SilverSat Avionics Board

Software and Command Description

Lee A. Congdon

4 April 2023
lee@silversat.org

INTRODUCTION

This document summarizes the hardware and software of the SilverSat Avionics Board. It briefly describes the interfaces with the SilverSat Power Board, Radio Board, Payload Board, and antenna. It also provides an overview of the commands received and sent by the Avionics Board.

REVISION HISTORY

Version 1.0, Apr 11, 2023

Initial release

TABLE OF CONTENTS

OVERVIEW

SilverSat is an educational project for students in middle school and high school. The students are building and programming a 1U cubesat that will take photos from space and transmit them to a widely available media service, currently planned to be Twitter. SilverSat will launch from the International Space Station courtesy of an educational grant from NASA.

HARDWARE

SilverSat hardware includes an EnduroSat structure surrounded by solar cells and four boards implementing the satellite capabilities. The boards are:

1. An EPS-I power board, also provided by EnduroSat, which stores power from the solar cells surrounding the structure and manages the distribution of power to the satellite
2. An Avionics Board, described in this document, which manages the timing of photo and communications on the Payload Board, processes commands from the ground, and manages the timing and content of beacon broadcasts
3. A Radio Board, which manages communications between the satellite and the ground station
4. A Payload Board, which captures images and communicates them to a media service, currently Twitter.

The Avionics Board, the Radio Board, and the Payload Board are developed and implemented by SilverSat. The antenna used by the satellite is provided by EnduroSat.

This document focuses on the hardware and software located on and accessed by the Avionics Board, including its microcontroller, watchdog timer, realtime clock, inertial measurement unit, and ferroelectric random-access memory. It also describes the interfaces to the Power Board, the Radio Board, the Payload Board, and antenna.

SOFTWARE

The software running on the Avionics Board implements initialization, services the watchdog, manages beacon timing, executes commands from the ground, and controls the Payload Board. The software is implemented using the Arduino software environment and is based on the standard setup() and loop() functions required by Arduino.

INITIALIZATION

When power is applied to the Avionics Board, the avionics loop software starts the Serial and logging functions used for board software development, monitoring, and testing. These functions will not be used in the flight software.

The avionics loop then initializes the Avionics Board and the interfaces to the Power Board, the Radio Board, and the Payload Board.

The Avionics Board then pauses for a predefined deployment delay interval.

After the interval, the Avionics Board deploys the antenna, waiting to verify that deployment is successful..

The Avionics Board then commences its standard process loop.

PROCESS LOOP

The process loop, instantiated in the Arduino loop() function in avionics_loop.ino, includes the following steps:

1. Determine whether the process is within the watchdog window. If so, trigger the watchdog.
2. Determine whether the beacon interval has elapsed. If so, send a beacon.
3. Determine whether a command has arrived from the ground or the Radio Board. If so, process the command.
4. Determine whether it is time to take a photo. If so, signal the Payload Board to take the photo.
5. Determine whether the Payload Board is requesting a shutdown. If so, turn off power to the Payload Board after a delay.

The process then repeats.

SOFTWARE ORGANIZATION

In addition to the overall structure provided by the Arduino setup() and loop() functions in the avionics_loop.ino file, the software is organized into classes providing the required capabilities. Most classes are declared in a header file and defined in a cpp file. The exceptions are the Beacon, Message, and Response classes which are header only.

The PowerBoard, AvionicsBoard, RadioBoard, PayloadBoard, and Antenna functions and data are defined as classes.

Hardware such as the EPS-I and the DS1337 are defined as classes. Some hardware has an additional abstraction and encapsulation layer, for example the PowerBoard and the ExternalRTC, respectively, for the devices just noted. This is also the case for the ExternalWatchdog and the IMU.

Command processing is controlled by the CommandProcessor class, which acquires commands from the RadioBoard, validates and parses them with the CommandParser class, and builds a command object with the CommandFactory class. The CommandProcessor then executes and deletes the command object.

The log_utility class generates standard log entries to support the development process. Logging will not be included in the flight software.

TESTING INFRASTRUCTURE

The silver-satAvionics_Software github repository contains versions of the software used for prototyping, unit testing and function testing in the AvionicsTesting1, AvionicsTesting2, and AvionicsTesting3 directories. The readme files in those directories provide additional information.

