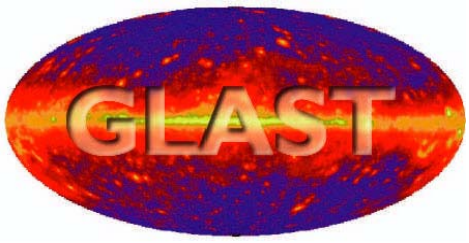

LAT Flight Software

Thermal Control System

Number: LAT-SS-02896
Subsystem: Data Acquisition/Flight Software
Supersedes: None
Type: Design Description
Author: J.J.Russell
Created: 16 October 2003
Updated: 22 January 2004
Printed: 23 August 2005

The requirements and specification of the LAT's Thermal Control System are presented. The emphasis is on the Flight Software aspects of the system, but a description of the hardware is presented to provide necessary context and where the realities of the hardware impact the software design.

Document Approval

Prepared By:			Oct 14,2003
	J. J. Russell	LAT Flight Software	Date
Approved By:			
	G.Haller	LAT Electronics Manager	Date
Approved By:			
	J.J.Russell	LAT Flight Software Manager	Date
Approved By:			
	<name>	<title>	Date

Record Of Changes

Version	Date	Title Or Brief Description	Entered By
0.1	14 October 2003	Draft Release	J. J. Russell

Contents

0 Thermal Control System 1
0.0 Introduction and Overview 1
0.1 Physical Elements 1
0.2 Software Components 1
0.3 Reference Documentation 2
1 Configuration 3
1.0 Master Startup Script 3
1.0.0 System Reinitialization 3
2 Inputs 4
2.0 Selection of Active Sensors and Heaters 6
2.0.0 Uploading of the Active Sensor Table 6
2.0.1 Method of Storage 6
2.0.2 Selection of the Active Sensor Table 6
2.0.3 Validity Checking 7
2.0.4 File Contents 6
2.0.4.0 Heater Element Selection 7
2.0.4.1 Sensor Id Format 5
2.1 Read Rate 7
2.2 Conversion 7
2.2.0 Conversion Process and Parameters 7
2.2.0.0 ADC to Voltage Conversion 8
2.2.0.1 Voltage to Resistance Conversion 8
2.2.0.2 Resistance to Temperature Conversion 8
2.2.1 Maintenance of Conversion Parameters 8
2.2.1.0 Format of the ADC to Resistance Conversion File 8
2.2.1.1 Format of the Voltage to Temperature Conversion File 8
2.2.1.2 Validity Checking 8
2.2.2 Conversion Method 9
2.2.3 Output Representation 9
2.3 Filtering the Readings 9
3 The Algorithm 10
3.0 Parameters 10
3.1 New Parameters 10
3.2 Method of Storage 10
3.3 Validity Checking 11
3.4 Representation 11
3.5 File Contents 11
3.6 Selection 12
3.6.0 The Algorithm 12
4 Monitoring 15

Figures

Figure 1 Algorithm example 14

Tables

Table 1 Active Heater and Sensor File Format 6
Table 2 Sensor Id Address Format 5
Table 3 The Algorithm Parameters for each Heat Pipe 10
Table 4 Data Contents of the TCS Algorithm Parameter File 11
Table 5 Monitoring Record Format 15

0 Thermal Control System

0.0 Introduction and Overview

The LAT Flight Software is responsible for implementing the Thermal Control System (TCS). This system is meant to fine-tune the thermal profile of the LAT with the goal of minimizing distortions to the mechanical structure. The TCS has no safety or survival implications. Those duties are left to the Spacecraft. If the Spacecraft determines that the thermal characteristics are out of tolerance, the Spacecraft takes over control of the system by simple shutting off the LAT's SIU.

0.1 Physical Elements

The physical system consists of 12 heater pipes, 6 on each of the GLAST's two radiators. Each heat pipe is equipped with two temperature sensors and one heater controller. The two temperature sensors measure the temperature of the ammonia in the reservoir and Radiator Interface Temperature (RIT). There are two types of sensors. All the reservoir temperature sensors are of one type, and all the RIT sensors are of another type.

Both the sensors and activators for the heaters have a primary and a redundant set. The primary sensors are readout by the primary Power Distribution Unit (PDU)[1] and the redundant sensors are readout by the redundant PDU. The readout method is via normal LATp[2] commands. The heater activators are on the SIB [4], accessible by SIU by a direct PCI read/write operation.

