Ball Aerospace COSMOS Open Source Command and Control System

Ryan Melton

Ball Aerospace & Technologies Corp. 1600 Commerce St., Boulder, CO 80301; 303-939-6771 rmelton@ball.com

ABSTRACT

Ball Aerospace COSMOS is a free and readily available open source command and control system for operations and test. It brings a set of functionality to the small sat community that has previously only been available in proprietary and expensive COTS solutions. A set of 15 applications provide automated procedures, realtime and offline telemetry display and graphing, logged data analysis and CSV extraction, limits monitoring, command and telemetry handbook creation, and binary file editing. COSMOS scripting offers the full power of the Ruby programming language allowing operators to send commands, verify telemetry, read and write files, access the network, and even send an email on completion. Advanced debugging functionality allows for single-stepping through procedures, setting breakpoints, and complete logging of all script and user interaction with the system. Detailed data visualization allows for custom screen creation, line and x-y plotting of data, and easy creation of custom 3d visualizations. Offline data analysis and data extraction capabilities make narrowing down anomalies easy. This presentation will discuss all the ways COSMOS can provide a superior, free, and open source command and control system to the small sat community.

INTRODUCTION

Aerospace COSMOS Ball empowers satellite developers of any size to easily create their own user interface for commanding and controlling a satellite or any other embedded system. COSMOS provides a fully featured test and operations system that provides commanding, automated scripting, test data visualization and much more. This paper discusses the huge amount of functionality available in Ball Aerospace COSMOS and its applicability specifically to small satellite developers.

TERMINOLOGY

The COSMOS system uses several terms that are important to understand. Many may be obvious to users within the aerospace industry, but the following table attempts to define these terms clearly for everyone.

Term	Definition
Target	A COSMOS Target is an embedded system that COSMOS can send commands to and/or receive telemetry from.

Command	A packet of information telling a target to perform an action.
Telemetry Packet	A packet of information providing status from a target. Telemetry packets are either periodically received or may be received in response to a command.
Interface	A Ruby class that knows how to send commands to and/or receive telemetry from a target. COSMOS comes with interfaces that support TCP/IP, UDP, and serial connections. Custom interfaces are easy to add to the system.
Ruby	The powerful dynamic programming language used to write COSMOS applications and libraries. Also the language used in COSMOS scripts and test procedures.
Configuration Files	COSMOS uses simple plain text configuration files to define commands and telemetry packets, and to configure each COSMOS application. These files are easily

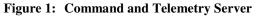
	human readable/editable and machine readable/editable.
Packet Log Files	Binary files containing either logged commands or telemetry packets.
Message Log Files	Text files containing messages generated by a tool.
Tool	Another name for a COSMOS application.

INCLUDED TOOLS

Ball Aerospace COSMOS comes with the following set of 15 applications that are directly available for use with minimal to no configuration.

Command and Telemetry Server

Command and Telemetry Server acts as the hub of the realtime portion of COSMOS. All commands and telemetry packets pass through this tool ensuring everything that happens is logged. It provides realtime commanding, telemetry reception, logging, limits monitoring, packet routing, and system status.

