UNIVERSITY OF OSLO
DEPARTMENT OF PHYSICS

WASAT

A graphical user interface for visualization of wave spectrograms

Runar Jørgensen

Department of Physics, University of Oslo
P.O.Box 1048 Blindern
N-0316 Oslo, Norway

OUP
UIO/PHYS/96-13 **Received: 1996-12-23**

REPORT SERIES

WASAT

A graphical user interface for visualization of wave spectrograms

Runar Jørgensen

Department of Physics, University of Oslo
P.O.Box 1048 Blindern
N-0316 Oslo, Norway

OUP
UIO/PHYS/96-13 **Received: 1996-12-23**

WASAT

A graphical user interface for visualization of wave spectrograms

Runar Jørgensen
Department of Physics, University of Oslo
P.O. Box 1048 Blindern
N-0316 Oslo, Norway

8th January 1997

Abstract

This report describes a technique for the decoding and visualization of sounding rocket data sets. A specific application for the visualization of three dimensional wave HF FFT spectra obtained from the SCIFER sounding rocket launched January 25, 1995, is made. The data set was decoded from its original data format which was the NASA DITES I/II format. A graphical user interface, WASAT (WAve Spectrogram Analysis Tool), using the Interactive Data Language (IDL) was created. The data set was visualized using IDL image tools overlayed with contour routines. The user interface was based on the IDL widget concept.

Contents

1 Introduction

The Research Section for Plasma and Space Physics, Department of Physics, University of Oslo, is regularly involved in national and international sounding rocket projects. In NASA payloads as SCIFER, the data flow adhere to a data format, digital telemetry system (PDP11/60) DITES I/II, developed by Wallops Flight Facility (WFF, 1988) (see section 2.1).

1.1 The SCIFER Campaign

The sounding rocket for Studies of the Cleft Ion Fountain Energization Region (SCIFER), was successfully launched 06:24 UT[1] on January 25th, 1995 from Andøya Rocket Range (69°17' N, 16°01' E) into the pre-noon polar cleft and cap. The payload reached an altitude of 1452 km. The duration of the flight was ≈ 20 minutes, i.e., 1207 seconds (UT: 06:24 – 06:45; local time: 07:23 – 07:45). Radio contact with the payload was lost at 06:45:56 UT.

The main scientific objective of the SCIFER campaign was to study the energization and outflow of ionospheric plasma in the dayside cusp/cleft region, which is believed to be the most important ionospheric ion source for the magnetosphere. The aim was:

- to document the electron and ion plasmas of the cusp/cleft ionosphere, including surrounding polar wind outflows,
- to determine the adequacy of frictional heating to deliver the observed up welling plasma flux,
- to determine the role of electron heating and pressure distribution in plasma energization and outflow,
- to determine the role of Birkeland currents, including collective effects and instabilities, in plasma heating and energetics,
- to determine the role of plasma waves in energy transfer, inheating and plasma outflow.

Scientists from the following institutions participated in the project in addition to the group at University of Oslo: Cornell University (Principal Investigator), University of New Hampshire, University of Alaska and NASA Marshall Space Flight Center, USA. The payload (see figure 1) provided a unique data set from the dayside polar cleft and cap regions.

2 NASA TDDS DITES II Digital Telemetry System

Data from a sounding rocket are digitized, formatted into blocks of frames and transmitted via a telemetry system using Bi-phase Pulse Code Modulation/Frequency Modulation (PCM/FM). This data includes both scientific and housekeeping data.

The telemetry data from the SCIFER payload was disseminated through NASA's Telemetry Data Distribution System (TDDS) and stored on one CD-ROM containing 218 MByte

[1] Local time: 07:24

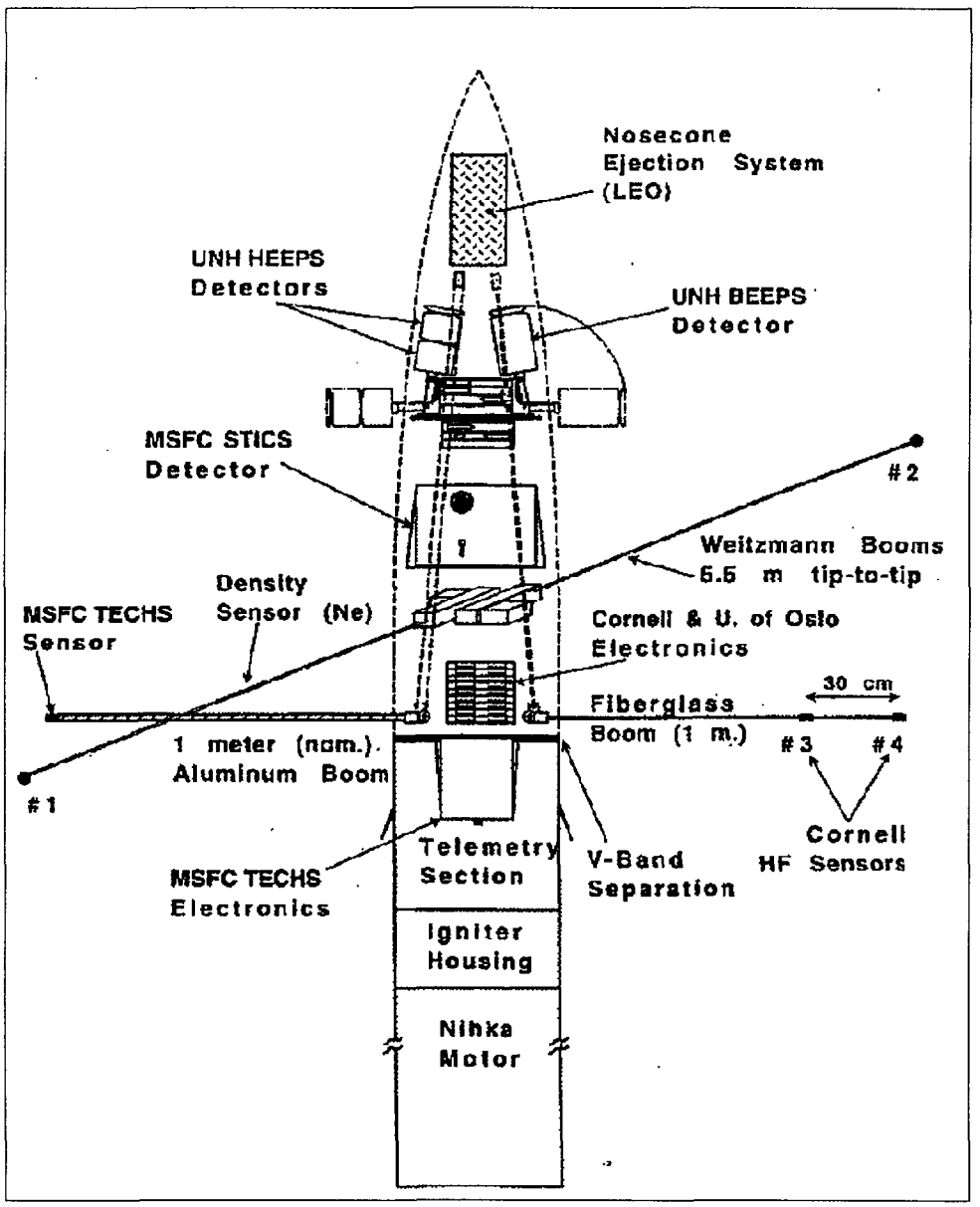

Figure 1: The SCIFER payload was designed to study the energization and outflow of ionospheric plasma in the dayside cusp/cleft region.