Control is exercised by simple turning the heaters on or off for a fixed period of time. No proportional control exists. Note that while physically there are primary and redundant heatere switches, the SIB provides no individual control; any control signal from the SIB always drives both the primary and redundant heater switches.

In the interest of completeness, it is worth mentioning that the SIB board has a watchdog timer overseeing the heater switches. If no write is performed to the heater control register on the SIB board within a 1 minute timeframe, the watchdog activates the heater switches.

0.2 Software Components

The major software components of the Thermal Control System are

1. Managing the configuration parameters. This is includes active sensor specification, the algorithm's parameters, etc.
2. Preparing the input temperatures
    a. Reading the ADC values from the PDU

   b. Converting the ADC readings to Celsius
   c. Filtering the input temperatures for bad values
3. Executing a periodic algorithm that determines whether to turn on the heater or not.
4. Applying the heater element settings by sending the appropriate commands over the LAT's internal communication network.
5. Maintaining a history of input parameters, filtering action and output state.

0.3 Reference Documentation

1. LAT-TD-01543, "Power Distribution Unit - Programming ICD specification", by Michael Huffer
2. LAT-TD-00606, "LAT Inter-module Communications", by Michael Huffer
3. LAT Flight Software Document, "Telecommand and Telemetry Formats", by Dan Wood
4. LAT-SS0-1539, "Storage Interface Board (SIB) Hardware Specification", by Dennis Silver
5. TCS algorithm, private communication from Jeff Wang

1 Configuration

The LAT FSW shall maintain and manage the configuration used by the Thermal Control System. The configuration specification falls into 4 distinct classes

1. Specification of which sensors and heaters are to be associated with each heat pipe
2. Parameters used in converting sensor ADC readings to degrees Celsius
3. Parameters controlling the algorithm
4. Parameters controlling the monitoring process

Configuration information is stored in the on-board file system. These files are updatable by a file upload telecommand [3]. A master file acts as a startup and configuration script.

1.0 Master Startup Script

The TCS software shall read the master startup script upon TCS system initialization. The master startup script consists of the names of files containing the following subscripts

1. File specifying the mapping of the sensors to physical ADCs
2. File specifying the ADC to temperature conversion parameters
3. File specifying the algorithm parameters, includes active sensors and heaters
4. File specifying the TCS control and monitoring parameters.

Note that there is no direct information in the master startup script, only the names of files containing the information. This method allows one to keep a number of alternate subscripts within the file system, using the master startup script as the selection mechanism.

1.0.0 System Reinitialization

Before any change can be made to the TCS system, it must be shutdown and reinitialized. This process is anticipated to be fast on the scale of the thermal system, so that reinitializing the TCS software will cause no significant perturbation to its functionality.

The remaining chapters are organized along functional lines. The exact format and contents of these files are described in the appropriate functional block.

2 Inputs

Management of the input measurements can be broken into four primary functions

1. Mapping of sensors to physical ADCs
2. Selection of which sensors to read
3. Reading of the physical sensors
4. Converting the readings to engineering units
5. Filtering the readings

2.0 Mapping Of Sensors to Physical ADCs

The 48 sensors (primary and redundant for the temperature reservoir and RIT sensors x 12 heat pipes) are each associated with an ADC on the PDU. Since this information is very static, it is broken off in a separate table.

2.0.0 Uploading

The mapping table shall be uploadable by a file upload telecommand. Because this table is anticipated to very static, the contents of this table are very unlikely to change. The table is broken off as a separate entity for management and modularity reasons.

2.0.1 Method of Storage

The mapping tables shall be maintained as named files in the LAT's onboard file system.

2.0.2 Selection

The TCS master configuration script shall specify the name of the mapping table to be used. The ground shall check the validity of sensor mapping table. Examples of common errors are specification of an invalid sensor id and mapping two or more sensors to the same ADC.

2.0.3 File Contents

The contents of the file shall be maintained as 12 sets of 4 numbers in the following format. Each set corresponds to 1 of the 12 heat pipes. The 4 numbers correspond to the 4 sensors associated with that heat pipe.

