AAU-Cubesat Ground Station Software Overview

03gr831

May 31, 2003

Contents

Chapter 1

General System Overview

This worksheets provides documentation for the software that is written for the groundstation that supports the AAU-Cubesat satellite. The following chapters will each describe one of the software entities that make up the complete solution, whereas this chapter will provide a general overview of the system, both from a hardware and software perspective.

1.1 Deployment Diagram

The physical configuration of the groundstation can be seen on figure 1.1, which depicts all the single hardware elements of the system.

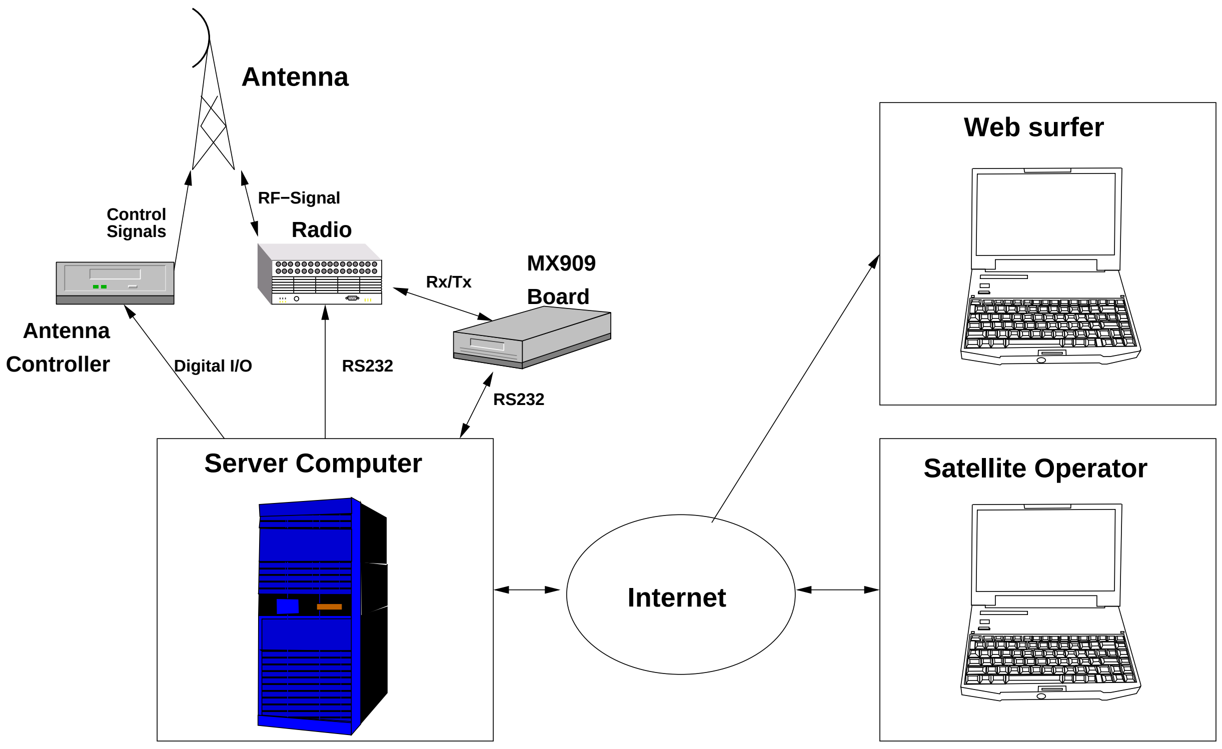

Figure 1.1: Physical diagram of the ground station configuration

The following will shortly summarize the task of each entity on the figure as well as the type of hardware used:

Antenna Controller: The YAESU G-5500 antenna controller amplifies control signals from the computer and applies the control to the antenna rotor. Further it measures the current angles (azimuth and elevation) of the antennas using potentiometers. These signals are feed back to the server computer.

MX909 board: Is a MCB-167 evaluation board that communicates with the server using a serial connection and implements the protocol needed to drive the MX909 modem that is attached to the evaluation board.

