Telemetry / Telecommand Frontend - IITMSAT

A Project Report

submitted by

ANOOP R SANTHOSH

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2014

THESIS CERTIFICATE

This is to certify that the thesis entitled **Telemetry / Telecommand Frontend - IITMSAT**, submitted by **Anoop R Santhosh**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Shankar. Balachandran
Project Guide
Assistant Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor Dr. Shankar Balachandran for his inspiration, support and guidance throughout this project and entire 4 years in general. He is a great teacher and a tremendous mentor.

I would also like to thank Dr. Harishankar Ramachandran, for the time he devoted towards discussing the design and motivating me in general.

My sincere thanks also goes to Dr. David R. Koilpillai, for his invaluable feedback and the inspiring talks during every IITMSAT session.

I would like to extend my gratitude to Mr.Akshay Gulati, the person who motivated me to join IITMSAT team . He has been a wonderful mentor , devoting a lot of his time discussing the design and implementaion.

I am grateful to the entire Computer Science Department ,IITMSAT team and my batchmates for their support.

Last, but not least , I am grateful to my parents and friends for their constant support and motivation.

ABSTRACT

KEYWORDS: TMTC FrontEnd, AX.25, IITMSAT

IITMSAT is a low orbit nano satellite being developed by students at IIT Madras. The satellite project has various subsystems, both hardware and software . Ground station software is an important part of it. TMTC Frontend is an important sub module within the ground station software. It essentially functions as the link layer and implements the link layer protocol which in this case is AX.25 .

The main functionality of this module includes AX.25 protocol encoding/decoding , flow control including acknowledgements and counters and packet reassembly. The overall design is loosely inspired by the SwissCube ground station design. But to satisfy IITMSAT mission requirements which are different from SwissCube, many modifications were made, especially in the telecommand transmitter part. The link layer protocol though based on standard AX.25 protocol, has been modified to meet our requirements.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CCSDS	Consultative Committee for Space Data System
CRC	Cyclic Redundancy Check
ECSS	European Cooperation for Space Standardization
GS	Ground Station
ISRO	Indian Space Research Organisation
MCS	Mission Control System
MDR	Mission Data Repository
MIB	Mission Information Base
SDR	Software Designed Radio
SSID	Secondary Station Identifier
TC	Telecommand
TM	Telemetry
TMTC	Telemetry Telecommand
VC	Virtual Channel

CHAPTER 1

INTRODUCTION

This chapter provides a brief introduction to IITMSAT in general and the particular problem statement i.e, TMTC Frontend .

1.1 Overview and Problem Statement

IITMSAT is a student initiated low orbit satellite project being developed at IIT Madras. A high energy particle detector is attached to the satellite and it can be used to measure changes in flux of charged particles in ionosphere. The project aims to study the correlation between earth based phenomenon and the changes in charged particle flux. There are various subsystems involved in this project , both hardware and software . A detailed discussion of all the subsystems is beyond the scope of this document , though a few relevant parts will be briefly discussed here.

1.1.1 Overview of IITMSAT

The main configurations/features of the satellite are as given below:

- **Orbit :** Low earth Orbit
- **Altitude :** 600 km to 800 km
- **Inclination :** >45 degrees
- **Mission Life Time :** >1 year

- **Dimensions :** 29 cm x 29 cm x 26 cm
- **Mass :** 15 kg
- **Payload :** High Energy Particle Detector

1.1.2 Ground Station Software

Ground station software is responsible for handling the operations of the satellite from ground and process the information transmitted by the satellite on downlink.It provides an interface for interacting with satellite and analysing data besides keeping track of various parameters of the satellite.The major modules within the ground station software are :

- **User Interface :** Lets the user select commands to control the satellite and displays various data collected by the satellite after processing and analysing it in appropriate ways.
- **Mission Control System** : MCS is responsible for converting the user input to CCSDS telecommands before dispatching them to the satellite . It is also responsible for archiving all packets sent and received. It is also the unit which schedules commands and processes the data received from satellite (CCSDS packets) on downlink. It interacts with UIs on one end and TMTC Frontend on the other. It transmits and receives CCSDS packets to/from TMTC Frontend.
- **TMTC Frontend :** This layer essentially performs like a link layer. It is discussed in detail below.