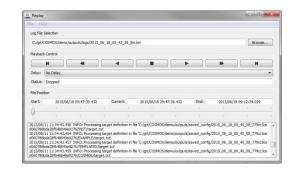


File Edit Help	1									
Interfaces Tar	ets Cmd Packets	Tim Packets	Routers	Loggine	Status					
Interface	Connect/Disconnect	Connected?	Clients	Tx Q Size	Rx Q Size	Bytes Tx	Bytes Rx	Cmd Pkts	Tim Pkts	
INST_INT	Disconnect	true	0	0	0	76	15046	4	196	
INST2_INT	Disconnect	true	0	0	0	0	15046	0	196	
EXAMPLE_INT	Connect	false	0	0	0	0	0	0	0	
TEMPLATED_INT	Connect	false	0	0	0	0	0	0	0	
COSMOS_INT	Disconnect	TH STATUS TEM	0 P1 = 64.7	0 77625 is YELL	0 OW HEAH	0	0	0	15]
015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0 015/08/11 1:29:0		TH_STATUS TEM LTH_STATUS TEM - Unknown 10 b TH_STATUS GROUD LTH_STATUS GROUD LTH_STATUS GROUD LTH_STATUS GROUD LTH_STATUS GROUD LTH_STATUS TEM	P1 = 64.7 MP1 = 64. tyte packe byte packe Dub2STA SMOS/der OUND2STA SMOS/der OUND2STA SMOS/der P1 = 84. PE NORM	77625 is YEL 77625 is YEL et starting: 0 ret starting: 0 ret starting: 0 ret starting: 1US = UNA 10S = UNA 5524500000 5524500000	OW_HIGH LCW_HIGH 000000000000 00100,0PCC 0010,0PCC 00218 RED_ 000218 RED_ 000218 RED_ 000218 RED_	00000000 00000000 ELOW ELOW HIGH OE 0xA8, TEN HIGH OE 0xA8, TEN	P 0.0) 3_cmd.bin	0	15	

Replay

Replay simulates the Command and Telemetry Server for telemetry packet log file playback. This enables use of any of the realtime tools with logged data. Replay is great for playing back scenarios and viewing them on telemetry screens.

Figure 2: Replay



Command Sender

Command Sender provides a graphical interface for manually sending individual commands. Drop down selection of every command and command parameter in the system makes sending individual commands easy. A history pane makes resending previous commands easy.

Target: INST				Command:	COLLECT	•	Send
Description: Star	ts a collect on the instru	ment					
Parameters:							
Name	Value or State		Units		Description		
TYPE:	NORMAL	0		Collect type			
DURATION:		1.0		Collect dura	tion		
OPCODE:		0xAB		Collect opc	de		
TEMP:		0.0	C	Collect tem	perature		
mmand History:	(Pressing Enter on the lin	ne re-exe	cutes th	e command)			

Script Runner

Script Runner executes test scripts and provides highlighting of the currently executing line. Scripts pause if any error occurs, breakpoints can be added, and lines can be reexecuted after a problem has been corrected.



le Edit Search Script Help			
olect_uti.rb 🛠 test.rb			
Running		Go Pause	Stop
best.rb			
9 puts "9" 10 puts "10"			
11			
12 play_wav_file(File.join(Co	🖌 🛃 Hazandous Command		
13 value = ask('Enter the val	Werning: Command INST CLEAR is Hazardous.		
15 cmd ("INST CLEAR")	Clearing counters may lose valuable information.		
16 17 S.times do [index]	Send?		
18 puts index	Yes No		
19 end 20			
21 if false			
22 puts false			
ript Output:			
1015/08/11 11:53:54.832 (test.rb:8): 8 1015/08/11 11:53:54.942 (test.rb:9): 9			
015/08/11 11:53:55.043 (test.rb: 10): 10			
1015/08/11 11:54:01.506 (test.rb: 13): User entered	I' for 'Enter the value'		

Test Runner

Test Runner provides a high level framework for system level testing including automatic test report generation. Test Runner brings the best features of software unit level testing to system level integration and test by breaking tests down into easy understandable test cases. Users can execute entire test procedures or just the specific test cases they need to run for integration or regression tests.

Figure 5: Test Runner

File Script H	ielp									
Pause on Erro	e e	I Manual	Test Suite:	ExampleTestSuit		Ψ.	Start	Se	etup	Teardon
Continue Test Case after Error		Test Group: ExampleTest		¥	Start	Se	etup	Teardo		
Abort Testing	after Error	Break Loop after Error	Test Case:	test_2		*	Start			
Executing Test Ca	se: ExampleTest	: setup	Pass: 0	Skip: 0	Pal: 0					0
	Paused				E	Go		Pause		Stop
ExampleTestSi		e test.rb							_ 0	
13 14 #	wait(2) nd test_1 veri ef test_case	ing #{Cosmos::Test. ifies requirement 1 s with long name 1 ing #(Cosmos::Test.	-							
12 e1 13 14 # 15 da 16 17 18 19 e1 20 21 #	<pre>wait(2) nd test_1 veri ef test_case puts "Runni Cosmos::Tes wait(2) nd</pre>	ifies requirement : s with long name_1 ing #{Cosmos::Test. st.puts "This test ifies requirement :	.current_ verifies	test_suite}	:#{Cosm					
12 en 13 14 # 15 da 16 17 18 19 en 20	<pre>wait(2) nd test_1 veri ef test_case puts "Runni Cosmos::Tes wait(2) nd</pre>	ifies requirement 1 with long name 1 ing #{Cosmos::Test. st.puts "This test	.current_ verifies	test_suite}	:#{Cosm					