Radio: The radio is the ICOM 910H that is used to modulate/demodulate the baseband signal onto the carrier frequency. This frequency is controlled by the server in order to account for the doppler effect and the baseband signal is sent to or received from the modem board

Server: The server runs on a standard PC and maintain the whole system, i.e. it calculates control signals to the antenna controller and doppler shift for the radio. Further it implements the AX25 protocol and decodes incoming information and stores it in a central database. Finally it handles communication with the satellite operators.

Operator: Operates the satellite from a computer that need not to be the server, but could be any computer attached to the internet. From a client program it is possible to control the server operations and send commands/data directly to the satellite if it is in range.

Surfer: Is a public user that is able to read statistics about the satellite from a computer anywhere on the internet. These informations are taken from the database and presented to the user through a web-page programmed in the PHP web scripting language.

1.2 Software Diagram

The diagram depicted in figure 1.2 shows the various pieces of software that make up the groundstation. Ovals are processes and square boxes are terminators and/or sources of data.

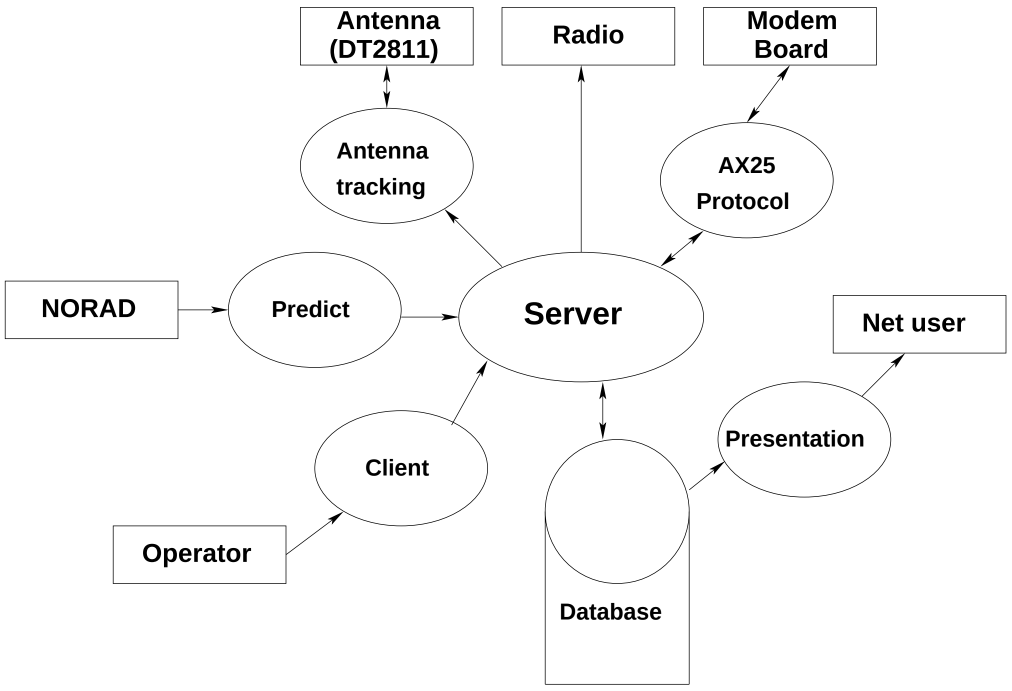

Figure 1.2: Diagram of software implementing the ground station

Again we will briefly summarize the functionality of the blocks focusing on the processes:

Antenna tracking: Receives an azimuth and elevation reference from the server and controls the antenna rotor until the desired orientation has been reached

AX25 protocol: Implements the AX25 protocol. Handles incoming data from the server and the modem and passes it through the protocol.

Database: The database stores and indexes all data acquired from the satellite and makes it available for queries from the presentation program.

Presentation: Is the web-server that handles requests over HTTP from a net user. Takes data from the database.

Client: Is the user control program that is used by the operator to give commands to both the groundstation server and the satellite directly

Predict: Is a piece of software that among other things predicts the position, angles to and doppler shift of the satellite. These data are passed to the server. The predict is updates with Kepler elements originating from NORAD.

The following chapters will describe in more details each of the software entities that make up the software of the ground station. This includes software written entirely by the group as well as pieces of third party software that has been adopted to use with the groundstation.

