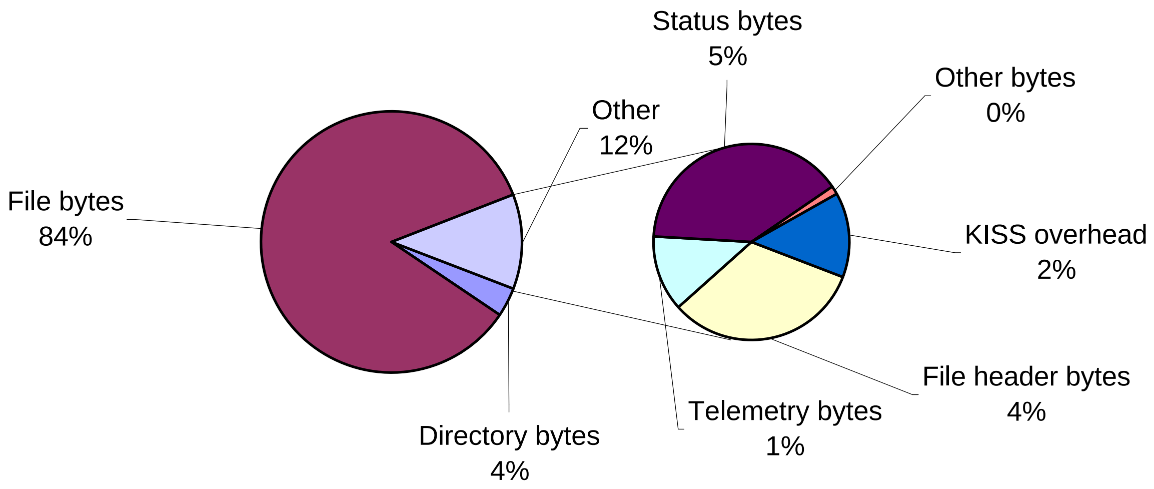


Figure 7.28: TO–31 protocol overhead analysis: July 1999

7.9 Notes and references

1. Geological Survey of Canada. *Canadian Geomagnetic Reference Field, 1998.*
   Internet: `http://www.geolab.nrcan.gc.ca/geomag/e_crgf.html`

2. J. White and R. Haighton, AMSAT-NA, personal communication.

3. C. Jackson, SSTL, personal communication.

4. W. Larson and J. Wertz, editors, *Space Mission Analysis and Design, 2nd edition.* Torrance: Microcosm and Dordrecht: Kluwer Academic Publishers, 1992.

5. M. Sweeting, “UoSAT microsatellite missions” in *Electronics & Communication Engineering Journal*, June 1992. Figure 11 is a closeup view of this experiment.

6. M. Davidoff, *The Radio Amateur’s Satellite Handbook.* Newington: ARRL, 1998.

7. C. Jackson, “UoSAT Spacecraft Operation”, paper presented at the 11th AMSAT–UK Colloquium, 1996.
   Internet: `http://www.ee.surrey.ac.uk/CSER/UOSAT/amateur/taskinfo.html`

8. A. Ward, “Low noise Amplifiers for 2304, 3456, 5760 and 10368 MHz using the ATF–36077 PHEMT” in *Proceedings of Microwave Update ’97.* Newington: ARRL, 1997.

9. G. Hoch, “Yagi Versus Aperture Antennas” in *The ARRL UHF/Microwave Experiemnter’s Manual.* Newington: ARRL, 1990.

10. W. Vogel and J. Goldhirsh, “Multipath fading at L band for low elevation angle, land mobile satellite scenarios” in *IEEE Journal on Selected Areas in Communications*, vol. SAC–13:2, February 1995.

11. J. Goldhirsh and W. Vogel, “Mobile satellite system fade statistics for shadowing and multipath from roadside trees at UHF and L band” in *IEEE Transactions on Antennas and Propagation*, vol. AP–37:4, April 1989.

12. H. Smith et al, “Characterisation of the high elevation orbit satellite-mobile radio channel at L and S bands” in *IEE Colloquium on Land Mobile Satellite Systems*, 1992.