Byte:	msb							lsb
	D7	D6	D5	D4	D3	D2	D1	D0
Word:	MSB							LSB
	byte1							byte 0

msb: Most significant bit
lsb: Least significant bit
MSB: Most significant byte (byte 1)
LSB: Least significant byte (byte 0)

Table 1: The table defines some byte/word orientation and nomenclature used in TDDS CD-ROM distribution. Numerical integer representations are defined for Word1:

Word 1 = Byte 1 Byte 0 = MSB LSB = FF FE Hex

That is, the MSB precedes the LSB (last in hexadecimal representation). D15 represents the most significant bit position; D0 represents the least significant bit position.

of reprocessed data. NASA did the reprocessing, i.e. filled gaps in the data sets and merged time words.

The University of Oslo (UiO) experiment was a HF receiver. The frequency range for the instrument was 10 kHz – 5.0 MHz. UiO instrument sampling rate was 10 M samples per second. The data were transmitted over Telemetry Channel #1 (TM1). Telemetry transfer rate was 3 200 Kb/s with PCM encoding and 10 bit resolution. There was transmitted 25 complete UiO FFT processed spectras per second. Each spectrum with 128 samples.

The CD-ROM uses the DITES II format (WFF, 1988).

2.1 Data File Format (CD-ROM DITES II)

The DITES II format includes 10 microsecond time-tagging resolution and a data/time status byte. Word 11, the number of merged time words, will therefore always be 4 (see table 3).

The format uses a 10-bit word as the smallest entity. When data is transformed to commercial computers, bytes and 2-byte words are used (see section 2.1.4). In a word, the MSB precedes the LSB, a format which is compatible with all Sun Workstations (see table 1).

The data processing at UiO was performed on a SUN SPARC station 20, HyperSPARC (2 × 125 MHz HyperSPARC microprocessors) running Sun Microsystems' SunOS 5.5.1 (Solaris 2.5.1).

2.1.1 Headers

The data set consists of three different types of records (see figure 2).

- The first record is a *header record* that contains up to 80 ASCII characters.
- The second type of record is a *data descriptor record* which describes the data in each file set.

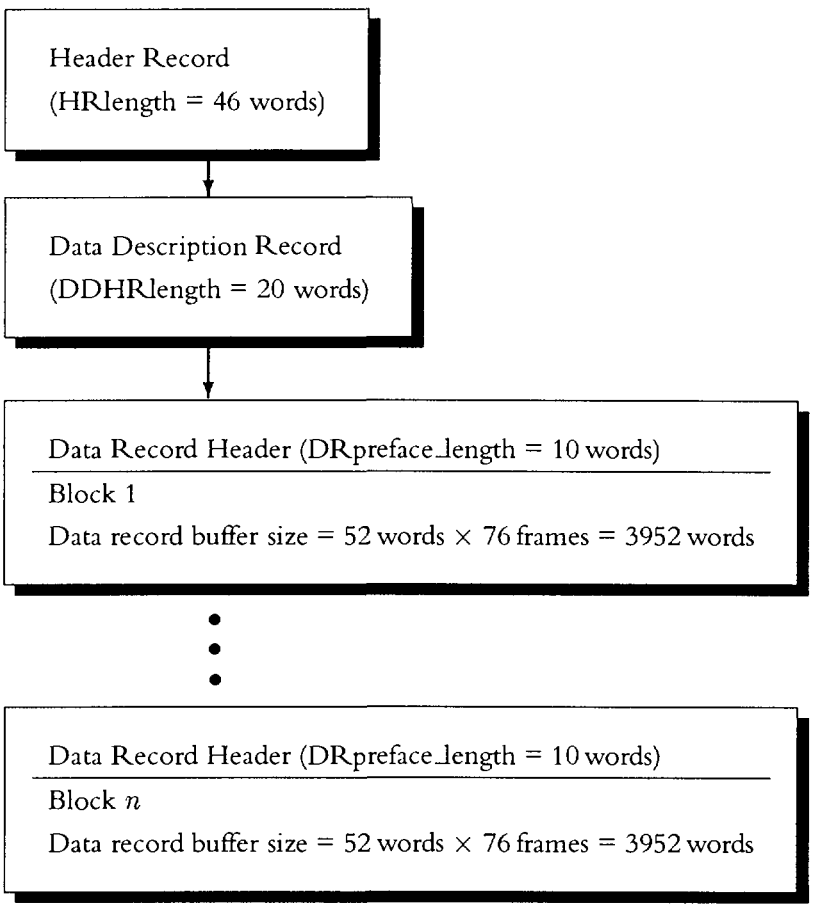

Figure 2: The figure gives a graphical representation of the structure of the DITES II format. The file has a total of 66 words (132 bytes) of header (Header Record and Data Description Record). The rest of the file is organized as blocks of data. Each block is 3962 words including 10 preface words (Data Record Header) for each block. There are > 27 000 blocks in the file `d4000611.dat`. Each block is organized as 52 words in 76 frames.

- The third type of record is the *data record.*

All record headers share a common record marker which is the number FF FE Hex[2]. Each record is preceded with this record marker.

The IDL program `headers.pro` is developed to read and display information contained in the headers (see appendix A.2).

Header Record This record is located only once and at the beginning of the data set on the CD-ROM distribution. The format is visualized in figure 2. Table 2 gives a breakdown of the header record contents.

From this record the value of W2, the record header length (`HRlength`) is loaded to define the data structure used by the IDL application.

Data Description Record This record is located at the beginning of each file set. That is, for the CD-ROM distribution of the SCIFER telemetry data, the Data Description Record (DDR) occur only once since the telemetry data is all contained in one single file named `d4000611.dat` (218.590 Mbytes) (see figure 2).

The values of three words are copied from this header for subsequent processing (see table 3). These are W2, the data description header length (`DDHRlength`); W9 (`DDR-frames`) which is the number of frames per buffer; and W12 which is the total number of words per frame including the merged time words.

In fact, each data buffer block could be defined using information in words W13 and W16 obtained from the data description record. All the same, this information is obtained from the Data Record header.

Data Record Every data block has this format. The data record is made up of a standard preface, an added preface, the data and an appendix (= 0). All measurement data is recorded with the time words merged at the end of each frame.

W2, data buffer size (`DBsize`), is the number of words for each block of data including 10 preface words; W3 (`block_nr`) is the block number, and W6 is decoded and the value is loaded into the `preface_words` variable.

2.1.2 The frame format

The frame is the smallest whole entity in the data format. One frame consists of 48 words and 4 merged time words which makes a total of 52 words (see table 3). All experiment data and housekeeping data are organized into a frame by the onboard PCM encoder. Only a few words in addition to the UiO experiment are described here.

The first word in each frame is a frame counter with range 0 – 7C Hex (see figure 3). Word 2 is subcommutated alternating measurements by the two probe potentials and the rocket body potential. Word 3 through Word 8 are aspect magnetometer data. Words 11 and 36 are the UiO FFT experiment.

[2]Represented in hexadecimal notation.