Chapter 2

Ground Station Server

The ground station server software is the central piece of software in the ground station software architecture. It is the server which collects information from the other processes and decides what should be done and when. The server software is running on a Debian Linux platform.

The server process operates as a state machine with eight different states, as depicted on figure 2.1. State transition is based on satellite orbital position and previous states. When the satellite enters the radio window the server transitions through a number of predefined states hereafter control is transferred to the satellite operator situated at a terminal connected to the server. The following will describe each of the states shortly.

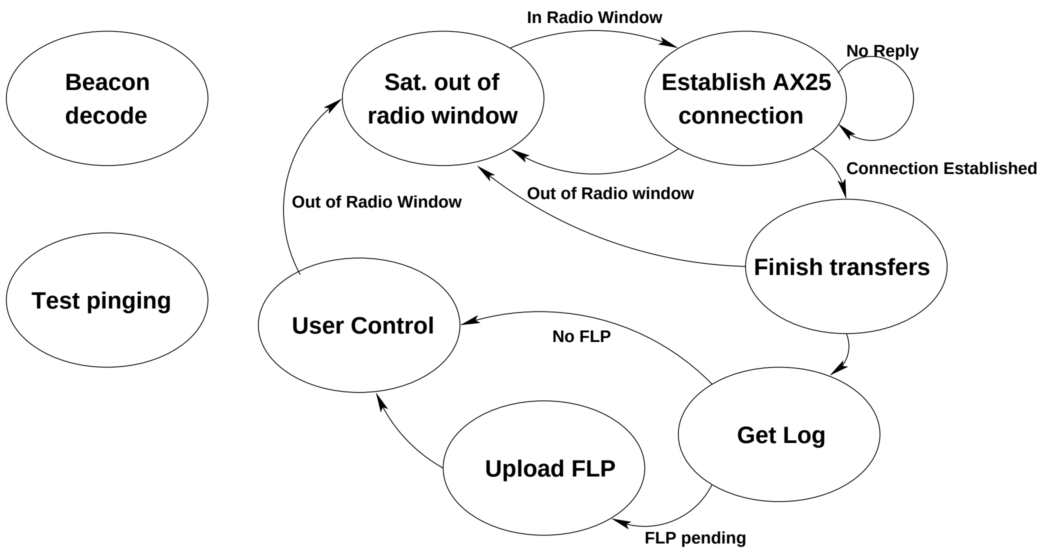

Figure 2.1: State diagram of the ground station server software

Sat. out of radio window: Here the satellite is below the horizon and communication is not possible. The server mainly sleeps, but wakes up occasionally to perform inter process communication and see if satellite has risen above the horizon.

Establish AX25 Connection: The satellite is now predicted to be above the horizon and the server tries to establish an AX25 connection each 5 seconds until success.

Finish Transfers: Before operating the satellite current AX25 transfers must finish. E.g. if on last contact the satellite started to download the log, but did not finish before contact was lost.

Get Log: At the first opportunity the server will request the satellite log to be downloaded in order to get all housekeeping information from the satellite to the database. If there are problems with the satellite the operator can tell the server prehand to skip this state in order to go directly to the user control state.

Upload FLP: If a flightplan is awaiting upload then this will happen in this state, else control will be transferred to the user.

User Control: Here the user is able to control the satellite directly through a client program running on a terminal in connection to the server.

Beacon Decode: The server can be forced in this mode from the client program in order to listen for the advanced beacon signal from the satellite. To leave this state again the client program must tell the server to resume normal operation.

Test Pinging: The server can be forced in this mode from the client program in order to continuesly transmit AX25 test signals each 5 seconds. This mode can be used to actively search for the satellite and radio equipment adjustment (orbit determination). To leave this state again the client program must tell the server to resume normal operation.

It should be noted about the figure that the states ”Upload FLP” and ”Get Log” also transitions to the ”Sat. out of radio window” state if satellite falls below the horizon, however the arrows have been left out on the figure for readability.

2.1 Interfaces

The following paragraphs will elaborate of the various interfaces that exists between the server and other software entities. In general the following text refers to the interfaces shown on figure 1.2.