The FlatSat testing version of the software is located in the FlatSat directory in the repository. In addition to the avionics_loop software, it also contains test simulators for the Antenna, the Power Board, and the Payload Board. An extensive set of tests for initialization, board interfaces, commands, and error conditions are located in the test_avionics directory.

These tests are structured to be run as a complete set beginning with board initialization or can be run individually. They are written in Python and use pytest as the test driver. A utility.py module provides convenience functions. Instructions for setting up and executing the tests are in the readme.

Also present in the repository is a Doxygen file that can be used to generate detailed documentation of the software. Again, instructions are in the readme.

AVIONICS BOARD HARDWARE

The Avionics Board hardware is defined in the Avionics Board repository.

The microcontroller is a Microchip SAMD21G18 32-bit Cortex-M0+ with 256K of flash and 32K of random access memory. The microcontroller runs the avionics loop software in the Arduino runtime environment. The Avionics Board includes an external Texas Instruments TPS3813-Q1 watchdog timer, a Maxim Integrated DS1337 real time clock, an Adafruit MPU-6050 inertial management unit, an Infineon Technologies 32K byte CT15B256J ferroelectric random access memory, and switches to control access to external I2C devices and two serial links. The critical I2C for the realtime clock and the microcontroller serial link are always enabled.

MICROCONTROLLER

The microcontroller pins are defined in the Arduino variant.h and variant.cpp files in the repository. There is no crystal on the Avionics Board, and the Ardunio environment must be configured to reflect that. The microcontroller software can use a bootloader or can be programmed with a hardware device. If a bootloader is used, it must be aware of the watchdog and either trigger or disable it during program loading. The load address of the software must be updated to reflect the presence or absence of a bootloader.

A timer in the microcontroller is used for beacon interval timing. The internal real time counter is not used. The internal watchdog may be used if parts availability of the external watchdog remains an issue.

WATCHDOG

The designed watchdog timer is not currently available. The microcontroller internal watchdog can be provisioned as a backup. The software currently accommodates this, but it is not fully implemented or tested. The external watchdog provides a lower bound and an upper bound for the window and must be triggered within the window. The internal watchdog currently provides only an upper bound for the window.

REAL TIME CLOCK

The DS1337 provides real time clock (RTC) services for the satellite, specifically payload photo timing. Since there is no power available when the satellite is in transit to orbit, the clock must be set by a ground command after the satellite is deployed. The SetClock command serves this purpose.

INERTIAL MANAGEMENT UNIT

The inertial management unit (IMU) is an accelerometer and gyroscope combination. It is not currently used in the Avionics Software. There is a potential use in determining the initial stability of the satellite and using the beacon to transmit that status.

FRAM

The ferroelectric random access memory (FRAM) is not currently used in the Avionics Software.

POWER BOARD INTERFACE

The Power Board stores, transfers and distributes power generated by the solar panels. The firmware on the Power Board manages the power flow and battery heaters as required. The Avionics Board reads Power Board status via the non-critical I2C bus from the board and transmits it to the ground in response to ground commands (GetPower) and in the power beacon character.

The Power Board has an I2C interface to modify its operation. That interface is not used. The Power Board also has an interface implementing the EnduroSat satellite management communications protocol. That interface and protocol are not used.

RADIO BOARD INTERFACE

The Avionics Board communicates with the Radio Board over the Serial1 interface. It receives commands from the ground, sends local commands to the Radio Board, and receives ACKs and RESponses to those local commands on the interface.

This interface is always active and the Avionics Board expects traffic to occur at any time. Ground commands are collected and if appropriate, the signature is validated. The command is then parsed, validated and executed. Some commands generate local traffic with the Radio Board.

The Avionics Board sends a local status request to the Radio Board when a GetComms command is received from the ground station. It sends a local halt command upon receipt of a TweeSlee command.

The Avionics Software also sends beacon local commands to the Radio Board, which adds its status character before transmitting the beacon in morse code. During initialization, the Avionics Software will direct the Radio Board to execute the backup antenna deployment procedure should it be required.

The Avionics Board sends local commands to the Radio Board on receipt of the radio commands listed in the table below.

The Radio Board sends a local response to the status request and radio commands. The Avionics Board converts these responses into messages and forwards them to the ground (via the radio).

SATELLITE COMMANDS

GROUND COMMANDS

Commands are transmitted from the ground, through the ground and satellite radios, and then to the Avionics Board via the Serial1 interface.

Commands are KISS encoded and KISS escaped, although the current implementation does not require escapes since commands and arguments are UTF-8 encoded text.