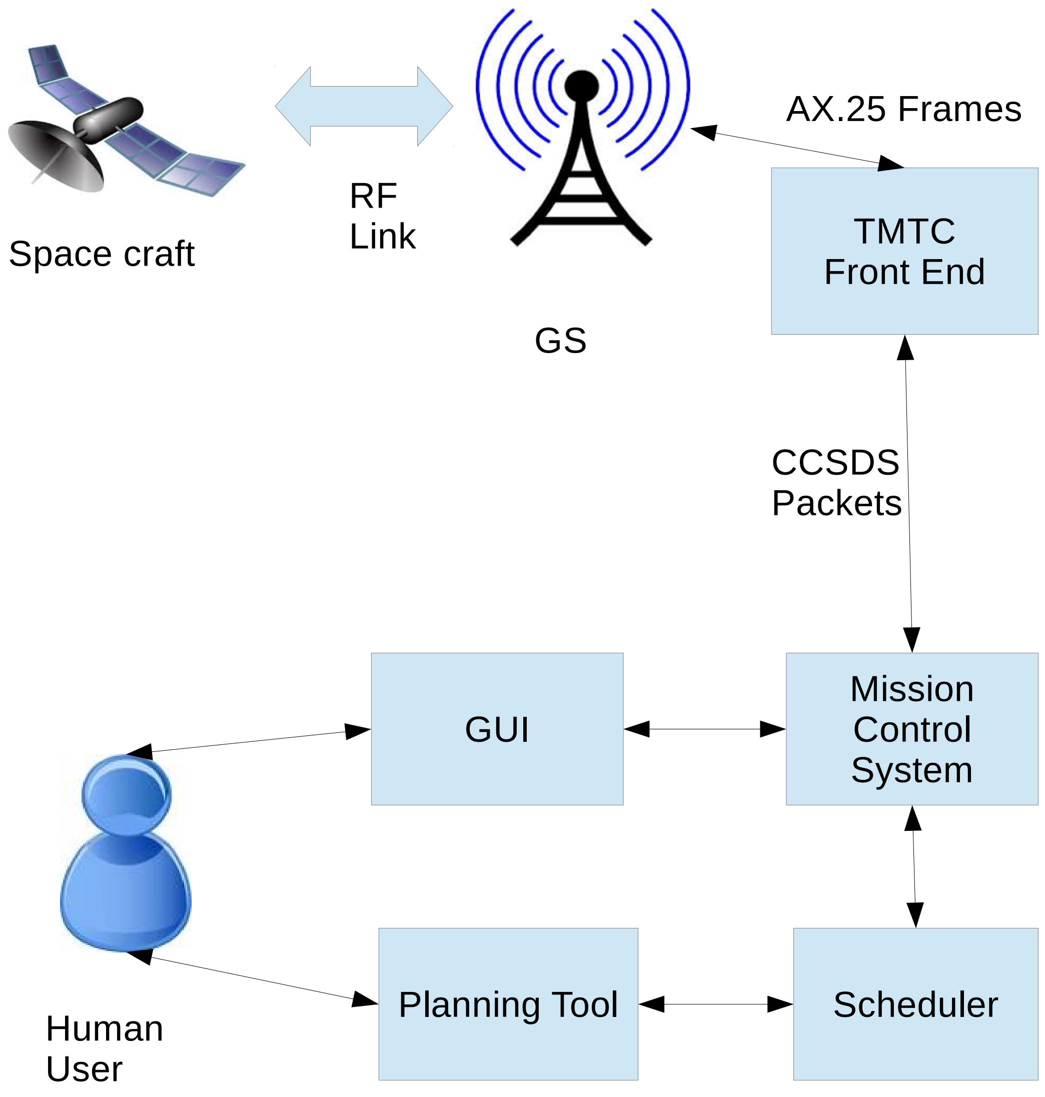


Figure 1.1: Ground Station Block Diagram

1.1.3 TMTC Frontend - The problem statement

TMTC Frontend essentially acts as a link layer between MCS and satellite. The main aim of this project is to implement a robust TMTC Frontend. TMTC Frontend is responsible for encoding of CCSDS packets received from MCS and decoding of AX.25 frames received from GS. It is also responsible for handling telecommand transmission and telemetry reception. It takes care of acknowledgments, packet resending etc.

1.2 Motivation

A robust link layer is very essential for proper communication with the satellite. This layer is responsible for encoding application data (CCSDS packets) to AX.25 protocol on uplink and vice versa on downlink.The communication link between earth based GS and satellite may not be very reliable. Hence we need a system which can take care of acknowledgements and resending of dropped packets without involving the application layer. This is important because the amount of time a communication link is open with the satellite in a day is very limited. So it is essential to handle packet resending ,acknowledgements and reassembly of received packets at this layer, rather than involving MCS (the application layer) which can slow down operations and affect the amount of data transferred. All data / command transfer between MCS and satellite happens through this layer.

1.3 Contributions of this work

Though the design is loosely based on GS software design of SwissCube, there are quite a few differences. The transmission of data was not a big concern in case of SwissCube, which is not the case with IITMSAT. Hence in SwissCube GS software, resending dropped packets is handled at MCS through manual control. However, in case of IITMSAT, we expect to transmit more commands/ data in uplink and cannot depend on manual control for packet retransmission. So our design involves automated retransmission at link layer itself, instead of involving application layer.

Another major difference from SwissCube is that we have modified the AX.25 frame to support 256 outstanding telecommand packets at a time where as SwissCube design allows only for 4. Owing to the manual control over packet transmission , 4 outstanding packets were sufficient for SwissCube. But this is not the case with IITMSAT.

1.4 Organisation of report

This report is organised into 6 chapters. Chapter 2 explains a few terminology involved and frame structure of the protocols used. Chapter 3 gives a detailed explanation of the design of TMTC Frontend. Chapter 4 discusses implementation details . Chapter 5 explains how the software system was tested. Chapter 6 is the Conclusion chapter .

CHAPTER 2

BACKGROUND AND PROTOCOLS

This chapter gives a brief background on IITMSAT ground station software, detailed specifications of TMTC Frontend and details of the protocol used.

2.1 Ground Station Software

As briefly mentioned in the introduction, ground station software is responsible for providing an interface to interact with the satellite and control the operations of the satellite from ground. The main user requirements are :

- Control the satellite and monitor its operations.
- Process the downlink data and provide payload data in human interpretable format.
- Monitor various house keeping parameters ,like voltage and alert users in case of an issue or takes action whenever possible.
- A GUI for easy human interaction.
- Implement application layer protocols in compliance with ECSS standards. CCSDS packets are used for application data encoding.
- Provide a reliable communication channel to the satellite.

Note : Within the CCSDS packet telemetry and telecommand concept , an on-board application process is defined as the source of telemetry source packets and the sink for telecommand packets [3].Throughout this report telecommand refers to commands send to the satellite from GS and telemetry refers to data received by GS from the satellite.