2.1.1 PREDICT

The PREDICT software run as a server that can be connected through a socket. With the softwarepackage there are examples of how this is done and basically such an example application has been adopted for use in the server software. This allows for very easy communication with the PREDICT server. The following functions is used:

char *send_command(char* host, char* command)

The command takes as arguments the hostname (`host`) where the server is running and a command string (`command`). And the function replies with a string containing the servers response to the command as an ascii string. E.g. using `"GET_SAT ORSTED"` will reply with a string of parameters for the rsted satelitte, this string is then parsed for the needed information.

2.1.2 Client

Whenever the server software receives a header:

```
struct Header
{
  unsigned long Length;
  char CommandType;
  unsigned short crc;
}typedef Header;
```

Then the Command type is examined and if it is **GsServerControl** then the function `ServerControl()` is invoked. This function will analyze the header and perform one of the following functions:

KeepMode Will stop server state transition, and thus force the server to remain in the current mode of operation.

ReleaseMode Will allow the server to follow normal state transition rules again after a KeepMode command.

ForceMode Forces the server to go to the specified mode

SkipLog When this command is issued the server will not request the log as the first action when in contact with the satellite

SkipFLP When this command is issued the server will not upload a new fligthplan to the satellite on the next contact.

2.1.3 Antenna Tracking

The antenna tracking software is implemented as an independent thread, and in order to pass reference variables to it (elevation and azimuth), they must be passed to it through a character device used for inter process communication. The character device is /dev/dt2811 and the structure to pass through the device is defined as follows:

```
struct control
{
  int elevation;
  int azimut;
}
```

2.1.4 Radio

The ICOM-910H radio is connected to the groundstation PC through a serial port (/dev/ttyS0) and a voltage level transformer. As implemented in the server the communication is uni-directional, i.e. the PC sends commands to the radio, but never listens for data or acknowledge signals from the radio. The frame format for the radio communication looks as follows:

FE	FE	60	E0	Command	Data	FD

The first four bytes are header bytes including addressing, which is trivial in our case with only two devices connected, the Command field specifies the command for the radio, the data field is data associated with the command and finally FD indicates end of frame.

When we want to reprogram the frequency to account for doppler shift then the command is 05 and the data field is 3 bytes of 6 binary coded digits (BCD) that describes the frequency with least significant digit first.

2.1.5 AX25 Protocol Process

The AX25 protocol is implemented as a separate process. This process is accessed using socket, where commands and data are sent in the following way:

First a command of one byte is sent (see figure 2.1). This command instructs the AX25 Process to:

1B	4	length
Command	Length	Data

Table 2.1: Protocol for communication between the server and the AX25 process.

Transmit: Send data to peer

Connect: Connect to peer

Disconnect: Disconnect from peer

Ping_Request: Send Ping to peer to check if it is alive

Get_Status: Get status of the AX25 process

After the command an integer (of four bytes) is sent, to indicate how much data is sent to the AX25 layer. Then, as the last part, the data is sent through the socket (size of data equals the length field).

When receiving from AX25, the same protocol is used, but instead of the server sending commands & data data to the socked owned by the AX25 process, the AX25 process sends commands commands to a socket owned by the server.

Commands send to the server are defined as follows:

CONNECTED: The AX25 process is now connected to the satellite. We can start transmitting data/commands.

DISCONNECTED: The AX25 process is now disconnected from the satellite. Data sent to the AX25 process (with the Transmit command) will be queued and sent when connected.

DATA_Indication: The AX25 process has forwarded connection oriented data from the satellite to the server.

UNIT_DATA_Indication: The AX25 process has forwarded picture data (connection less data) from the satellite.

PING_Reply: The satellite has responded to our ping request (it is alive and in range).

When the AX25 process has sent a packet to the server, the server immediately receives it, and acts upon it according to the command. If the command is DATA_Indication, it inserts it into into a linked list. This linked list is read by the main process of the server (using the command **data = rx_AX25()**), which acts upon as described in section 2.2.1.

When the server needs to send information to the AX25 process, it uses the command **tx_AX25(Command, data)**, where ”Command is the command to the AX25 process and data is a pointer to data for the command. When transmitting (using the Transmit command) data must have the type **Header**.