Set	Offset	Width (bytes)	Fields/Meaning
Sensors Heater Pipe 0	0x00	1	Primary Reservoir Temperature Sensor ID
	0x01	1	Primary RIT Sensor ID
	0x02	1	Secondary Reservoir Temperature Sensor ID
	0x03	1	Secondary RIT Sensor ID
Sensors Heater Pipe 1	0x04	1	Primary Reservoir Temperature Sensor ID
	0x05	1	Primary RIT Sensor ID
	0x06	1	Secondary Reservoir Temperature Sensor ID
	0x07	1	Secondary RIT Sensor ID
.	.	.	.
Sensors Heater Pipe 11	0x2B	1	Primary Reservoir Temperature Sensor ID
	0x2C	1	Primary RIT Sensor ID
	0x2E	1	Secondary Reservoir Temperature Sensor ID
	0x2F	1	Secondary RIT Sensor ID

Table 1 Active Heater and Sensor File Format

2.0.3.0 Sensor Id Format

Each sensor ID shall be one byte in size. The value consists of two fields, one specifying the sensor's ADC group and the other the channel number within the PDU. Pictorially, the format is

G_2	G_1	G_0	C_4	C_3	C_2	C_1	C_0

Table 2 Sensor Id Address Format

All group numbers (0-7) are valid, but only channel numbers 0-19 are valid. For example, VCHP Reservoir Heater 6 address is ADC group 0, channel 5.

2.0.4 Validity Checking

The ground shall check the validity of the mapping table table prior to file upload,

In addition, the TCS system will check for invalid sensor id's or the mapping of 2 or more sensors to the same sensor id will be treated as a fatal error. This check will be done prior to file commitment and also prior to usage.

2.1 Selection of Active Sensors and Heaters

An input parameter table shall determine which sensors are read and which sensors and heaters are used in the thermal control system algorithm. Note that these are two distinct sets of data maintained in a single table

1. The set of sensors to read
2. The set of sensors and heaters to use in the algorithm

The second set of sensors **must** be a proper subset of the first. Sensors that appear only in the first set will be read and reported in the telemetry. Sensors that appear in both will be read and used in the TCS algorithm.

All sensors are read from the PDU's[1] collection of ADCs using LATp[1,2] commands. The mapping file discussed in section 2.0 is used to construct the LATp commands.

The heaters are accessible on the SIB[4] via direct PCI read/write operations.

2.1.0 Uploading

An active sensor and heater table shall be uploadable by a file upload telecommand.

2.1.1 Method of Storage

Active sensor tables shall be maintained as named files in the LAT's onboard file system.

2.1.2 Selection

The TCS master configuration script shall specify the name of the active sensor table.

2.1.3 File Contents

The contents of the file shall be maintained as 4 sets of 12 bit numbers in the following format.

Set	Offset	Width (in bytes)	Fields/Meaning
Read List, Temperature Sensors	0x0	2	List of the Primary Reservoir Temperatures
	0x2	2	List of the Primary RITs
	0x4	2	List of the Second Reservoir Temperatures
	0x8	2	List of the Secondary RITs
Active List Temperature Sensors	0xA	2	List of the Primary Reservoir Temperatures
	0xB	2	List of the Primary RITs
	0xC	2	List of the Second Reservoir Temperatures
	0xE	2	List of the Secondary RITs
Active Heater List	0x10	1	List of the 6 Left and 6 Right Heater Elements

Table 3 Sensor Selection File Format

2.1.3.0 Sensor and Heater Element Selection

The sensor/heater element selection consists of following 12 bit mask. The least significant bit corresponds to heater pipe 1.

				LH6	LH5	LH4	LH3	LH2	LH1	RH6	RH5	RH4	RH3	RH2	RH1

Table 4 Active Sensor and Heater Selection List Format

A *0* indicates not to use that sensor/heater and *1* indicates to use that sensor/heater. The grayed out boxes are unused and must be programmed to 0. Note that disabling either one or both of the heaters does not alter the reading or processing of their associated sensors. It only affects whether that heater will be activated if requested by the TCS algorithm.

2.1.4 Validity Checking

The ground shall check the validity of the active sensor table.

In addition, FSW shall check that MBZ bits that are 0 and any algorithmic use list that is a proper subset of its corresponding read list. If either is condition is not satisified, they will be treated as fatal errors.

2.2 Read Rate

All active sensors will be read 3 times at the beginning of the main loop execution.