The GUI interacts with MCS. MCS can be considered the application layer. It is responsible for converting user commands into CCSDS packets , by reading commands and bit encoding from Mission Information Base(MIB) .Telecommands are defined in MIB. MCS is also responsible for intelligent scheduling of telecommands.It also archives all telecommands , telemetry and housekeeping data in Mission Data Repository (MDR). It is also responsible for decoding received telemetry(CCSDS packets) based on the encoding defined in MIB. It processes the data and displays it to the user in appropriate format through a GUI. It can be provided with support for processing payload data and represent it in human readable form or can just dump the payload data to some sink , which takes care of it. It also keeps track of housekeeping parameters like voltage , on board time etc and alerts the user in case of an issue or take action when possible. It performs monitoring of the satellite completely.

MCS interacts with users on one side, taking in commands and displaying received data. After processing the data and converting them to CCSDS packets, it forwards those packets to TMTC Frontend. TMTC Frontend as mentioned above forms a link layer and is responsible for encoding packets to AX.25 protocol and transmitting it to the satellite . This is the uplink part. On downlink , the MCS receives telemetery CCSDS packets from TMTC Frontend. MCS then processes it , decodes data and takes appropriate actions or displays it on GUI.

2.2 TMTC Frontend

TMTC Frontend is the layer between MCS and GS hardware. It acts as a link layer for communication between GS and satellite . It is expected to provide the following functionalities :

- **Uplink :**
  - Encoding CCSDS packets (telecommand) to AX.25 frame.
  - Archive AX.25 telecommand frames.
  - Flow Control (resending, counters, acknowledgements etc).
- **Downlink :**
  - Decode AX.25 frame (telemetry) and reassemble the application data to obtain CCSDS packets.
  - Archive raw AX.25 telemetry frames.
- **Replay :**
  - Provide functionality to replay archived raw AX.25 telemetry frames.

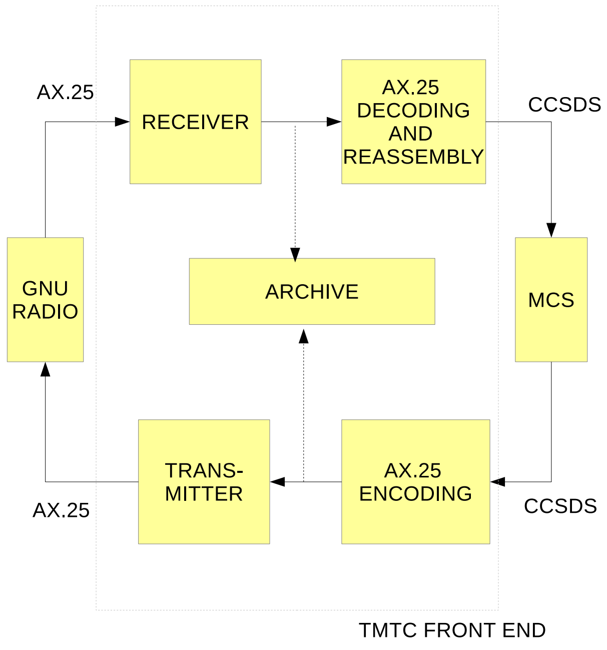


Figure 2.1: TMTC Frontend Block Diagram

TMTC Frontend will be discussed in detail in next chapter.

2.3 Link Budget

Communication link with a satellite may not be extremely reliable. Hence the link should be modelled properly. However such a communication link can be affected by many factors like radiations, ionospheric charged particles etc. For modelling such a communication link, a complex set of parameters need to be considered. Hence rather than creating a model of our own, we have decided to use the SwissCube estimates. Since the altitude and parameters of both satellites are quite similar, this would be a good approximation. The bandwidth according to SwissCube model is 1200 bits/sec and error rate in 7%.

2.4 Data Link Layer Protocol - Modified AX.25 Frame structure

AX.25 is a data link layer protocol derived from X.25 protocol suite and designed for use by amateur radio operators[1]. AX.25 is most frequently used to establish direct point to point links.Since the overall IITMSAT design is based on SwissCube design, we have decided to use AX.25 frames modified to meet our needs. SwissCube uses their own modified version of AX.25.

There are generally three types of AX.25 frames:

- **Information Frame :** Used for data transfer.

- **Supervisory Frame :** Used for supervisory link control.
- **Unnumbered Frame :** Provides additional control over the link.

These frames can be distinguished by the value of control field. The protocol and acknowledgement mechanisms which comes with this protocol does not suit our needs. For example , the protocol allows for only 8 outstanding packets, while our system needs more than 8 out standing packets.

We have decided to use only the frame structure of AX.25 protocol for our system and not the flow control mechanism described by the protocol. We have modified the frame structure to suit our needs by adding extra fields and changing the interpretation of some fields (from the standard protocol).We are just using the frame encoding with our modifications and not any other feature. For the sake of convenience this modified version will still be referred to as AX.25 protocol/frame throughout this document.

The modified frame structure is briefly described below. The header common for both uplink and downlink will be discussed first, followed by modifications specific to each of them.

2.4.1 Modified AX.25 Frame

The frame structure is as shown below :

Flag	AX.25 Transfer Frame Header (128 bits)				Information Field	Frame Check Sequence	Flag
	Dest. Address	Source Address	Control Bits	Protocol Identifier			
8	56	56	8	8	0 – 2048	16	8

Figure 2.2: AX.25 Transfer Frame Format.Adapted from [1]

Few important fields are briefly explained below :

- **Destination and Source Address :** Both have similar structure and length of 56 bits. Destination and Source represent satellite or ground station depending on whether it is uplink or downlink. The address field structure is given below:

Call Sign (48 bits)			SSID 8 bits
C1 8 bits	C2 – C5	C6 8 bits	

Figure 2.3: AX.25 Address Field.Adapted from [1]

  - **CallSign :** 6 upper case letters.They are placed in C1 to C6 fields.
  - **SSID :** 4 bit integer used to identify multiple stations using the same call sign . They are placed in fields 3 to 6, with fixed values for other fields.