Packet Viewer

Packet Viewer provides realtime visualization of every telemetry packet that has been defined. Values within packets are displayed in a simple key-value format that requires no configuration. An autocomplete search bar makes finding values easy.

ile View	Help	File View Help						
		(Q					
larget: IN	ST	Packet: HEALTH_STATUS						
Description:	Health and status from the instrumer	30						
	Item	Value						
22	COLLECTS:	0						
23	TEMP1:	23.601 C						
24	TEMP2:	4.566 C						
25	TEMP3:	76.799 C						
26	TEMP4:	8.568 C						
27	ARY:	["0 V", "1 V", "2 V", "3 V", "4 V", "5 V", "6 V", "7 V", "8 V", "9 V"]						
28	DURATION:	5.0	ŀ					
29	COLLECT_TYPE:	NORMAL						
30	ARV2:	["0.0 C", "0.0 C"]						
	ACCRCMAD.							

Telemetry Viewer

Telemetry Viewer provides custom telemetry screen functionality with advanced layout and visualization widgets. Tabs, graphs, limits bars, and other animated displays can be quickly created. Also, Telemetry Viewer can autogenerate a base set of screens for every telemetry packet that can be customized as needed.

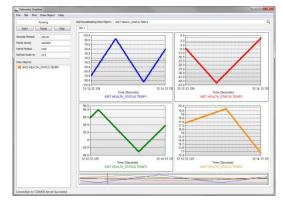
Figure 7: Telemetry Viewer



Telemetry Grapher

Telemetry Grapher provides realtime and offline graphing of telemetry data. Supports both line and x-y style plotting, with multiple tabs, plots, and items per plot. Includes built-in analysis functionality to graph min, max, difference, and standard deviation.

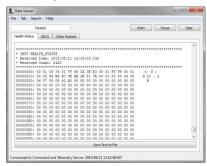
Figure 8: Telemetry Grapher



Data Viewer

Data Viewer provides text based telemetry visualization for items that don't fit into other data visualization paradigms. Great for scrolling log displays and memory dumps.

Figure 9: Data Viewer



Limits Monitor

Limits Monitor monitors telemetry with defined limits and shows items that are currently out of limits or have violated limits since the tool was started. Expected violations can be easily ignored.

Figure 20: Limits Monitor

its Log			
nitored Limits State		Red	
EMP3:	61.443C	Ignore	
EMP3:	61.443C	Ignore	
XAMPLE STATUS IS	STALE	Ignore	
ROUND 25TRTUS:	UNAVAILABLE	Ignore	
ROUND 2STATUS:	UNAVAILABLE	Ignore	
EMP1:	-92.162 C	Ignore	
BHP1:	-92.162 C	Ignore	
ROUND ISTATUS:	UNAVAILABLE	Ignore	
ROUND 1STATUS:	UNAVAILABLE	Ignore	
EMP2:	-13.301C	Ignore	
B4P2:	-13.301C	Ignore	

Telemetry Extractor

Telemetry Extractor extracts telemetry packet log files into CSV data. Highly configurable and supports batch processing to output multiple files at once.

