MOSES Flight Software
A Guide to the Source

David Keltgen, Roy Smart, Jackson Remington

March 11, 2015

Contents

1 Revisions **2**

2 Introduction **3**

3 Main Subsystems **4**

4 Specifications for Operating the MOSES Instrument **5**

5 Architecture Description **7**

6 Main Thread **8**

7 Science Timeline Thread **9**
7.1 Read Out Electronics (ROE) 9

8 Image Writer Thread **12**

9 GPIO Thread **13**

10 FPGA Server **14**

11 House-keeping Link Protocol (HLP) **15**

12 Dictionary of Abbreviations **16**

1 Revision History

Revision	Date	History	Initial
0.1	03-11-2015	Created	DJK

2 Introduction

The MOSES flight software controls the operations of the MOSES instrument, as well as providing telemetry and data transfers to the ground station. This reference guide will go into further detail about the implementation of the blocks in the MOSES Flight Software Functional Diagram.

3 Main Subsystems

Main Subsystems...

1. Tri-M VDX104+ : Main flight computer board. Contains the CPU, SD hard drive, RS-232 ports.

2. CTI FreeForm PCI-104 Virtex 5 FPGA: Connected to the VDX104+ through PCI bus. Main function is to capture science data produced by the ROE. Also implemented on the FPGA is all of the output/input GPIO lines required for operation.

3. Read-out Electronics (ROE): The ROE is the interface between the FPGA and the CCDs that capture scientific data. Upon receiving a command from the flight computer to begin readout it will clock the data contained within the CCDS and stream it to the FPGA.

4. Power board: Board that manages the power systems on the instrument. The flight computer applies a high value to whichever power subsystems have been requested to be activated and then strobes a latch to turn on/off the subsystems.

5. Power board: Board that manages the power systems on the instrument. The flight computer applies a high value to whichever power subsystems have been requested to be activated and then strobes a latch to turn on/off the subsystems.

6. Telemetry transmitter: Science data is sent from the FC to a radio transmitter, which sends the data back to the ground at 10 Mbit/sec

7. Housekeeping link: Consists of two separate radio connections, hkup and hkdown. Hkup operates at 1200 baud and sends commands from the ground to the instrument. Hkdown operates at 9600 baud and sends replies from the instrument to the ground.

4 Specifications for Operating the MOSES Instrument

Specifications...

1. Operate power systems on the instrument. The power systems include:

   (a) Shutter Driver

   (b) ROE

   (c) Premod Filter

   (d) Temperature Control Systems

   (e) Voltage Regulators

   (f) H-alpha Camera

   The flight software is directed to turn on/off power systems through the HLP link.

2. Command the read-out electronics (ROE). This is accomplished using an RS-422 serial connection between the FC and the ROE. Common task include: commanding exposures, flushing the CCDs, and requesting housekeeping data.

3. Receive and save science data. The ROE sends science data over a ?32 Mbit/sec? parallel connection. The FSW should be fast enough so as to not miss any science data.

4. Send science data over telemetry. NASA has provided a 10 Mbit/sec serial line that connects with a radio to the ground. During the course of the mission, the FSW should send as much science data back to earth as possible to mitigate data loss from the harsh re-entry environment.

5. Respond to timers and uplinks. Timers are single lines provided by NASA that instruct the instrument when to execute the different parts of the experiment. Timers include:

   (a) Dark Exposure Start

   (b) Data Start

   (c) Data Stop

   (d) Sleep

dark exposure start, data start, data stop, and sleep. Dark exposures are those which dont open the shutter during the data gathering phase of the science timeline. Uplinks are similar to timers in that they have the same functionality, except they are tied to a physical button on the ground. Using this interface, a user could control the experiment from just the push-button on the GSE.