Header Record contents		
Record Marker	(W1)	FF FE Hex
Header Record Length (words)	(W2)	46
Header Record ID	(W3)	0
Stream ID (4 characters)	(W4 & W5)	TDDS
CD/Tape number	(W6)	1
ASCII string test	(W7-W46)	`40006 Kintner TM#1 50 by 32 format 10 bit wds VLDS Direct`

Table 2: Breakdown of the header record

Data Description Record contents		
Record Marker	(W1)	FF FE Hex
DDR length (words)	(W2)	20
DDR ID	(W3)	FF FD Hex
Run ID (4 characters)	(W4 & W5)	TDDS
Stream ID (4 characters)	(W6 & W7)	TDDS
Words per frame	(W8)	48
Frames per buffer	(W9)	76
Scaling factor	(W10)	0
Merged words (time words in record)	(W11)	4
Total words in frames	(W12)	52
Preface words above 8	(W13)	2
Buffer appendix size	(W14)	0
Input buffer size [words]	(W15)	3952
Output buffer size [words]	(W16)	3962
Frames per subframe 1	(W17)	32
Frames per subframe 2	(W18)	0
Record Mode (0 = word; 1 = byte)	(W19)	0
Two ASCII blanks	(W20)	32 32

Table 3: Breakdown of the data description record

Data Record contents		
Record Marker	(W1)	: FF FE Hex
Data Buffer size [words]	(W2)	: 3962
Block number	(W3)	: 1
Stream ID	(W4 & W5)	: TDDS
(MSB) Appendix words; Preface words	(W6)	: 0; 10
Input buffer size	(W7)	: 3952
TFE Status (bitflags)	(MSB, W8)	: 0
Frame number in block	(LSB, W8)	: 0
Magnetic Tape Write Status	(W9)	: 1
Not used (*nil*)	(W10)	: 0

Table 4: Breakdown of the data record

Frame	W1	W2	···	W11	···	W36	···	W49	W50	W51	W52
1	0 74	2 9B	···	0 E	···	0 18	···	76 0	50 83	50 62	9 2
2	0 78	3 D7	···	0 18	···	0 A	···	77 0	50 83	50 62	9 2
⋮	⋮	⋮		⋮		⋮		⋮	⋮	⋮	⋮
76	0 20	3 D7	···	1 DC	···	1 41	···	23 0	50 84	50 62	9 2

Figure 3: Depicted in the figure is the content of the first frames and the position of the frame counter FCount (W1), UiOFFT1 (W11), UiOFFT2 (W36), and the timing words (W49 through W52). All entries are presented in hexadecimal notation.

UiOFFT1	UiOFFT2	UiOFFT3	UiOFFT4	UiOFFT5 ⇒ UiOFFT128
0	03 FF	0	RNG	Data

Exponent				Mantissa					
9	8	7	6	5	4	3	2	1	0
c	c	c	q	q	q	q	q	q	q

Figure 4: Upper illustration shows the organization of one complete FFT spectrum, while the lower illustration shows the compressed code organization with the exponent, c, taking the upper 3 bits and the quantization, the mantissa, taking the 7 lower bits.

2.1.3 Binary Coded Time Format

Time is merged at the end of each frame (W49 – W52). The DITES II format is designed for 10 microsecond time-tagging resolution. At the time of the SCIFER launch, NASA had not implemented this timing accuracy. Thus, the 10 microsecond resolution is replaced with zeros in the excess timing words, i.e. LSB in W49 is always nil (see table 5). Therefor time acurracy is 1 ms.

Time of launch (TOL) occurred at 06:24:48:000:00 UT on January 25, 1995. A down telemetry site was established at Tromsø Satelitte Station (69°59' N, 19°23' E). The station did not have line-of-sight to the rocket until ≈ 19 seconds after launch, thus first time code in the data set is 025:06:25:08:376:00. The launch occurred on the whole second which is usual for NASA launches.

The time is BCD formatted and organized as described in table 5.

2.1.4 FFT Spectrum Data Format

A DSP[3] unit was used to carry out on-board data analyses in real time (Brondz, 1989). A spectral power density estimate is produced by the DSP-analyzer using 10-bit words. One spectrum consists of 128 entries including a 4-word header (LW1 – LW4) with ID sequence, 00 03 FF 00 Hex, and Range word ahead of the data (see figure 4), thus the total number of samples per spectrum is 124 words.

In order to achieve maximum efficiency in the available telemetry each sample is normalized and compressed (Brondz, 1989). Data compression uses a modified A-law compression algorithm, where 12-bit words are compressed into 10-bit. The floating point representation of the measurement values is illustrated in figure 4. The exponent c of the scale factor is placed in the msb 3 bits, while the fractional part, the mantissa, is occupying the lower (lsb) 7 bits.

The transfer of 10-bit words to commercial standard words (16 bit = 2 byte) is performed prior to the dissemination of the data (on CD-ROM). Thus, bytes and words can be read using standard techniques as described by IEEE standards and supported by SUN (see section 2.1).

[3]Digital signal processor

W49				W50				W51				W52			
B2		B1		B4		B3		B6		B5		B8		B7	
MSN	LSN	MSN	LSN	MSN	LSN	MSN	LSN	MSN	LSN	MSN	LSN	MSN	LSN	MSN	LSN
100 μs	10 μs	10 ms	1 ms	1 sec	100 ms	1 min	10 sec	1 hour	10 min	1 day	10 hours	100 days	10 days	status	status

Byte	MSN				Units	LSN				Units
1	x	x	x	x	10 ms	x	x	x	x	1 ms
2	0	0	0	0	100 μs	0	0	0	0	10 μs
3	x	x	x	x	1 min	0	x	x	x	10 s
4	x	x	x	x	1 sec	x	x	x	x	100 ms
5	x	x	x	x	1 day	0	x	x	x	10 hours
6	x	x	x	x	1 hour	0	x	x	x	10 min
7	s	0	s	s	status	s	s	s	s	status
8	x	x	x	x	100 days	x	x	x	x	10 days

Table 5: TDDS – Microsecond Accuracy (10^{-6}). There are four time words (8 bytes) merged at the end of each frame. The actual time is BCD formatted, i.e., represented as nibbles (4 bits), with 2 nibbles in each byte. (MSN = Most Significant Nibble; LSN = Least Significant Nibble) (Bretthauser, 1994)

The fifth word (LW5) in the FFT spectrum data format is the measuring range, which can take values 0 to 15. The value of the range is valid through each spectrum and gives a dynamic range of > 60 dB. The actual values of the measurement is obtained using the following algorithm.

$$\log_2 10 \cdot \log_{10} \cdot \texttt{mantissa} + \texttt{exponent} + (15 - \mathrm{RNG}) \quad (1)$$

The IDL implementation for extracting the exponent and mantissa, and calculating the value is presented in section 3.1 .

3 Interactive Data Analysis using IDL

The group have invested in Research Systems' Interactive Data Language (IDL) which was used for the purpose of data visualization of the SCIFER data set. IDL's powerful graphical user interface toolkit is used for creating the interactive graphical display.

The program structure is made up of functions and procedures preceding two event-handling procedures and a procedure which sets up, creates and initializes the widgets (see section 3.2). Then the control is handed over to the window manager.