- **Control bits :** 8 bits long. Originally used to differentiate between the types of frames mentioned above, we no longer use it in our version. It is assigned a fixed value of 00000011 (0x03). This actually refers to an unnumbered frame.

- **Protocol Identifier :** 8 bits long.Originally used to represent the layer 3 protocol used, we have modified this field to indicate if a frame is an acknowledgement frame or not. Value 00000011(0x03) indicates that it is an acknowledgement frame. Otherwise value is 11110000(0xF0), which as per standard protocol indicates no layer 3 protocol implemented.

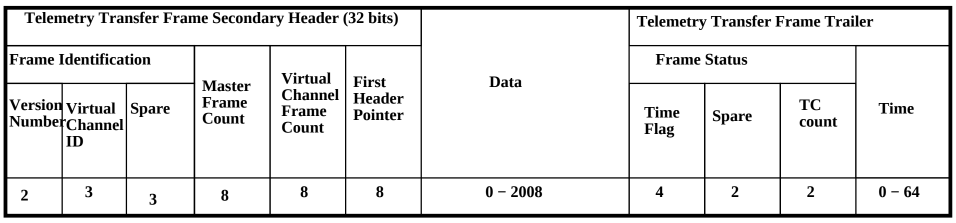
- **Frame Check Sequence :** CRC-CITT is used to calculate a 16 bit CRC. The polynomial used is $x^{16} + x^{12} + x^5 + 1$.

2.4.2 Telecommand Information Field Usage

To number the frames , the first byte of the information field is used as a frame counter. Thus we have a modulo 256 counter for frames. Rest of the information field carries telecommand data , with a maximum size of 2040 bits or 255 bytes.Virtual channels are not supported on uplink.

2.4.3 Telemetry Information Field Usage

Telemetry information field has much more modifications than telecommand. An extra header and trailer are added. It supports virtual channels too. The information field structure is given below:



Telemetry Transfer Frame Secondary Header (32 bits)						Data	Telemetry Transfer Frame Trailer			
Frame Identification			Master Frame Count	Virtual Channel Frame Count	First Header Pointer		Frame Status			Time
Version Number	Virtual Channel ID	Spare					Time Flag	Spare	TC count	
2	3	3	8	8	8	0 – 2008	4	2	2	0 – 64

Figure 2.4: AX.25 Telemetry Header and Trailer

Secondary Header (32 bits)

The secondary header is 32 bits long. It provides support for virtual channels and large data transfer (which will be discussed in detail later). The important fields are :

- **Frame Identification**
  - Version Number (2 bits) : Fixed to zero (00).
  - Virtual Channel ID (3 bits) : Supports upto 8 virtual channels. Eg. 111 - represents data belongs to channel 8.

  - Spare (3 bits) : Reserved for future use. Value set to 000.

- **Master Frame Count (8 bits)** : Total frame counter , ie., irrespective of the virtual channel. (modulo 256 counter).
- **Virtual Channel Frame Count (8 bits)** : Counter for a particular virtual channel (which corresponds to the current channel) . (modulo 256 counter).
- **First Header Pointer (8 bits)** : Specifies the octet number within the data field that contains the first octet of the first packet header.This allows the reassembly of packets even when previous packets were lost. It has value 11111111(0xFF) if no packet header starts in the frame.

Trailer (0 to 72 bits)

- **Frame Status (8 bits)**
  - **Time flag (4 bits) :** Gives size of time field. 000 indicates no time field. If bit 0 is set to 1, bits 1 to 3 indicate the size of time field in octets + 1.
  - **Spare (2 bits) :** Reserved for future application. Set to 00.
  - **TC Counter (2 bits) :** No longer used in the modified version. Set to 00.
- **Time (0 - 64 bits) :** On Board time . Represents the time at which end flag of previous frame was transmitted.

2.4.4 Acknowledgement Frame

As mentioned earlier, a frame is identified as acknowledgement frame by the value 0x03 in protocol identifier field. Individual packet numbers , each a byte long are placed sequentially in the informtaion field. An acknowledgement frame does not contain a counter (for telecommand) or telemetry header .

CHAPTER 3

DESIGN OF TMTC FRONTEND

The TMTC Frontend consist roughly of four major sub modules.

- AX.25 Packet encoding/decoding
- TC Transmitter
- TM Receiver
- Replay Controller

3.1 AX.25 Packet encoding / decoding

This module implements a library for encoding / decoding data to/from our modified AX.25 protocol frame structure. Frames are represented as a byte array. Implementation details will be discussed in next chapter.

3.2 TC Transmitter

The telecommand transmitter is based on a state machine.Initially it was decided to use to the SwissCube transmitter algorithm. However it was quite limited for our requirements.

3.2.1 Limitations of Swiss cube design

- It can support only 4 outstanding packets.
- Packets are transmitted one at a time.
- Transmission of packets are manually controlled by the user through MCS.
- There was no option of resending at TMTC Frontend layer.
- In the event of packet drop, the information is carried back to the application level, where a human user decides to resend the packet again.

3.2.2 Modified Design

The manual control from application layer on transmission will make it really inefficient. No support for automatic resending of packets at TMTC layer also is an issue. This design was fine in case of SwissCube , as they did not expect much data transfer with the satellite. That is not the case with IITMSAT. We expect to have more control over the satellite than SwissCube and hence the above restrictions can be a serious concern.It slows down operations . This is especially important as IITMSAT plans to have only one ground station. This severely restricts the amount of time a communication link can be opened with the satellite as it comes into the field of view for a short time every day.Hence we modified the design to support more outstanding packets and support for resending at TMTC layer. The main characteristics are :

- Packets are transmitted when transmitter is on and positive beacon signal is received.
- All the packets in the ready queue are dispatched at the same time one after another. The number of packets is expected to be in the order of 10, which is way less than the 256 outstanding packets allowed.