6. Respond to HLP packets. These packets are sent by the EGSE software on the ground, and provide an additional interface to control the instrument. Two RS-232 connections are provided by NASA to facilitate this interface: HKUP and HKDOWN. Possible types of HLP packets include:

   (a) Uplink

   (b) Shell

   (c) Power

   (d) Housekeeping Requests

   (e) Mission Data Aquisition Requests

5 Architecture Description

The MOSES instrument needs to be able to respond to IO on several different interfaces. This is problematic in sequential programming, as input could be lost while the computer is in a another part of the program or output could be lost while the software is waiting for something else. This issue is alleviated by using a threaded software architecture, which can execute separate subprograms concurrently. Linux provides excellent libraries for threaded programming, known as POSIX threads (pthreads). Pthreads allows the flight software to execute the science timeline, while still being available to respond for input or write data to the hard-disk drive. The challenge with this threaded architecture is one of synchronization. Each thread operates independently, and steps must be taken to ensure that the program executes in the proper order and no memory is accessed simultaneously by two or more threads. In the MSFW thread synchronization is accomplished through so-called lockingQueues. lockingQueues are like a normal queue data structure, except that they take advantage of mutex locks to lock the queue until the accessing thread has completed its operation on the queue. LockingQueues are implemented anywhere in the program where it is necessary to pass data in between threads. Another thread synchronization technique utilized in the MFSW is signals. Signals are objects provided by the Linux OS that allow separate threads or processes to get each others attention through a binary flag. In the MFSW, signals are used to instruct the main process to shut down the flight software.

6 Main Thread

Main Thread..

7 Science Timeline Thread

Sci ti Thread... As the name implies. the science timeline thread controls the overall timeline of the experiment. Science timeline starts off by ensuring that the ROE is active, otherwise the taking exposures would be pointless. Like all of the threads, science timeline is initialized at startup, but waits on a signal, SIGUSR1 in order to continue. Once the signal has been received, the thread will set the current sequence from the already initialized sequence map and begin taking exposures. The takeExposure function starts off by flushing the ROE CCDs 5 times in order to remove any accumulated charge on the CCDs .It then opens the shutter and then uses a sleep function to wait for the designated exposure time(we will have to do some timing experiments to determine if sleep will be accurate enough). If the exposure is a data sequence, then additional time will be added to the exposure length, as represented by the variable PULSE in sciencetimeline.h. This additional time is used to account for the difference in the time it takes to open and close the shutter. Once the exposure is taken, meta data of the image will be saved to the image struct, and sent over telemetry. After that, the thread commands the ROE to read out its exposure data. This process will take four seconds, and this thread can sleep during that time. The last task for the science timeline thread to complete is to enqueue the image to the image writer thread so the image writer thread can independently start writing data to the SD card while the science timeline thread moves to the next exposure. This process is repeated for each exposure in the specified sequence.

7.1 Read Out Electronics (ROE)

The ROE is capable of being in several different modes. Those different modes are:

Default Mode: This is the mode that the ROE is initially in on startup. In default mode, only one command can be sent to it, which is the exitDefault() command. While in default mode, all the ROE does is simply readout an exposure every twelve seconds.

Command Mode: This is the mode that the ROE is in after the exitDefault() command has been sent. The ROE will be in command mode for the majority of the flight. In this mode, commands can be sent such as setting the ROE to a new mode, reading out the CCDs or requesting Housekeeping values.

Selftest Mode: In selftest mode, the ROE will not read out the CCDs at all, but will instead read out a predefined sequenceof vertical bars. This can be useful for testing the flight computers data acquisition abilities. Once the ROE is in selftest mode, it must be reset via the reset() method in order to exit selftest mode. One would also need to exit default mode again after the reset.

Stims Mode: In stims mode, a predefined pattern is generated in much the same

way as the selftest pattern. The difference is that in stims mode the pattern is fed through the CCD readout circuits and thus any extraneous noise and other anomalies show up in the pattern. Also, unlike selftest mode, stims mode can be exited by using the stimOff() method.