2.3 Conversion

The raw ADC readings shall be converted to Celsius. This conversion will be controlled by a set of parameters stored in the on-board file system.[1]

2.3.0 Conversion Process and Parameters

The conversion is logically a three-stage process, first converting the ADC counts to a voltage, then converting the voltage to a resistance and finally converting the resistance to a temperature. Thus, there are three sets of parameters to consider,

1. The ADC to voltage conversion
2. The voltage to resistance conversion
3. The resistance to temperature conversion.

In practice, one can combine these three transformations into a single transformation, taking an ADC value directly to a temperature However, from a maintenance viewpoint, it may prove to be wise to store the transformations separately and combine them during TCS initialization time.

[1] This section on converting ADC counts to engineering units is provided possibly only for completeness. It may be this service will be provided as a part of the normal housekeeping gather function.

2.3.0.0 ADC to Voltage Conversion

This conversion is a property of the chosen ADC. A linear conversion, specifying a gain and an offset will be used. This assumes the differential non-linearity is small compared to the accuracy desired.

2.3.0.1 Voltage to Resistance Conversion

This conversion is a property of the electrical circuit. The circuit serves two functions

1. Provides a convenient place to measure a voltage
2. Maps this voltage into the usable range of the ADC.

The conversion is straightforward but may change over time if the reference resistor or the source voltage varies. Variation in the reference resistor or source voltage effectively changes the offset and gain. It is likely very hard to separate the effect of the ADC, see section 2.3.0.0, from this effect. Therefore, in practice, these two effects will be convolved together.

2.3.0.2 Resistance to Temperature Conversion

The resistance to temperature conversion is provided in table format. Depending on the type of sensor purchased, there may be a table for each sensor or possibly a table for each sensor type. The salient point is that this table is a property of the sensor and is stable for that sensor for the lifetime of the mission. This is contrast to the previous two conversion which may drift over time.

2.3.1 Maintenance of Conversion Parameters

In general, two files shall be maintained for each sensor.

1. A set of two parameters specifying a linear transformation of ADC counts to a resistance
2. A resistance to temperature table

Here, it has already been admitted to that the first two transformation, see sections 2.3.0.0 and 2.3.0.1, will be specified as a single transformation.

2.3.1.0 Format of the ADC to Resistance Conversion File

This will be specified as two constants giving the offset and gain of a linear transformation. The ADC is a 12-bit ADC, (0 to 4095) with physical reading likely being limited to less than 3000. This conversion will map the ADC reading into a resistance range of 2-2500 ohms for the RIT sensors and 900,000-1500 ohms for the reservoir sensors. These constants will be determined by placing the sensor on a plate of a known temperature.

This name of this file will identify the sensor. For example RITP0 would identify heat pipe 0's primary RIT sensor.

2.3.1.1 Format of the Voltage to Temperature Conversion File

Each sensor comes with a table specifying the conversion of volts to temperatures. Experience will dictate whether a distinct table is needed for each sensor or whether a universal table can be used for each sensor type.

2.3.1.2 Validity Checking

Validity of the temperature conversion parameters shall be the responsibility of the uploader.

2.3.2 Conversion Method

The conversion method shall be implemented as a table lookup, followed by a linear interpolation. To accomplish this, code run at system initialization time will convolve the resistance to temperature tables and the gain and offset values for each sensor to produce a table converting ADC counts to a temperature range. Once the temperature range has been located, the final temperature value will be computed by linear interpolation within the range.

2.3.3 Output Representation

The resultant temperature shall be represented as signed 32 bit numbers, with the most significant 24 bits devoted to whole degrees C and the least significant 8 bits devoted to fractional degrees C. This gives a very large range (+/- 16,000,000 C), with a precision of 1/256 C. The range is far beyond anything that the LAT will encounter, however there is no advantage of limiting this to 16 bits. The precision exceeds the capability of the sensors, but does allow a number of arithmetic operations to be performed without losing significant digits.

2.4 Filtering the Readings

The goal of the filtering process is to produce a representative and believable number for each of the 12 sensor sights. Only active sensors, see section 2.1.3, will be considered.