- Transmitter is half duplex. So we are not implementing an explicit timeout. Acknowledgement for a frame sent in a transmitter period is expected to come in the immediate next receiver period.
- If acknowledgement is not received in the immediate next reception, packet is resent.A packet will be resent a fixed number of times, after which packet drop will be announced to MCS.

The state diagram is as given below :

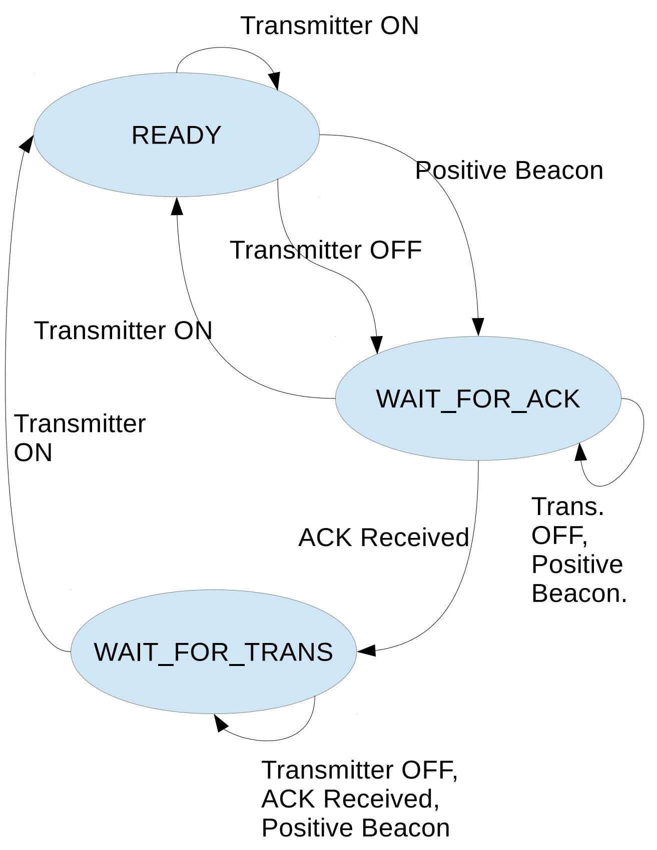


Figure 3.1: TC Transmitter State Diagram

The states are :

- READY : Indicates that transmitter is ready to sent packets and transmitter is on. Positive beacon is the trigger to send frames.
- WAIT_FOR_ACK : Waiting for acknowledgements for outstanding packets.If transmitter is turned on in this state, it adds all outstanding packets to resend queue. If acknowledgement frame is received, state changes to WAIT_FOR_TRANS.
- WAIT_FOR_TRANS : Waiting for transmiter to switch on. When transmitter is on, moves to READY state.

The external triggers are :

- Transmitter ON : Indicates switch to transmission mode.
- Transmitter OFF : Indicates switch to reception mode.
- Positive Beacon : Indicates that satellite is in field of view and ready for reception.
- Ack received : Indicates the reception of acknowledgement packet by receiver.

Besides this, the transmitter archives all AX.25 packets in a database with timestamp.

3.3 Telemetry Receiver

TM Receiver receives the telemetry from GS hardware (or in IITMSAT case , SDR).Before going into details of how receiver processes it , a few important concepts are discussed.

3.3.1 Virtual Channels

There are various subsystems on board the satellite which needs to communicate with GS, like Housekeeping, payload etc. All these subsystems require a communication link with GS, but only one physical data channel exists. The idea of virtual channels alleviate this problem.