Nominal Operation

The ReadOutElectronics object is instructed to exit default mode, reset, and then exit default mode again. The cause for this seeming redundancy is that commanding the ROE to exit default mode while in command mode has no effect. However, commanding the ROE to reset while it is in default mode causes a software error. Therefore, we exit default mode once to make sure that the ROE is commandable, tell it to reset so that we know what configuration it is in, and then finally exit default mode again to make it commandable in a known configuration. The default port for the ROE is /dev/ttyS1 (COM2). The main job of the ROE is to read out the data on the CCD cameras. The two main functions required to acheive this are the flush() and readout() functions. Flush is used to clear any accumulated data on the cameras and readOut is used to get the data from the cameras. The biggest difference between the two, is that flushing the ROE is faster and doesnt send any data down the data link. It is recommended that the ROE be flushed five times in succession before every exposure. Readout does just what one would expect, it reads out the cameras. It is suggested that the ROE be allowed four seconds for reading out data between exposures. The only notable point of confusion for this method is that it also requires a block id, which is basically just a byte that defines how the ROE needs to be read out. There are unique ids for reading out the ROE normally, reading out while in selftest mode, reading out while in stims mode, and there are even some undocumented ids which were used for testing during the construction of the ROE. All of these block ids are documented in the roe.h file.

As a last point of interest, there are also methods for getting and setting the analogue electronics parameters inside the ROE. These methods normally go unused and are usually only useful during testing. These parameters consist of 8 bytes which control the ROEs behavior. In fact, it is by setting these values that the ROE is placed into selftest and stims mode. In those methods these parameters are written automatically. It is suggested that these parameters only be changed if an experienced ROE operator /technician knows what they are doing.

In regards to the Housekeeping data of the ROE, the values that are returned are the raw hex values. These raw hex values are then passed on from the flight software to the ground station. The ground station then displays these raw values, unconverted.

Mutex locks have been implemented in the ROE functions since multiple threads could potentially be trying to access it at the same time. An example of this could be that the Science timeline thread could be telling the ROE to flush the CCDs, while a packet sent from the GSE could be requesting HK values. Although the likelihood of this clashing of threads causing errors in small, it is important to make sure that the

signals being sent to the ROE are exactly what we intended them to be.

8 Image Writer Thread

Image Writer Thread...

Through time testing, it was determined that it takes approximately one second to write each ROE image to disc. Since this mission is very time critical, it was determined that this time was too long to leave as a serial process in the science timeline thread. The result of moving this process of writing images to another thread will result in a large percentage of flight time saved, allowing more exposures captured during the flight.

A ROE image struct is used to store all the image data and meta data for each image that is read out. Here is a brief overview of the image struct: Values for the origin, instrument, observer, and object typically will not change each flight. However, if this software was used under different circumstances, these values could be changed to reflect that. Bitpix, the bits per pixel value, along with the width and height, are set. Channels consists of a single character value that can be anded and ored with channel define values to enable the desired channels. Other values unique to each image include the filename, its actual name, the date and time the exposure started, its duration, the size in pixels of each channel, the name of the sequence the exposure is in, as well as the number of exposures in that sequence.

The image writer thread starts by assigning data to the image struct inside the write_data function. Once all of these values are assigned, it goes to write the file. This is broken down into two tasks, writing the actual image data to disk, and writing the xml file to disc. The xml file contains all of the metadata.

Once the file is written to disc, a pointer to that image is placed into the telemetry queue, where it the telemtry thread performs the task of sending that data to the ground. With this being the last thing the image writer thread needs to do with the local data, it frees it to make room for the next image.

9 GPIO Thread

GPIO Thread...

10 FPGA Server

FPGA Server Thread...

11 House-keeping Link Protocol (HLP)

HLP Threads...

12 Dictionary of Abbreviations

---

1. HLP - Housekeeping link protocol
2. ROE - Read-out electronics
3. FC - Flight computer stack (VDX + FPGA)
4. MFSW - MOSES flight software
5. VDX - VDX104+ (Flight computer)
6. GSE - Ground station equipment