Each ground command arriving at the satellite will have the second byte (after the initial FEND) set to 0xAA to distinguish it from local traffic, which has a second byte of 0x00. Local traffic is radio responses to commands from the Avionics Board to the Radio Board. These commands and responses are described in the Silversat Local Commands document.

Ground commands are case sensitive and shown in the tables below. Arguments are UTF-8 encoded and blank separated.

Ground commands may be signed to insure integrity. If so, they consist of a sequence number of up to 10 digits, a 16-byte salt, the command, and the 32 byte HMAC. Each section of the signed command is separated by a '|' character. The salt and the HMAC are converted to the UTF-8 representation of their hex values for transmission.

Ground commands sent to the Avionics Board receive an ACK and a RESponse, which is dependent on the command. Invalid and unknown commands, including those with the wrong number or type of parameters, may receive a NACK rather than an ACK and no response. Commands can receive an ACK and fail to execute.

Command length is limited to 256 bytes or the available space in the frame buffer.

CHANGE SATELLITE STATE

COMMAND	PARAMETERS	DESCRIPTION

SetClock (SRC)	Year, month, day, hour, minute, second: integers in UTF-8 encoding	Set realtime clock
BeaconSp (SBI)	Interval in seconds: integer in UTF-8 encoding	Set beacon spacing
PicTimes (SPT)	Year, month, day, hour, minute, second: integers in UTF-8 encoding	Set time for photos; queue length is 5
ClearPicTimes (CPT)		Empty PicTimes queue
UnsetClock (URC)		Unset the real time clock

GET SATELLITE STATE

COMMAND	PARAMETERS	DESCRIPTION
ReportT (GRC)		Reply with real time clock setting; subsumes GetTime
GetPicTimes (GPT)		Reply with photo schedule
GetTelemetry (GTY)		Reply with telemetry
GetPower (GPW)		Reply with power status
GetComms (GRS)		Reply with Radio Board status
GetBeaconInterval (GBI)		Reply with beacon interval

INVOKE SATELLITE OPERATION

COMMAND	PARAMETERS	DESCRIPTION

NoOperate		Acknowledge; subsumes Ping
SendTestPacket		Reply with test message
PayComms		Start Payload Board in communications mode; subsumes Begin tweet
TweeSlee		Stop Payload Board communications; subsumes Halt
Watchdog		Force watchdog timeout

INVALID AND UNKNOWN COMMANDS

COMMAND	PARAMETERS	DESCRIPTION
Invalid		Invalid; used for testing
Unknown		Unknown; command not recognized

RADIO COMMANDS

COMMAND	PARAMETERS	DESCRIPTION
ModifyFrequency	Frequency: 9 UTF-8 digits	Modify radio frequency
ModifyMode	Mode index: 1 UTF-8 digit	Modify radio mode
AdjustFrequency	Frequency: 9 UTF-8 digits	Adjust radio frequency temporarily
TransmitCW	Duration: 2 UTF-8 digits	Transmit carrier wave

BackgroundRSSI	Duration: 2 UTF-8 digits	Report average RSSI
CurrentRSSI		Report current RSSI
SweepTransmitter	Start frequency: 9 UTF-8 digits, stop frequency: 9 UTF-8 digits, steps: 3 UTF-8 digits, dwell: 3 UTF-8 digits	Transmit carrier over range of frequencies
SweepReceiver	Start frequency: 9 UTF-8 digits, stop frequency: 9 UTF-8 digits, steps: 3 UTF-8 digits, dwell: 3 UTF-8 digits	Measure RSSI over range of frequencies
QueryRegister	Register: 5 UTF-8 digits	Report register value

DEPRECATED COMMANDS

COMMAND	PARAMETERS	DESCRIPTION
Halt		See TweeSlee
s_call_sig		Set call sign
g_call_sig		Get call sign
GetPhotos (GPC)		Reply with photo count

LOCAL COMMANDS

Local commands are sent from the Avionics Board to the Radio Board and from the Radio Board to the Avionics Board via the Serial1 interface.

Commands are KISS encoded and KISS escaped, although the current implementation does not require escapes since commands and arguments are UTF-8 encoded text.

Each local command will have the second byte (after the initial FEND) set to 0x00 for local ACKs, local NACKs, and local RESponses, or to one of the values in the table below..

These commands and responses and the command arguments are described in detail in the [Silversat Local Commands] document.