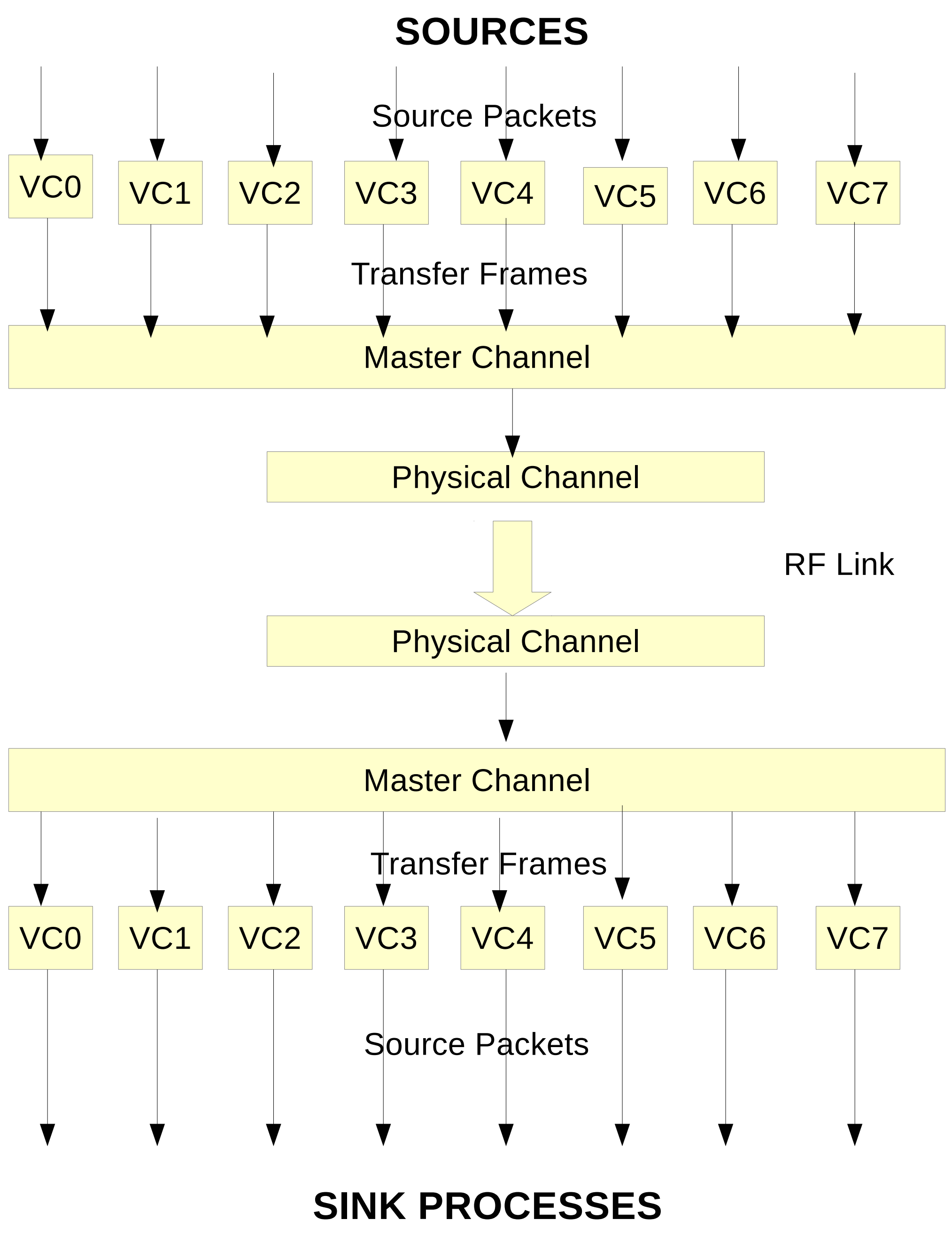


Figure 3.2: Virtual Channel. Adapted from [2]

Each subsystem is assigned a virtual channel to communicate with ground station and these channels are multiplexed together into the same physical channel. Each frame has 3 bits to indicate which virtual channel the data belongs to. Thus it can support up to 8 virtual channels . It gives an indication on how to process/forward the data at GS . Also it gives all subsystems a fair chance of communication with GS. In the absence of virtual channels , a heavy subsystem like payload may occupy the master channel for most part, giving data light subsystems very less chance for communicating with ground station. Virtual channels ensure that all subsystems get a fair chance. Satellite just sends a packet with empty information field in case there is no data to be transmitted on that virtual channel in that particular slot. As mentioned in previous chapter, the telemetry secondary header supports virtual channels.

3.3.2 Large Data Transfer and Reassembly Unit

In case of telecommand ,it is quite possible to have commands of fixed size and it can usually be carried by the information field of a single AX.25 frame. This is not the case with telemetry. Though we might be able to fix the length of standard services, it is not possible in case of payload data. Payload data length can vary over a large range. Splitting payload data at application level is not a good idea as it will lead to cumbersome processing later at MCS. This is the case with large payload packets. On the other side, payload packets could be quite small in size and more than one packet could fit into the information field of one AX.25 frame. Transmitting them separately adds the header overhead to both the packets and it is a waste of resources to do so. It would be better to fit in both of them into same AX.25 frame. Large Data Transfer Services help in solving these issues.The idea was proposed in [3].

In large data transfer service, all the packets to be transmitted are sequentially attached one after another.This creates a single service data unit. This data unit is then split into fixed size (usually the size of information field of link layer protocol) and each such unit is transmitted in each frame.

Our design implements this service with the help of first header pointer in secondary header of AX.25 telemetry. This field indicates the octet in data field where the first byte of the header of first packet resides. Header of subsequent packets can be obtained after finding the length of each packet(We assume CCSDS packets and read the length from the predefined length field). First header pointer helps in recovery of CCSDS packets even when the previous packets are lost.