The resultant temperature values of working sensors will be subjected to a consistency checking and smoothing algorithm. The exact algorithm has not been determined. This will be done once the characteristics of the input sensors are learned. The following is meant to illustrate the nature and general features of such an algorithm

1. The three reading will be checked for consistency against the known accuracy of the sensors.
    a. If the three measurements are consistent, an average of the three will be used. If the measurements are not consistent, the hi and lo value will be discarded, leaving the middle value as the working value.
2. When both primary and redundant sensors are present, the working values of two sensors will be compared against each other for agreement.
    a. If the two values are consistent with the expected accuracy, the average of the two values will become the working value for this sensor position.
    b. If the two working values disagree, both will be compared to a recent history of the readings. The value most consistent with the recent past will be used.
    c. If neither sensor is close to the recent history, a smoothed value will be used for that iteration.
    d. If two such extrapolations are used in succession, the sensor will be flagged as broken. The LAT must then implement a response policy. The simplest policy is to declare that the system is inoperable and return control of the thermal system to the Spacecraft. This would necessarily mean terminating all other activity, including data taking by the LAT.
3. If only one of the two sensors is operating, this value will be taken as the working value and subjected to the same consistency checks with the recent history.

3 The Algorithm

Lockheed provides the LAT's FSW team the TCS algorithm in conceptual form[5]. The FSW team translates this algorithm into actual code. The FSW team is responsible for managing and updating this code.

The algorithm itself is relatively simple. It treats each of the 12 sensor/heater pairs as distinct objects; there is no cross-coupling between the elements. Not only are the heat pipes treated as in isolation, the same algorithm runs on each of the heat pipes, although with a different set of parameters.

The algorithm inputs consist of

1. A set of 4 static but uploadable and updatable parameters controlling the algorithm
2. The 12 input working values for the sensors
3. A 12 x 1 bit state vector reflecting the action taken on the previous iteration of the loop.

3.0 Parameters

There are 4 parameters for each of the 6 heater pipes. These parameters, along with representative values are

Parameter Name	Meaning	Representative Value
ResLo	Reservoir Low	-65.0° C
RitLo	RIT Low Limit	-5.0° C
RitHi	RIT High Limit	-4.0° C
DeadBand	Deadband region	6.0° C

Table 5 The Algorithm Parameters for each Heat Pipe

3.1 New Parameters

The LAT FSW shall accept new sets of parameters via a 1553 file upload telecommand.

3.2 Method of Storage

Sets of parameters will be maintained as named files in the LAT's onboard file system.

3.3 Validity Checking

The ground shall check the validity of the parameters and any constraints imposed on them. FSW will reject any parameter set with RitLo >= RitHi.

3.4 Representation

The parameters shall be represented as signed 16 bit quantities, with the upper 8 bits representing whole degrees Celsius and the lower 8 bits representing fractional degrees. This gives a range of –127 to +128 C^o and a precison 1/256 C^o. This affords adequate range and precision.

3.5 File Contents

The contents of the parameter file shall consist of 4 parameters for 6 heat pipes. These parameters and their layout is

0x00	Heater Pipe 0 - Reservoir Low Limit
0x02	Heater Pipe 0 – RIT Low Limit
0x04	Heater Pipe 0 – RIT High Limit
0x06	Heater Pipe 0 – Deadband
0x08	Heater Pipe 1 - Reservoir Low Limit
0x0A	Heater Pipe 1 – RIT Low Limit
0x0C	Heater Pipe 1 – RIT High Limit
0x0E	Heater Pipe 1 – Deadband
0x10	Heater Pipe 2 - Reservoir Low Limit
0x12	Heater Pipe 2 – RIT Low Limit
0x14	Heater Pipe 2 – RIT High Limit
0x16	Heater Pipe 2 – Deadband
0x18	Heater Pipe 3 - Reservoir Low Limit
0x1A	Heater Pipe 3 – RIT Low Limit
0x1C	Heater Pipe 3 – RIT High Limit
0x1E	Heater Pipe 3 – Deadband
0x20	Heater Pipe 4 - Reservoir Low Limit
0x22	Heater Pipe 4 – RIT Low Limit
0x24	Heater Pipe 4 – RIT High Limit
0x26	Heater Pipe 4 – Deadband
0x28	Heater Pipe 5 - Reservoir Low Limit
0x2A	Heater Pipe 5 – RIT Low Limit
0x2C	Heater Pipe 5 – RIT High Limit
0x2E	Heater Pipe 5 – Deadband

Table 6 Data Contents of the TCS Algorithm Parameter File

3.6 Selection

During runtime initialization, the TCS system shall select two such files, one for each radiator, by file name. The names of files will be made known to the system in the Thermal Control master initialization script. Nothing prohibits the system from using the same set of parameters for both radiators.