2.1.6 Database

Information is inserted to the database by calling the following helper function.

```
int dbinsert(int dbtable, Log *structure)
```

The dbtable parameter specifies the kind of data, e.g. PCU_HK, and the structure parameter is a pointer to the data itself.

2.2 Functionality

The following subsections will describe the main functions of the server software.

2.2.1 Datasplitter

Whenever data is incoming from the satellite then the server receives **Header** structures from the AX25 protocol process, see figure 1.2. These structures need to be analyzed and stored in the database. This is handled by a function within the server called the **datasplitter()**. To see how it works, observe figure 2.2. At all the terminators of the figure data is stored in the database.

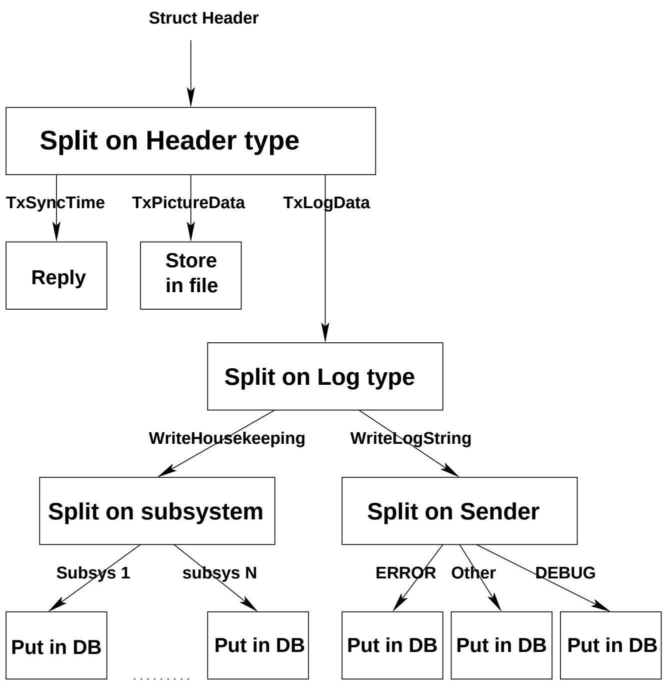

Figure 2.2: Functional diagram of the datasplitter function

At the top the incoming **Header->Type** field is investigated. If it is a `TxSyncTime` request, ie. part of the satellite time synchronization algorithm, a reply will be generated and sent to the satellite and the event will be logged in the database. If the Type field is a `TxPictureData` the incoming datablock will be stored in a filebuffer and when all datablocks has been received, or by command from the client program, the file will be stored in the database as a Binary Large OBject (BLOB).

If the Type field is `TxLogData` then **(Log*)Header->data** will be analyzed and the data will be either Housekeeping information (`WriteHouseKeeping`) or data for the log(`WriteLogString`). If the data is housekeeping information it will be split according to subsystem and for each subsystem decoded according to the specific housekeeping datastructure and stored in the database.

If the data is log data the sender field will be analyzed and the data will be put in one of three logs in the database according to the sender. The possibilities are: ERROR i.e. a log of all errors, DEBUG i.e. a log of all debugging information sent from the satellite and ACTIVITY, ”other” on figure 2.2 which is the activity log of the satellite for normal operation.

2.2.2 Antenna and Radio Control

Periodically the **Antenna_Radio_Control()** function is called from the main loop. The functionality of this function is to contact the PREDICT server and get updates values for: satellite azimuth, satellite elevation and doppler shift. These data is acquired by connecting to the socket interface of the predict server.

The function maintains the global boolean variable **in_range** which indicates if the satellite is in the radio window or not. When this variable is true then azimuth and elevation angles are sent to the antenna rotor controller through the /dev/dt2811 block device:

```
if (in_range==TRUE)
   {
     con.azimut=(int)(az+0.5);    //+0.5 is for rounding off
     con.elevation=(int)(el+0.5); // to nearest integer

     write(dt2811_fd,&con,sizeof(control));
   }
```

Further the radio operating frequency calculated from the base frequency and the doppler-shift are sent to the radio using a serial port of the computer.

2.2.3 Telecommand Forwarding

Commands to the satellite is received by the server from the client program through a socket interface. Upon reception the `Header->Command` field to check whether it is a command for the satellite or the server. If the field is `GsServerControl` the header is handled by the **ServerControl()**-function.