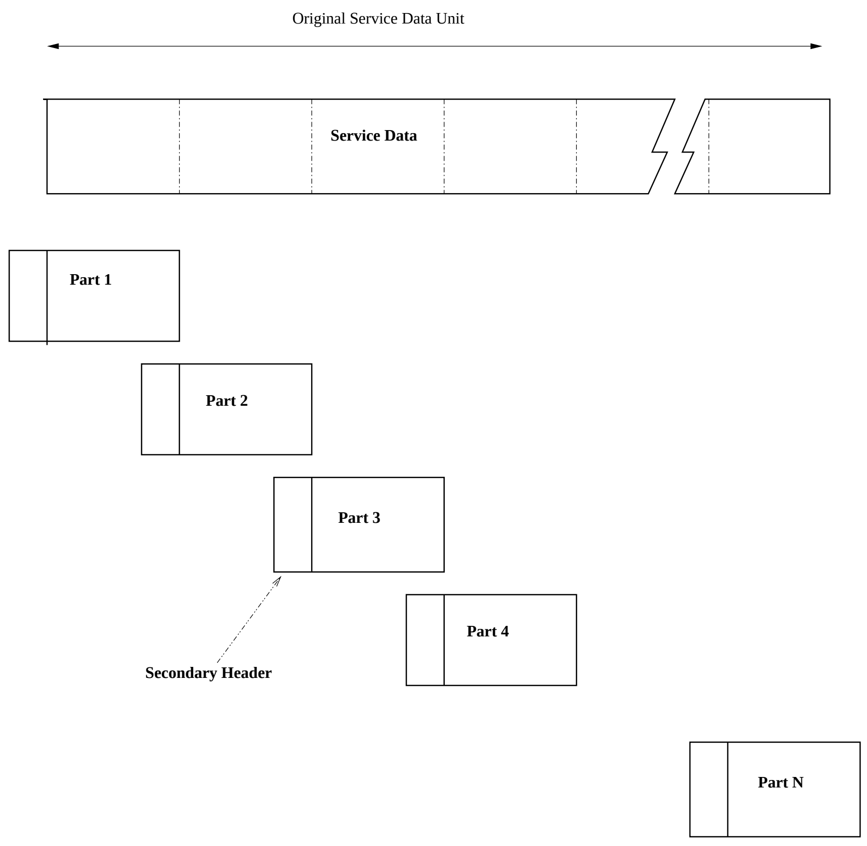


Figure 3.3: Large Data Transfer Service. Adapted from [3]

Reassembly unit is responsible for reassembly of packets at GS. There is a reassembly unit corresponding to each virtual channel. Reassembly unit takes care of packets arriving out of order. It maintains a buffer of all frames received on that channel and fills in data as and when the frames arrive , completing the packets. It creates a new buffer for each fragment of data and fills in the buffer when new frames arrive until the number of bytes specified by the length field is filled.

3.3.3 Overall design of TM Reeciver

Processing steps after reception of an AX.25 telemetry frame in that order is as follows.

1. Receive AX.25 frame.
2. CRC check to detect frame bit errors.
3. Check the GS and Satellite SSID and Callsign.
4. If the frame is an acknowledgement frame, trigger Acknowledgement received of receiver.
5. Otherwise Archive the raw frame in database .
6. Forward the frame to appropriate virtual channel.
7. Transfer the frame to reassembly unit of the virtual channel.
8. After a CCSDS packet is completely reassembled, forward it to the MCS on appropriate port or channel.

3.4 Replay Controller

Replay controller lets the user replay archived AX.25 raw frames. It provides a small GUI where user can enter the virtual channel ID and the time range of in which the packets he/she wants to replay was received by the receiver. Replay controller reads all frames in the range for that particular virtual channel from the database. It then creates a reassembly unit and forwards the frames one by one to the unit. Upon completion packets are forwarded to MCS as usual.

CHAPTER 4

IMPLEMENTATION DETAILS

This chapter discusses the implementation details of TMTC Frontend.

4.1 Development Tools and Environment

- **Development Language :** Java 1.6.0
- **Development Environment :** Eclipse IDE
- **Database server :** MySQL 5.5.35
- **Operating System :** Ubuntu 12.04

4.2 Development phases

The development was carried out in four phases:

1. Implementation of modified AX.25 frame encoding/decoding.
2. Implementation of TC Transmitter.
3. Implementation of TM Receiver (including reassembly unit).
4. Implementation of Replay Controller.

4.3 AX.25 encoding/decoding

This module implements the functionality to encode and decode AX.25 frames. It is implemented in the package **AX25**. This package implements all the features of the modified AX.25 frames which we are using in IITMSAT.The important classes are discussed below.

4.3.1 AX25Frame

This class provides functionalities which are common to both telemetry and telecommand frames. This include :

- Converting the frame to a byte array.
- Generating a frame from a byte array.

4.3.2 AX25Telecommand

This class inherits from AX25Frame. It implements features specific to telecommand, which in IITMSAT case is just the Master Frame Count.

4.3.3 AX25Telemetry

This class inherits from AX25Frame. It implements features specific to telemetry, which are the secondary header and trailer in the information field (Discussed in chapter 2).

4.3.4 AX25AddressField

This class handles the address fields in the AX.25 frame header.

4.3.5 AX25FrameIdentification

This class implements the frame identification field which is a part of telemetry secondary header.

4.3.6 AX25FrameStatus

This class implements the frame status field which is a part of telemetry trailer.

4.4 Telecommand Transmitter

This module implements the TC transmitter logic (state machine discussed in chapter 3). This module is implemented in the package **TCTransmitter**.The state is represented by a Java enum which can take values READY, WAIT_FOR_ACK, WAIT_FOR_TRANS. The logic is implemented in the class TCTransmitter.The important functions are :

1. ProtocolEncoding() : Application data is passed as an argument to this function. This function encodes the data into AX.25 frames including CRC generation.It then archives the frame in the database and adds it to the EncodedPacketQueue, which is the list of packets encoded and ready to be transmitted.
2. DispatchPackets() : This function is called when positive beacon is received on READY state. It first dispatches ack frame, then frames in ReSendQueue and finally frames in EncodedPacketQueue.

3. Various triggers like

    - positiveBeacon() : Called on reception of positive beacon signal
    - receiveAck() : Called on reception of Ack frame by receiver.
    - TransmitterON() : Called when switched to Transmitter mode.
    - TransmitterOFF() : Called when switched to Receiver mode.

Besides this, the class holds a list of outstanding frames and counters indicating the number of times they were transmitted.

4.5 Telemetry Receiver

This module implements the TM receiver logic (discussed in previous chapter). It is implemented in the package **TMReceiver**.There are two important classes:

1. TMReceiver : Entire logic of the telemetry receiver, which includes checking validity (CRC), addresses, archiving, forwarding to reassembly unit etc. are implemented in this class. It has a list of reassembly units , each corresponding to one virtual channel.

2. ReassemblyUnit : It implements the packet reassembly logic. TMReceiver calls the function ReassemblePacket() of this class with a AX25Telemetry frame as an argument. This function buffers information field of the frame until a packet can be created.

4.6 Replay Controller

Replay controller as described earlier provides the functionality of replaying archived telemetry packets. The logic behind replay controller was explained in previous chapter. It is implemented in the package ReplayController. It has a small UI , which lets the user enter VCID, start and end timestamps.