All event driven widget programs are structured in this same manner. There are two parts: a definition module and an event handler module. In the WASAT application structure there are additional functions and procedures, a definition procedure, and two event handling procedures in the event handler module (see section 3.2.2).

3.1 Implementation of I/O routines and data structure

Referring to the DITES data structure (see figure 2) it seems natural to implement a design that reflects this structure in the application. The purpose is to make the data sets easy to handle. The steps leading up to viewing of the spectrograms are the following:

- Open and examine the SCIFER data set file i.e. initialize some values.
- Create a structure based on the values obtained.
- Reposition the file pointer to the desired location.
- Read the requested amount of data.
- Run through these data and select, convert and store UiOFFT words together with time information.
- Recalculate (i.e. to the proper range) UiOFFT words.
- Prepare and plot spectra.

Several procedures and functions are created to facilitate these steps. One important step which is not implemented as a procedure or function is the positioning of the file pointer.

The task is fairly simple. Since the DITES format is block oriented, input values (`start_value` and `stop_value` in seconds) needs to be converted to the appropriate block numbers. And then subtracted to obtain the number of blocks to read.

By inspection one block stores data for a $\sim$ 48.87 ms time slot. A number based on printing the timing information for the first and final frame in blocks positioned randomly throughout the data file. The positioning of the file position pointer, c, thus given by the following algorithm:

```
   delta_sec = 4.8877e-02
   start_value = fix(start_value/delta_sec)
   stop_value = fix(stop_value/delta_sec)
   no = stop_value - start_value
   c = long(HRlength*2) + long(DDHRlength*2) + long(dr_buf_size*2)*start_value
;;Position the pointer and
;;read into the data structure
   openr,dtafile,file,/get_lun
        point_lun,dtafile,c
        readu, dtafile, data
   close, dtafile
   free_lun,dtafile
```

3.1.1 Common blocks

First we have to set up four common blocks in order to make variables available throughout the program.

`wasat_common` contains variables used to handle the IDL data structure and its parameters.

`DITES_struct_common` contains variables obtained from the original DITES II format.

`info_common` contains variables for widget message handling

`plot_common` contains variables for plot window id and output filenames.

For the complete description of the content of these common blocks, please refer to the source code.

3.1.2 Functions and procedures

To make maintenance and future enhancements easier, the program structure is divided into functions and procedures. The subsequent list is a short description of the functionality of the functions and procedures used.

HEADERS This procedure reads and displays the information contained in the header structure of the DITES II format. Information is displayed as described in section 2.1.1.

READ_HEADERS This procedure reads and sets the critical values obtained from the headers. The values set here are used globally.

TIME This function converts eight time bytes, where the timing information is coded according to the scheme outlined in table 5, into readable time values in a structure and return this structure to the caller.

TOSEC This function uses a structure of time information to calculate and return a floating point value which is the number of seconds after lift of.

RANGE this function re-scales a signal value. Referring to the floating point representation outlined in figure 4 this routine extracts the exponent and the mantissa from the compressed signal value and does the numerical computation of the signal value to be presented.

First the range value for the spectrum is found. It can have up to 64 values, thus, is bit shifted 6 positions to the right. The range value is common for all samples in every spectrum.

Each signal value is also bit shifted. The IDL function `ishft(s,p)` where `s` is the scalar to be shifted, and `p` is the number of bit positions and the direction of the shift. Thus, `ishift(1,1)` will give the scalar value 2 since the first bit is shifted one position to the left. The signal value is checked to see if it is nil in which case its value is set to the range value.

```
for j = 0, tot_spectra-1 do begin
   RNG(j) = ishft(fix(plait(4,j)),-7)
   for i = 0, samples-1 do begin
      if (plait(i,j) EQ '0000'X) then $
        plait(i,j) = 0 $
      else begin
        EXP = ishft(fix(plait(i,j)),-7)
        MAN = plait(i,j) AND '007F'X
        plait(i,j) = 3.32193*alog10(MAN) $
                   + float(EXP) + (15-float(RNG(j)))
      endelse
   endfor
endfor
```

The algorithm outlined is consistent with the algorithm presented in equation 1 on page 10.

FIND_SPECTRAS This function selects two words, UiOFFT1 and UiOFFT2, in addition to the eight time bytes from each frame. The following is an excerpt of the selection and plaiting part of this algorithm.

```
for j = 0, n-1 do begin
   for i = 0, DDRframe-1 do begin
       FFT1 = fix(data(j).dr_record(*,i),(uiofft1-1)*2)
       FFT2 = fix(data(j).dr_record(*,i),(uiofft1+FFTW2OFZ-1)*2)
                                          •
                                          •
       plait(spectcount,h) = sync(0)
       plait(spectcount + 1,h)= sync(1)
       plait(spectcount + 2,h)= FFT1
       plait(spectcount + 3,h)= FFT2
       spectcount = spectcount + 4
                                          •
```

```
        TAL = data(j).dr_record((DDRmwords*2 - timebytes):*,i)
    endfor
endfor
```

The signal values are plaited and stored in a two-dimensional spectrum array, `plait(x,y)`, where `x` is the number of samples per spectrum and `y` is the number of spectrums specified by the user. The time bytes are stored in a separate one-diminsional array. Both the plaited signal values and the time bytes are returned to the caller.

The select loop starts with a search for the first occurence of a spectrum synchronization sequence (refer to the soruce code). All of the timing information at this point is discarded. A check is made for the entire sequence (see section 2.1.4 and figure 4). After a match, all subsequent signal values and corresponding timing information is stored in their respective arrays.

`DISPLAY` This procedure sets up the proper display routines, selects the requested plotting device, and displays the content of the three-dimensional spectrogram. The spectra are displayed in the preferred manner by the use of two predefined IDL routines `transpose` and `reverse`. The spectrogram is presented as an grey-scale intensity image overlaid with a contour plot which supplies the enumerations and labeling of the axis (IUG, 1995, p. 15-8, method 2).

3.2 The Graphical User Interface

The WASAT 's GUI interface is created using the IDL widget toolkit. A «widget» is defined to be a graphical element used in a widget program (ITC, 1996). Widget programs are organized hierarchically (see figure 5). A widget can for example be such things as buttons, text fields, draw area (graphics window), etc. The application is controlled via this graphical interface.

Each widget is required to have a specific relationship to other widgets as they are created, a socalled family tree of widgets.

3.2.1 Widget Bases

Referring again to figure 5, the top-level widget is named `MAIN13` or `Top_base`. A «base» is a widget which can contain or hold numerous other widgets including other base widgets. All widget programs start with a top-level base widget, which holds all the other widgets. Thus, `Top_base` has three children; `BASE2 (Main_menu_base)`, `BASE3 (info_base)` and `DRAW30 (display_area)`. All three children are created with `Top_base` as parent identifier. The top-level base widget is the start of all widget hierarchies, thus, this special widget is created without a parent identifier parameter.

It is essential to understand the purpose of each of the three children of the `Top_base`.

The `display_area` is the «canvas» for the display of processed spectras. All spectrograms will be displayed within this region. It is not resizable nor can it be scrolled. It transparently handles mundane tasks such as refreshing the window after it was occluded and then exposed.