3.6.0 The Algorithm

The algorithm is executed in a 1 second periodic loop. The algorithm is present below. The return value of the algorithm is two 12 x 1 bit vectors. One 12 bit vector indicates which heaters should be turned on. The other 12 bit vector keeps track of state information. This state information is passed into the algorithm on the next pass. If the next pass of the algorithm does not occur within 10 secs (TBD) of the previous pass, the state vector will be reset to all *0's.*

For illustration purposes only, the algorithm is presented in the form of the following piece of pseudo-code.

```
/*
   The algorithm accepts 12 sets of parameters and temperatures along with a
   state vector. The state vector consists of two 12 bit masks, one keeping
   track of a history state and the other indicating whether the heater should
   be turned on.
*/
unsigned int TCS_calculate(const TCS_parameters   prms[NUMBER_OF_HEAT_PIPES],
                           const TCS_temperatures tmps[NUMBER_OF_HEAT_PIPES],
                           unsigned int           oldState)
{
    /*
     | Start the new state with the old state's history. All heaters will be
     | set to the off state unless otherwise determined by this algorithm
    */
    unsigned int newState = oldState & HISTORY_MASK;
    for (heater = 0; heater < NUMBER_OF_HEAT_PIPES; heater++)
    {
        unsigned int heater_on  = (    0x1 << heater);
        unsigned int history_on = (0x10000 << heater);
        /*
         | Get the parameters and temperatures, then form the delta temperature
         | between the RIT and the reservoir for this heater/sensor pair
        */
        int ritLo      = prms[heater].ritLo;
        int ritHi      = prms[heater].ritHi;
        int resLo      = prms[heater].resLo;
        int delta      = prms[heater].delta;
        int ritTmp     = tmps[heater].rit;
        int resTmp     = tmps[heater].reservoir;
        int dTmp       = rit - reservoir;

        /*
         | If the reservoir is too cold, approaching the temperature that
         | ammonia freezes, request that heater be turned on.
        */
        if (resTmp < resLo)  newState |= heater_on;

        /*
         | If the RIT is too cold and outside the delta limit, request that the
         | heater by turned on. Change the history state to ON.
         */
        else if ( (ritTmp  < ritLo)  && (dTmp > delta) )
        {
            newState |= heater_on;
            newState |= history_on;
        }

        /* If the RIT is below the high limit and it's history was on… */
        else if ( (ritTmp  < ritHi) && (state & history_on) )
        {
             /* …then turn the heater on if the dTmp is large enough */
             if (dTmp > delta) newState |= heater_on;
        }

        /*
         |  If the RIT temperature is not above the high limit and the
```

```
            |  delta temperture between the RIT and the reservoir is large
            |  enough, mark the history state as off.
            */
           else if ( (ritTmp < ritHi) && (dTmp >= delta) ) newState &= history_on;


           /* Mark the history state as off if the RIT is above the high limit  */
           else if (rit  > ritHi)                          newState &= ~history_on;


           return newState;
        }
```

Figure 1 Algorithm example

4 Monitoring

A telemetry packet shall be composed for the purpose of monitoring the inputs and the actions of the TCS. This packet will be logged to the SSR. The format of the data will be a standard CCSDS header followed by the data. The data shall consist of

Offset	Size	What
0x00	3 bytes	Active Sensor mask (24 bits)
0c03	3 bytes	Active Heater mask (24 bits)
0x06	3 bytes	Input status vector, 2 bits for each heat pipe 0 = OK, 1 = 1 Primary Sensor Bad, 2 = Redundant Sensor Bad, 3 = Both Bad
0x09	2 bytes	12 bits of state information
0x0b	48 bytes	24 16 bit temperature values representing the input temperature (Heat Pipe 0, reservoir temperature, RIT temperature; Heat Pipe 1, reservoir temperature, RIT temperature). Values > 128 C and less than –128 C are pinned to their respective limits.
0x3b	3 bytes	Algorithm result bit vector (24 bits, 12 state and 12 heater activate bits)

Table 7 Monitoring Record Format

The maximum size of this packet is 62 bytes + the size of the header. If this is logged 1/sec, the impact of downlink bandwidth is small, but not negligible (~.5Kbps/300K, but the real figure of merit is .5Kbps out of the bandwidth devoted to non-science data, about 30Kbps). If necessary, this data can be compressed.