The Avionics Board does not send local ACKs, local NACKs, or local RESponses. It ignores local ACKs and NACKs.

Commands arguments are UTF-8 encoded and blank separated.

Command length is limited to 256 bytes or the available space in the frame buffer.

COMMAND	VALUE
Local Frame	0x00
Remote Frame	0xAA
Beacon	0x07
Release Antenna	0x08
Get Status	0x09
Halt	0x0A
Modify Frequency	0x0B
Modify Mode	0x0C
Adjust Frequency	0x0D
Transmit CW	0x17
Background RSSI	0x18
Current RSSI	0x19

Sweep Transmitter	0x1A
Query Register	0x1C

PAYLOAD BOARD INTERFACE

The Avionics Board communicates with the Payload Board via nine digital IO pins. The three input pins are voted, and the board determines state based on the majority of the pins.

Upon startup, the Avionics Board sets the payload on output pins high, or off.

When the time for a picture arrives, or when the ground sends a command to communicate, the Avionics Board sets the payload states pins appropriately and then turns on the three payload on pins. These pins are then voted on the Payload Board. The Avionics Board waits for a predetermined time before checking for a payload shutdown request.

When the Payload Board has completed its work, it raises the three shutdown pins. Again, the three shutdown pins are majority voted. When the Avionics Board senses the shutdown request, it delays a predetermined interval to allow the Payload Board to complete shutdown, then turn off the payload power.

The Avionics Board has access to a pin signifying that the Payload Board is over current.

ANTENNA INTERFACE

The Avionics Board communicates with the antenna via the non-critical I2C interface.

During startup, the Avionics Board initializes the satellite, waits for the deployment delay, then executes the antenna deployment protocol. Antenna Algorithm 1 is executed for all doors. The Avionics Board then waits for a predetermined period and reads the status of the doors. If all are open, execution proceeds.

If one or more doors are closed, the Avionics Board executes Antenna Algorithm 2 for all of the doors. It then waits a predetermined interval and reads the status of the doors. If all are open, execution proceeds.

If one or more doors are closed, the Avionics Board sends a local command to the Radio Board to execute the backup antenna deployment procedure.

SOURCES

Arduino IDE https://www.arduino.cc/en/software

Arduino Language Reference https://www.arduino.cc/reference/en/

Wikipedia ASCII Definition https://en.wikipedia.org/wiki/ASCII

Arduino Log Reference https://github.com/thijse/Arduino-Log

CPP Reference https://en.cppreference.com/w/

SilverSat https://silversat.org/

Learn CPP https://www.learncpp.com/

Avionics Board Hardware https://github.com/silver-sat/Avionics_Board

Avionics Board Software https://github.com/silver-sat/Avionics_Software

Python.org https://www.python.org/

CircuitPython https://circuitpython.org/

Adafruit https://www.adafruit.com/

SAMD21 Datasheet
https://ww1.microchip.com/downloads/en/DeviceDoc/SAM_D21_DA1_Family_DataSheet_DS40001882F.pdf

Watchdog Datasheet
https://www.ti.com/lit/ds/symlink/tps3813-q1.pdf?ts=1681244431222&ref_url=https%253A%252F%252Fwww.google.com%252F

Real Time Clock Datasheet
https://www.analog.com/media/en/technical-documentation/data-sheets/ds1337-ds1337c.pdf

Inertial Management Unit Overview
https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and-gyro

Ferroelectric Random Access Memory Datasheet https://www.infineon.com/dgdl/Infineon-CY15B256J-SXE-DataSheet-v04_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee624b26e3a

Arduino Cryptography Library https://rweather.github.io/arduinolibs/crypto.html

Visual Studio Code https://code.visualstudio.com/

Visual Studio Code Arduino Extension https://github.com/microsoft/vscode-arduino

SilverSat github https://github.com/silver-sat

Arduino CLI https://blog.arduino.cc/2020/03/13/arduino-cli-an-introduction/

pytest https://docs.pytest.org/en/7.3.x/

SilverSat Local Commands https://docs.google.com/document/d/1Vwpk0ab0HoC62mU7A1fQwpmhmtmZO0VwPtXjQipe0v0/edit?usp=sharing

KISS Protocol http://www.ax25.net/kiss.aspx

KiCad https://www.kicad.org/

Doxygen https://www.doxygen.nl/

The Power Board and Antenna description documents are held under a non-disclosure agreement with EnduroSat.