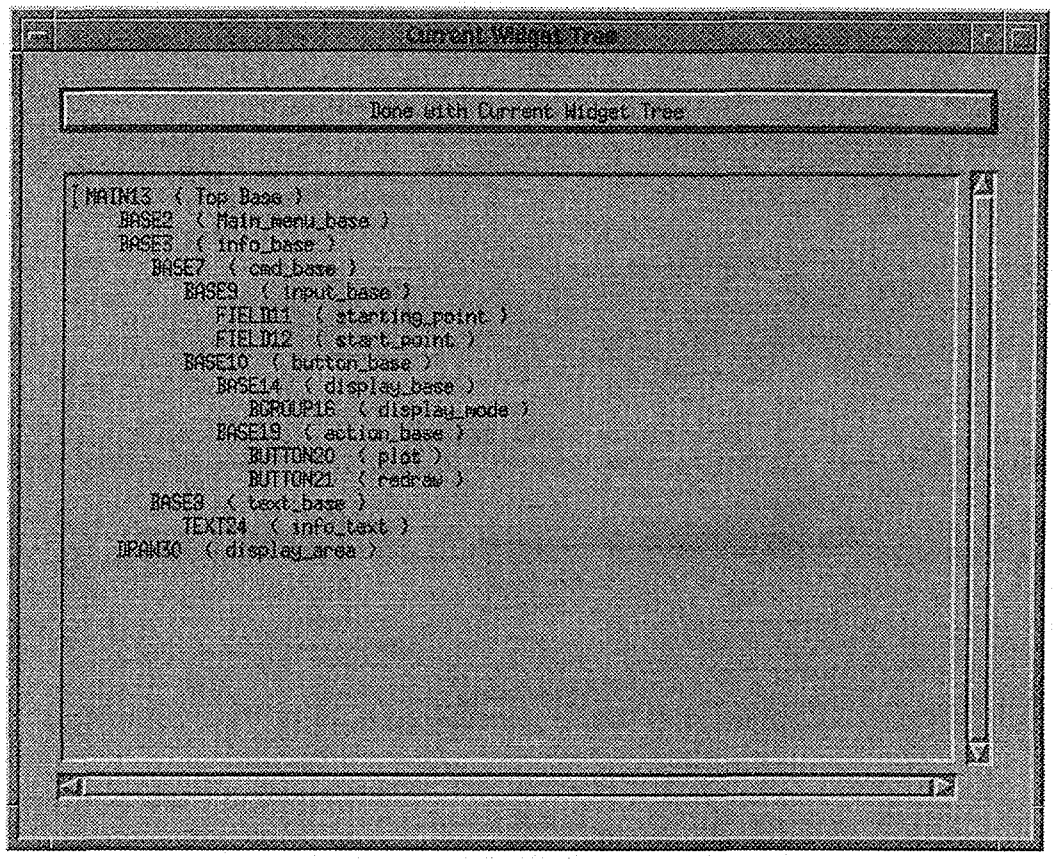

Figure 5: This illustration is a screen-dump from the IDL widget editor `wided` which show the application base structure. The extensive use of bases is primarily to obtain the desired visual appearence of the application.

The `info_base` is subdivided into a command base, `cmd_base`, for actions and input and a `text_base` for messages from the application. Nor size or appearence are affected by entering new values into the `cmd_base` or by manipulating the tools in the menu bar.

3.2.2 Implementation of the widget hierarchy structure and event loops

Widget programs are event driven, that is, they spend most time in an event loop doing nothing but waiting for the next event to occur. An «event» is normally some kind of user interaction (e.g., pressing a button). This is also the case for the WASAT application, but in addition avents are created by some of the previously described procedures.

When an event occurs the event driven program is supposed to process the event as quickly as possible (prefferably do something useful), return to the event loop and wait for the next event to appear.

There are two event handeling procedures in the WASAT application; `Menu_bar_event` and `Root_event`, in the event handler module, and one widget definition procedure, `wasat`, in the definition module. The code in the definition procedure is executed once while the

code in the event handeling procedures, at least in the case of `Root_event` procedure, are executed every time an event occurs in the program.

`Menu_bar_event` This procedure handles all events which originates from manipulating the menu bar in the application. Events to be processed here are handed over from the `Root_event`.

`Root_event` This procedure handles all events for the application. Events originating from the menu bar is passed by this event handler to the `Menu_bar_event` for detailed processing. The purpose of using two event handlers is to enhance efficiency in the countinous running event handeling processing. Thus, events not originating from the menu bar will not execute the event handeling code of the menu bar. Thus, saving processing time.

`wasat` This procedure defines and creates all the widgets in the application. Widgets are not displayed automatically after they are created, they are only stored in IDL's memory. Widgets are realized by executing:

```
; Realize top-level base and all its children
widget_control, Root, /realize
```

Also, default values are initialized and the data structure is created.

A widget event is a message returned from the window system as a result of interaction with the widget program. The IDL routine `XManager` registers the program with the window system, setting up the event loop, and monitoring the program for widget events that occur within it. Thus, before leaving the definition part a call to `XManager` is made:

```
; Hand things over to the X manager:
XManager, 'Root', Root, group_leader = group
```

4 User Manual

WASAT is written in `IDL Version 4.0.1 (sunos sparc)` and contains ~ 1100 lines of code (including comments), 10 functions and procedures and some 30 global variables in four «common blocks». The purpose of the WASAT user interface is to provide the analyst with easy-to-use software tools to manipulate and display spectrograms from VLF wave experiments.

The current version of the document applies to the WASAT version 2. WASAT is the property of the Research Section for Plasma and Space Physics, Department of Physics, University of Oslo and can only be acquired with explicit permission.

4.1 Menu contents

The WASAT application uses the available window manager, preferable the X window system and looks accordingly (see figure 6). It contains a single window with a menubar, an action/input area and a display area. This window is not resizable.