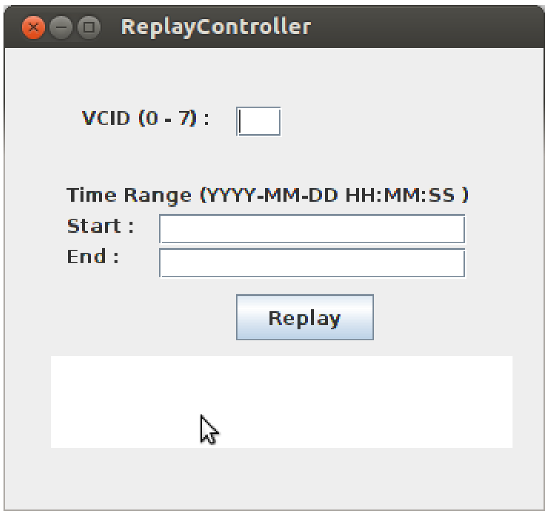


Figure 4.1: Replay Controller GUI

4.7 Archiving Frames

Both telecommand and telemetry frames are archived.It was decided to use SQL as the Database because it supports simultaneous accesses and can handle large amounts of data. Besides that other modules of GS software also use SQL. Both telemetry and telecommand archive tables have three fields. These are TimeStamp, Counter and Frame. Frame field stores the byte array .MySQL is the database used.

CHAPTER 5

TESTING

Testing the TMTC Frontend module completely requires a working MCS and SDR . Unfortunately neither of these are available. Even a robust simulator for both ends are not available as certain details of those two modules are still under review.

So to check the robustness of TMTC Frontend , all units were tested independently. Still the complete system had to be tested. For this, based on the available details about expected working of MCS and SDR,similar situations were simulated. Though the simulation may not reflect the MCS and SDR operations completely, care was taken to make sure that it tests all functionalities expected of TMTC Frontend under closely approximate situations.

5.1 AX.25 encoding/decoding tests

Application data was encoded into AX.25 frame. The corresponding byte array was then decoded and all values/fields were found to be correct. This was done for both AX.25 telemetry and telecommand.

5.2 Reassembly Unit tests

Various situations like packets spread over frames, more than one packet in a frame, out of order arrival etc. were tested.

5.3 Replay Controller

Entries were made in the database (AX.25 Telemetry Archive). The corresponding timestamps and vcid were entered into the GUI and packets were observed to be getting reassembled.

5.4 Receiver and Transmitter

Single packet tests were ran on both receiver and transmitter. Other than that, MCS and GS operations were simulated for different situations like frame drop , resend , different number of frames , different time period etc and tested on TMTC Frontend.Two small dialog boxes were created which would display transmitted and received packets , to observe the testing process.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Current State

With the available requirements and specifications, TMTC Frontend for IITMSAT was developed in Java. All specifications up to now are implemented. Transmitter was developed according to the current expected scenario. Similar is the case with receiver. Replay Controller functionality was also implemented. A small GUI was also created for the replay controller. With the available details and specifications, the system was tested and was found to be working.

6.2 Future tasks

- Run more tests by creating MCS and GS simulators as and when more specifications about their working are available.
- Discuss and incorporate changes suggested by ISRO panel during design review .
- Decide on the internal time representation.
- Implement time correlation report.
- Integrating with MCS and GNU Radio (GS) once they are completed .
- Testing the complete GS software system after integration. This includes load testing over a period (for example : one week).

APPENDIX A

ASSUMPTIONS

Various assumptions made during the development of TMTC Frontend are :

- Beacon decoder is not a part of TMTC Frontend. It is assumed that beacon is decoded separately , analysed and TMTC Frontend will be triggered on positive beacon.
- Connection between ground station and satellite is half duplex. The system is almost fixed to be half duplex. As of now switch between transmitter and receiver happens after fixed time in TMTC Frontend. It can be modified easily to trigger the switch between two externally.
- SDR (in this case GNU radio) is responsible for bit stuffing and adding flags at the end of each frame.
- TMTC Frontend receives individual frames from SDR and not a continuous sequence of frames.
- Communication with TMTC Frontend can be through simple sockets or more advanced message queues. The function which communicate with other modules are left open. When a communication method is decided, it can be implemented by editing those functions without any other changes.
- Various parameters like maximum resend count, GS and satellite address etc are stored in MissionConstants class. Default values are assumed, which can be changed easily by changing the values in the class.
- MySQL is the database used. It is assumed to be installed on the machine. A seperate setup script will be provided, which will create database and add necessary tables. This has to be run once at the beginning on a new machine .

REFERENCES

[1] Beech, William A., Douglas E. Nielsen, and Jack Taylor. AX.25 Link Access Protocol for Amateur Packet Radio , 2.2nd ed. Tucson, Arizona: Tucson Amateur Packet Radio Corporation,July 1998.

[2] Packet Telemetry. Recommendation for Space Data System Standards, CCSDS 102.0-B-5. Blue Book. Issue 5. Washington, D.C.: CCSDS, November 2000.

[3] Ground systems and operations Telemetry and telecommand packet utilization,ECSS–E–70–41A. Noordwijk,The Netherlands.:ESA Publications Division,January 2003.

[4] Yann Voumard. TM/TC Front End Phase C,S3-C-S E-1-3-TMTC Front End Report .Issue 1.Noordwijk,The Netherlands,ESA,June 2008.

[5] Florian George and Benoit Cosandier. SwissCube Ground Segment Phase B,S3-B-S E-1-5-Ground Segment.Issue 1.Noordwijk,The Netherlands,ESA ,June 2008.