If the header is for the satellite then it is forwarded to the AX25 protocol for uplink using the **Tx_AX25()** function. However if the command field is `RxSyncTime` then the server time is stored in a variable, since this time must be used to calculate a time difference when a reply is received from the satellite. Also if the command is `TxPictureData` then the requested picture block number is saved in a variable, such that it is clear for the server what picture block it receives when unnumbered information frames begins to stream in from the AX25 layer.

Chapter 3

AX25 Protocol Process and Modem Driver

3.1 AX25 Protocol Stack

The communication protocol between the ground station and the satellite is chosen to be AX25, which is an amateur radio protocol. The protocol itself is implemented similar to the implementation on the satellite, which is described in the worksheet ”OBC Software”. Only the interfaces between the layers, upwards to the server and downwards to the mx909 are different, because the software is implemented on another physical platform.

The mx909 (Mobitex) implementation is done on an evaluation board connected to the PC running the AX25 protocol through a serial line. Therefore the interface is defined for a connection through a serial connection.

The interface between the layers inside the AX25 process is implemented using shared memory among the threads doing the actual work of the process. When sending information from one layer to another, it allocates memory for the information sent, it then inserts the information into a linked list and signals the other thread by means of a semaphore. When the other thread receives the signal, it takes the information from the linked list and acts upon it.

Sockets are used as the interface to the server program. The interface is described in section 2.1.5.

3.2 MX909 Modem Driver

The modem driver for the MCB-167 evaluation board employs the exact same software as on the satellite, with the only difference being that the RTX166 operating systems is not used. Therefore the calls from the code to the operating system has been replaced with code that fulfills a similar functionality.

Chapter 4

Antenna Tracking Program

In order to communicate with the satellite, the antenna of the ground station must point towards the satellite when it is in range. This is done by means of a driver, which makes the antenna follow a reference given by the server program. References for the antenna are given as shown below,

```
  struct ref_t  int elevation; int azimuth;  ref_t;
```

where elevation is the vertical angle of the satellite, given in the range zero to 180 degrees, with zero being horizontal pointing forwards, and 90 being zenith. Azimuth is the horizontal orientation, with angles in the range from zero to 450 degrees, with 45 or 405 degrees being North and 135 degrees being East.

The antenna tracking program is implemented as a character device (/dev/dt2811), with references being written to the device and current angles read from the device. Status and current angles can also be accessed, reading /proc/dt2811. With the program being implemented as a character device, it must be inserted a a module before it can be accessed. Thereafter it can be accessed as a normal character device with the following C functions open(), read(), write() and close(). The controller algorithm of the antenna tracking program uses a hysteresis of one degree in horizontal movement and a half degree in vertical movement to make it more gentle (and make the motors last longer). The precision of the the antenna tracking is about one degree.

It should be mentioned, that if the motor in one of the directions does not move for a few samples (when instructed to). An error is reported and shown in /proc/dt2811, and when reading /dev/dt2811, a negative angle will be read in the current orientation. Moreover the motor can not be instructed to move in the current orientation until the module is reinserted.

Chapter 5

Client Program for User Control

In order to facilitate user inputs a client program for the server has been written which is used to generate telecommand packets for the satellite (see figure 1.2). The following will document this piece of software.

The program offers the user three general possibilities:

- Send control commands to the server
- Generate and upload fligthplans
- Send direct commands to the satellite

5.1 Server Commands

The following list of commands can be sent from the client to the server in order to control the server operations:

- Keep Mode
- Release Mode
- Force Mode
- Skip Log
- Skip FLP

The Keep Mode and Release Mode is used to ”capture” the server in a specific mode, such that normal state transition does not take place. When a Keep Mode is issued the server remain in the current mode, and when the server receives a Release Mode the server is again free to follow normal state transition of modes. For explanation of modes see figure 2.1.

The Force Mode commands puts the server in a specific mode where from mode transitions follow normal rules unless Keep Mode has been issued in which case the server will remain in the forced mode.

Skip Log and Skip FLP each makes the server bypass the corresponding state in figure 2.1 on the next satellite fly-over. This allows the operator to skip these modes in order to maximize time with user control, e.g. if there are problems with the satellite (Theoretical possibility).