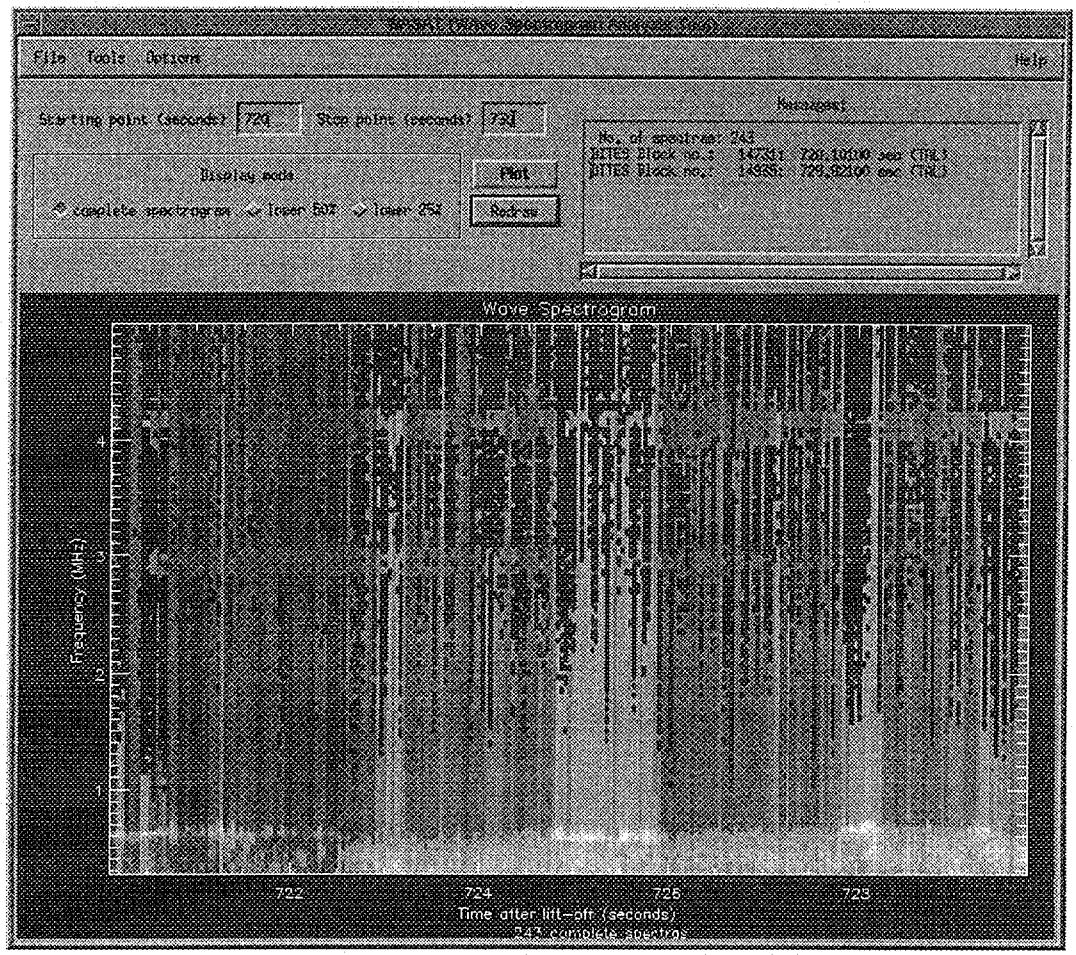

Figure 6: The WASAT application appearence when using the X window manager.

4.1.1 The menu bar

The menu bar contains four pull-down menus. The `File` menu is for manipulating files, saving plots in various graphical formats and `Quit` to terminate the application. Graphical formats is provided for rendering of the views in the display area. The implemented formats are encapsulated postscript, GIF[4] and JPEG[5]. The postscript format facilitates the use of WASAT generated illustrations/spectrograms in documents, reports and articles using e.g. LaTeX2$_\varepsilon$/Word, etc. An example of a postscript rendering is given in figure 7.

Some color routines are provided in the `Tools` menu. The tools provided are the standard IDL procedures `Xloadct` which contains $\sim$ 42 predefined color tables, and `Xpalette` for the detailed manipulation of the tables themself. These predefined procedures provides a

[4]GIF (Graphics Interchange Format): A standard defining a mechanism for the storage and transmission of raster-based graphics information.

[5]JPEG (Joint Photographic Experts Group): A standardized image compression mechanism. JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes.

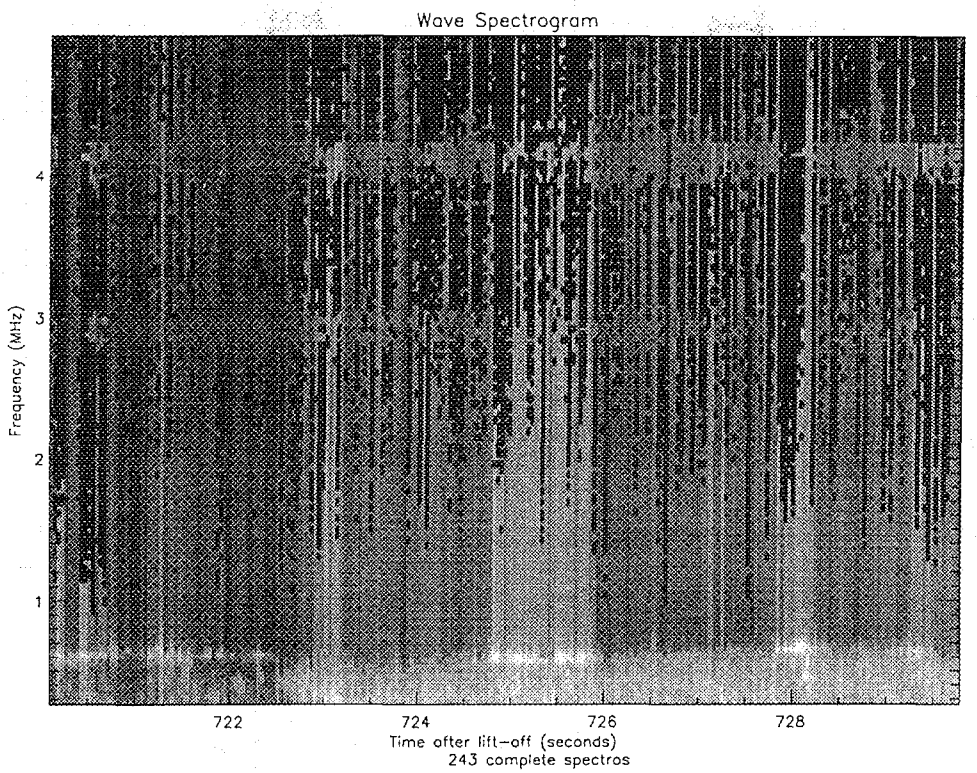

Figure 7: The figure shows an example of a postscript rendering.

widget interface to optimize the appearence of the current spectrogram. All changes due to the manipulation will be reflected automatically in the display area and in the rendering of a view by the use of the `save` option in the `file` menu in the menu bar.

Xloadct: This procedure provides a widget interface to routines with pre-defined color tables that can be loaded and manipulated using this tool. Tables can be stretched and Gamma corrected interactively.

Xpalette: This widget procedure allows the user to create own color tables using a set of three sliders. This procedure can interpolate the space between color indices (to create «smooth» color transitions) or edit individual colors.

A `Help` menu is placed to the far right as customed in most window systems. When the user selects «`Help`»from the menu bar a list of topics is presented. Clicking on an item brings up any available help.

4.1.2 The action/input area

In the action/input area are placed text fields, and several buttons placed in two groups. The two text fields are for submission of the start point and end point, in seconds, of a sequence in the rocket flight to be examined. Underneath is an area called «`Display mode`». Pressing one of the buttons gives a different display. For example, pressing the button labeled «`lower`

50%» displayes only the lower half og the spectrum, i.e., up to ~ 2.5 MHz. Default value is full spectrum.

After the submission of approriate values, the «Plot» button must be pressed to generate a spectrum. Also some central information, e.g., the number of spectrums plotted is displayed in the message window to the far right.

Whenever the display, for any reason, is cluttered or misaligned it should be sufficient to restore the visual appearence by pressing the «Redraw» button.

4.1.3 The draw area

All spectras are displayed in the graphical area which occupies most of the application space.

4.2 Getting started

There is no need for actual typing of any code to run the WASAT application.

A detailed description of the construction and maintanence is provided is sections 2 and 3 in addition to the extensive documentation within the program itself. The purpose here is to outline the basic steps used to create and manipulate a spectrogram. The below discussion is completely generic.

There are four steps that needs to be performed:

1. cd to the proper directory where the datafile(s) and the application is stored

   ```
   setenv WASAT /mn/aurora/ground/pc/rockets/scifer/wasat
   cd $WASAT/↵
   ```

   where the environment variable $WASAT is the directory where the program and the datafile are located. A permanent solution should be made by entering the first line into the $HOME/.user_cshrc-file.