5.2 Flightplan Editor

The flightplan editor allows the user to:

- Add/Delete/Edit entries in a flightplan
- Save and load fligthplan on/from disk
- Upload flightplans

It is possible to put the following types of commands into a flightplan:

1. Get Status
2. Take picture
3. Set Acsmode
4. Execute command

Chapter 6

Database Design

The main purpose of the database is to make it possible to log the data to and from the satellite in a functional way. It must be possible to extract any transaction that have been made at any given time. That is:

- It must be possible to extract any command sent to the satellite
- It must be possible to extract any data sent to the satellite
- It must be possible to extract any information retrieved from the satellite

The overall structure is shown in figure 6.1

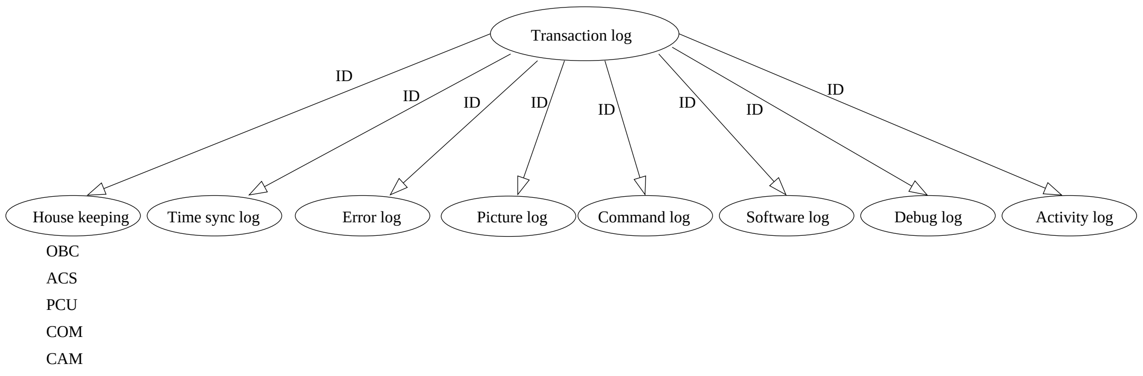

Figure 6.1: The overall structure of the database.

When communication between the ground station and the satellite is established the direction (if it is to or from the satellite) is logged in the transaction log. Transaction log contains the direction of the data, timestamp and the ID of the communication between the ground station and the satellite. This ID is unique and passed on to the appropriate log. This means that every table in the database contains a column with the ID of the transaction log.

The House keeping contrains house keeping information from each subsystem. That is OBC, ACS, PSU, COM, and CAM. Time sync log contains data from when the Ground Station and the satellite synchronize time. Error log contains a log of any errors which has occurred on the satellite. Pictrue log is a list of all the pictures retrieved from the satellite. Command log contains every command sent to the satellite, but not the software itself (if new software were to be uploaded) as this is placed in the software log. The Debug log contains debugging information from the satellites subsystems and finely the activity log contains normal activity on the satellite.

An example of how the communication works could be that a logfile is requested from the satellite. The transaction is logged in transaction log and assigned an ID as following:

Inserted into transaction log:
Direction=”up”, timestamp

The ID is returned from the query.

Inserted into command log:
Id=”ID”, command=”request logfile”

The logfile is then received from the satellite and this transaction is also put into the log.

Inserted into transaction log:
Direction=”Down”, timestamp

The ID is returned from the query and the logfile-information is split up into the appropriate logs. In this example the log only contains one entry (witch is highly unlikely).

Inserted into activity log:
Id=”ID”, string=”error string”

In this way the user is able to locate any information sent or received at any given time to the satellite using the transaction log and then finding the table witch contains the transaction ID. In the same way a time for a given task can be found by (if we wish to find the data at witch a logfile was retrieved) looking in the table command log and finding any incident where the line “request logfile” exist and then look in the transaction log and match the ID and Direction=”up” to the timestamp.

Chapter 7

Data Presentation Software

The software to present the downlinked data to operators and users have not been developed at this point, since it has rather low priority compared to the more important software and hardware systems of the satellite. Development of the presentation program is currently scheduled to commence end January or in the start of February.