2. Start up idl.
   ```
   idl↵
   ```

3. Run the IDL interpreter on the WASAT application.
   ```
   IDL> .compile wasat↵
   ```

4. Start up the application.
   ```
   IDL> wasat↵
   ```

The first two steps is not actually an application necessity, only that of the current operating system. Step 3, the WASAT module compilation, compiles the WASAT application and stores it in IDL's memory. The 4. step activates the application.

The user do not need to know any of the details of the WASAT compilation, initialization or how WASAT processes X window events or other interfacing routines. The application is controllable by either keyboard or mouse.

When creating a spectrogram the user will have to supply two values (in seconds) which represent respectively the beginning and the end point of the time slot to be investigated.

These two values must be given either as integer seconds or as floting point numbers. No other entries will be accepted.

After supplying the two values, press the «Plot» button. Appropriate messages will be displayed in the message area.

If the display by any reason appear cluttered or misaligned press the «Redraw» button.

The displayed spectrogram can easily be saved for subsequent printing to e.g., paper by using the «Save» option under the «File» menu in the menu bar.

5 Conclusion

Scientific researchers have a clear need for the capability of visualizing their numerical datasets quickly and efficiently. Results should be apparent at first view, and the time needed to produce the visualization should not be excessive. The crusial part of the visualization process is association of visual images with the numerical data.

The WASAT application provides a easy-to-use graphical interphase to the analysis of spectrogram data, tailored to handle two-dimensional spectrograms from the SCIFER sounding rocket campaign (see section 1.1). Also WASAT provides a suite of tools for visual manipulation of these spectrograms. The application is runnable under window systems for IDL. Preferably the X window system under which this application has been created.

5.1 Performance

There has not been any speed tests nor benchmark tests to measure WASAT's performance.

The calculation tasks performed are simple and not specifically time consuming. On the other hand, the data file is quite large, 218 Mbytes, and the search for correct words in each frame is a very time consuming task. A more thorough understanding of IDL array handeling routines will most certainly provide for a more efficient search routine (i.e., replacement of the FIND_SPECTRAS procedure) and ultimetly speed things up quite a bit.

Plotting is fairly simple and is assumed that not much gain in speed is worth the effort. However, blank plottings and annoying misalignments should be dealt with.

There are some noise in the spectras mainly due to drop-outs and not having implemented windowing in the original hardware design. On some occations, changing from full spectrogram to lower 50% will change the appearence of the displayed spectrogram. Thus, some improvements can be made to enhance the visual appearence of the spectrograms.

5.2 Lessons learned

The intention of creating such an application as WASAT was to

- allow the user to produce and manipulate spectrograms with ease
- make it robust enough so it doesn't crash when someone does something unexpected.

An ancient law of computer programming states that a 100% robust program is not able to perform anything. WASAT has a few caveats, and thus, is not very robust. Unfortunatly, due to the lack of manpower WASAT is not insured to do something reasonable with unsuspected actions or events.

5.2.1 Variable Spectrum Length and end-of-spectras

One of these caveats are due to the variable length of a spectrum. The application can crash if the spectrums are found to be much longer than 128 samples including the syncronization sequence.

Also, the last data stored on the CD-ROM file are from around 990 seconds time of flight. This is around 16 minutes in flight, Telemetry lost contact with the rocket around this point.

The only remedy existing when this application crashes is to restart the application, and alter, prefferably reduce the time slot requested.

Nevertheless, most of the data of interest has been test run several times without such sideeffects. Should the application crash, enter to the IDL-prompt the following sequence, and replace the numbers in the text fields.

```
IDL> retall
IDL> xmanager
```

5.2.2 Non-linear time positioning

Ideally the time elapsed between each successive block, e.g., between two data record headers, in the DITES format should be constant. That is, each data block containes an amount of data collected of equal amount of time for each block. Based on this assumption a simple algorithm was implemented to allow for the positioning of a file pointer (see section 3.1). There is an enormous time saving in not having to read through the entire file to find the appropriate place to start obtaining and processing data.

Tests has proven that the assumption of a linear time lapse per block throughout the data file unfortunately is false.

Depending on which part of the rocket flight that is examined the spectrograms generated can be premature or too late. An example is the time slot 760 – 770 seconds after liftoff. When entering these numbers the algorithm places the position pointer at block nr. 15 550 which contains data from 759.008 seconds into the flight. That is, one whole second, $\approx$ 50 spectrums premature. On the other hand, if the desired time slot is 360 – 370 seconds, the displayed data are from the 370.2 – 379.9 time slot. That is, ten seconds too late.

One immediate solution is to run through all the data once, store in an array the time information from the first frame in each block together with the block number and use this array as a lookup table to be able to place the file pointer correctly. Such a scheme has not been implemented yet.

5.3 Points for further development and adaptations

This application was designed with the intent to be used by subsequent flights adhering to the DITES format. To tailor the program code some global values needs to be replaced, such as:

```
file    = '../d4000611.dat'
uiofft1 = 11
fftw2ofz= 25
samples = 128
```

```
Time_of_lift_off = [0,2,5,0,6,2,4,4,8,0,0,0,0,0]
```

However there might be necessary to do additional changes then just changing the above mentioned global constants.

Acknowledgement

This work was performed at the Research Section for Plasma and Space Physics, Department of Physics, University of Oslo. The author wish to thank Efim Bronz, Lars T. Lyngdal, University of Oslo for their thorough insight into the SCIFER campaign technical obscurities and Bård Krane for his insight into the world of IDL.

References

Andersen, B. and Sørensen, B. (1996). Space Research in Norway 1995, *annual report*, Norwegian Space Centre, Oslo, Norway.

Bretthauser, J. W. (1994). Telemetry Data Distribution System; Distribution Media Formats. NASA/GSFC/WFF Code 622.3/Telemetry Systems Section. Draft.

Brondz, E. (1989). A two-channel wave analyzer for sounding rockets and satellites, *Report Series; UiO/PHYS/89–03*, Department of Physics, University of Oslo, Norway.

IRG (1995). *IDL Reference Guide.* Version 4, Volume 1–2.

ITC (1996). *Learning IDL; An interactive training course.*

IUG (1995). *IDL User's Guide.* Version 4.

Jørgensen, R. (1995). *Study of the Small Computer System Interface (SCSI) in Ground Based Space Research Equipment*, Master's thesis, Department of Physics, University of Oslo, Norway.

Kintner (1995). Cornell 1.6 MB PCM format/PCM measurement list, Cornell University, USA. SCIFER – 40.006 Internal notes. Draft.

WFF (1988). Sounding Rocket User's Handbook, *Reference manual*, NASA Goddard Space Flight Center, Wallops Flight Facility, USA.

A Program Listing

In this section are listed some useful IDL programs for examination of the DITES II format headers.

A.1 raw.pro

```
; Routine for printing the headers
; without any formating in raw form

; Data are printed either as ASCII or HEX coded.
; Created by Runar Jrgensen
; Date: Fri Jun 28 11:19:26 1996

HRlength =46
DDRlength=20
DRprelength=10
TWR = 52 ; Total words in Record
Frames_per_buffer = 76

a=bytarr(132+(3962*2)+20+52*2)

file = '/mn/aurora/ground/pc/rockets/scifer/rockets2/d4000611.dat'
openr,df,file,/get_lun
readu,df,a
close,df
free_lun,df

print,'-----------------------------------------------------------------
-'
print,'Header Record'
print,format = '(18A4)',a(0:(HRlength*2)-1) ;46 words in Header Record
print,'-----------------------------------------------------------------
-'
print,'Data Description Record'
print,format = '(18A4)',$       ;20 words in Data Description Record
  a(HRlength*2:((HRlength+DDRlength)*2)-1)
print,'-----------------------------------------------------------------
-'
print,'Data Record'
data_block= ((HRlength+DDRlength)*2+DRprelength*2)-1
print,format = '(18A4)',$
 a(((HRlength+DDRlength)*2):data_block)
print,'-----------------------------------------------------------------
-'
print,' '
for i=0, Frames_per_buffer-1 do begin
    print,'Frame no:',i+1,' ------------------------------------------
--------'
    print,format = '(18Z4)',$
      a(data_block+1+(Total_Words_in_Records*2)*i:data_block+(TWR*2)*(i+1))

END
```

A.2 headers.pro

```
; Routine for printing header information
; contained in NASA/GSFC/WFF DITES II formats
;
file = '../d4000611.dat'
;HRlength=46
testbytes = BYTARR(400)

;*******************
; Header Record
;*******************
openr,dtafile, file, /get_lun    ;openr for reading only
f=fstat(dtafile)                 ;Get some file information
readu,dtafile,testbytes          ;Read the first bytes to check
close,dtafile
free_lun,dtafile

;Extract relevant fields (byte) from the header
HRM1B = byte(testbytes,0)        ;Should be FFx
HRM2B = byte(testbytes,1)        ;Should be FDx
HRlength = fix(testbytes,2)      ;Number of words in Header Record

print," "
print,'              Printing Header Information '
print,' File: ',f.name,' Size:',f.size/1000000.,' MBytes'
print,' Header Record'
print,'-------------------------------------------------------'
print, format = '("Record Marker (W1):",2Z3)',HRM1B, HRM2B
print, format = '("Header Record Length (W2):", I3)', HRlength
print, format = '("Header Record ID (W3):",I3)', fix(testbytes,4)
print, format = '("Stream ID (4 characters; W4 & W5): ",2A1,2A1)', $
  string(byte(testbytes,6)), string(byte(testbytes,7)), $
  string(byte(testbytes,8)), string(byte(testbytes,9))
print, format = '("CD/Tape number (W6):",I3)', fix(testbytes(11))
print, format = '("ASCII string test (W7-W46): ",A0)', $
  string(byte(testbytes(12:*)))

raw=testbytes
new_point=HRlength*2
testbytes= testbytes(new_point:*)
DDRlength=fix(testbytes,2)

print,' '
print,'Data Description Record'
print,'-------------------------------------------------------'
print, format = '("Record Marker (W1):", 2Z3)',$
 byte(testbytes,0),byte(testbytes,1)
print, format = '("Data Description Record Length (W2):",  I4)', $
  fix(testbytes,2)
print, format = '("Data Description Record ID (W3):", 2Z3)',$
  byte(testbytes,4), byte(testbytes,5)
print, format = '("Run ID    (4 characters; W4 & W5): ", 2A1,2A1)',$
  string(byte(testbytes,6)),string(byte(testbytes,7)),$
  string(byte(testbytes,8)),string(byte(testbytes,9))
```

```
print, format = '("Stream ID (4 characters; W6 & W7): ", 2A1,2A1)',$
  string(byte(testbytes,10)),string(byte(testbytes,11)),$
  string(byte(testbytes,12)),string(byte(testbytes,13))
print, format = '("Words per frame (W8):",  I4)', fix(testbytes,14)
print, format = '("Frames per buffer (W9):",  I4)', fix(testbytes,16)
print, format = '("Scaling factor (W10):", I3)', fix(testbytes,18)
print, format = '("Merged words (time words in record; W11):", I3)',$
  fix(testbytes,20)
print, format = '("Total words in record (W12):", I3)', fix(testbytes,22)
print, format = '("Preface words above 8 (W13):", I3)', fix(testbytes,24)
print, format = '("Buffer appendix size  (W14):", I3)', fix(testbytes,26)
print, format = '("Input buffer size [words]  (W15):", I5)', $
  fix(testbytes,28)
print, format = '("Output buffer size [words] (W16):", I5)', $
  fix(testbytes,30)
print, format = '("Frames per subframe 1 (W17):", I3)', fix(testbytes,32)
print, format = '("Frames per subframe 2 (W18):", I3)', fix(testbytes,34)
print, format = '("Record Mode (0 = word; 1 = byte) (W19):", I3)', $
  fix(testbytes,36)
print, format = '("Two ASCII blanks (W20):", 2I4)',$
  byte(testbytes,38),byte(testbytes,39)

testbytes = testbytes((DDRlength*2):*)
DRlength = fix(testbytes,2)
print,DRlength
print,' '
print,'Data Record
print,'-------------------------------------------------------------------
-'
print, format = '("Record Marker (W1):",T27,2Z3)',$
  byte(testbytes,0),byte(testbytes,1)
print, format = '("Data Record length (words)(W2):",I5)', fix(testbytes,2)
print, format = '("Block number (W3):",T26,I4)',fix(testbytes,4)
print, format = '("Stream ID (W4 & W5): ",T26, 4A1)', $
  string(byte(testbytes,6)),string(byte(testbytes,7)),$
  string(byte(testbytes,8)),string(byte(testbytes,9))
print, format = '("(MSB) Appendix words; Preface words (W6):",I4,";",I4)',$
  byte(testbytes,10), byte(testbytes,11)
print, format = '("Input buffer size (W7):",T31,I5)', fix(testbytes,12)
print, format = '("TFE Status (bitflags):",T32,Z4)',  byte(testbytes,14)
print, format = '("Frame number in block (W8,LSB):", I4)',byte(testbytes,15)
print, format = '("Mag Tape Write Status (W9):",T34,I2)',fix(testbytes,16)
print, format = '("Not used (nil, W10):",T32,I4)', fix(testbytes,18)
print,' '
print,format = '(18A4)',raw(0:(HRlength*2)-1) ;46 words in Header Record
print,'-------------------------------------------------------------------'
print,format = '(18A4)',$       ;20 words in Data Description Record
  raw(HRlength*2:((HRlength+DDRlength)*2)-1)
print,'-------------------------------------------------------------------'
print,format = '(18A4)',$
  raw(((HRlength+DDRlength)*2):((HRlength+DDRlength)*2+10*2)-1)
end
```

FYSISK INSTITUTT

FORSKNINGS-GRUPPER

Biofysikk
Elektronikk
Elementærpartikkelfysikk

Faste stoffers fysikk
Kjerne- og energifysikk
Plasma- og romfysikk
Strukturfysikk
Teoretisk fysikk

DEPARTMENT OF PHYSICS

RESEARCH SECTIONS

Biophysics
Electronics
Experimental Elementary Particle Physics

Condensed Matter Physics
Nuclear and Energy Physics
Plasma and Space Physics
Structural Physics
Theoretical Physics

ISSN - 0332 - 5571

Universitetsbibliotekets trykkeri
Oslo