Figure 11: Telemetry Extractor

Telemetry Extracto						
le Mode Item	Help					
Configuration						
Configuration File:	C:/git/COSMO5/demo	o/config/tools/tim_extra	.tor/tim_ex	tractor.txt	Save	Browse
ITEM INST ADCS	TIMESECONDS					
ITEM INST ADCS	TIMEFORMATTED					
ITEM INST ADCS						-
ITEM INST ADCS						
ITEM INST ADCS						
ITEM INST ADCS						
ITEM INST ADCS						
ITEM INST ADCS	VELX					-
THEM INCLASSES	OU Y FORMATTED		-			
			Q	Add Item	Add Packet	Add Target
Target: COSMOS	Packet: LIMITS_	C * Item: ITEM	•	Add Item	Add Packet	Add Target
Column Name:		Text:				Add Text
Downsample Second	1000					·
Downsample Second	s: 0.0					
File Selection						
						Remove
Log Files:	og Files: Browse.					
C/gr/COSMOS/c	remo\outputs\logs	2015_06_18_09_47_39	,tim.bin			
Output File:	C:\git\COSMOS\demo\outputs\jogs\2015_06_18_09_47_39_tim.txt				Select	
Time Period Start:	N/A			Clear	Select	
Time Period End:	me Period End: N/A			Clear	Select	
Packet Log Reader:	Cosmos::PacketLog	Reader				Select
Process		Open in Text			Open in E	

Command Extractor

Command Extractor extracts command packet logs into human readable text.

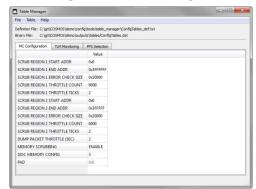
Figure 12	: Command	Extractor
-----------	-----------	-----------

File Mode Hel	P		
Log Files:		Browse	Remove
C:\git\COSMOS\d	lemo\outputs\logs\2015_06_18_09_47_44_cmd.l	bin	
Output File:	C: (git)COSMOS/demo/outputs/logs/2015_06_18_0	9_47_44_cmd.txt	Select
Output File: Time Period Start:	C:\git\COSMOS\demo\putputs\logs\2015_06_18_0 N/A	9_47_44_cmd.txt Clear	Select Select

Table Manager

Table Manager is a binary file editor that can be used to create or edit configuration tables or other binary data.

Figure 13: Table Manager



Handbook Creator

Handbook Creator creates html and pdf documentation of available commands and telemetry packets.

Figure 14: Handbook Creator

🛃 Handbook C 💷 💷 🔀
File Help
Create HTML Handbooks
Create PDF Handbooks
Create HTML and PDF Handbooks
Open in Web Browser

Launcher

Launcher provides a graphical user interface for launching each of the tools that make up the COSMOS system. Supports launching any application that can be started from the command line.

Figure 15: Launcher



SYSTEM ARCHITECTURE

The Figure 16 shows how the 15 applications that make up the COSMOS system relate to each other and to the targets that COSMOS is controlling. On the next page, the key aspects that make up the COSMOS System are discussed in detail.

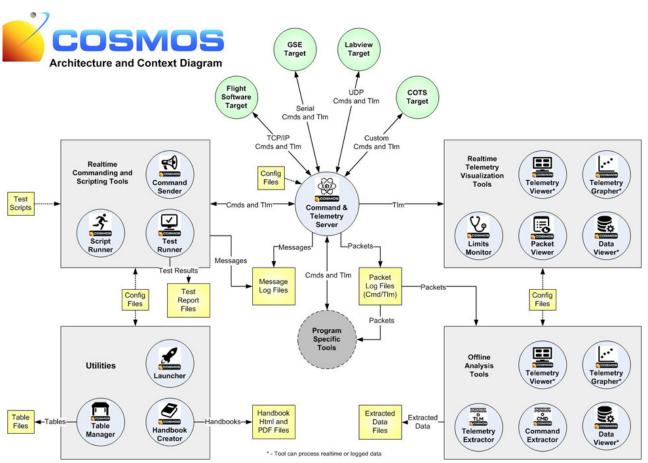


Figure 16: COSMOS Architecture

Key Aspects

- 1. The COSMOS tools are grouped into four broad categories
 - a. Realtime Command and Scripting
 - b. Realtime Telemetry Visualization
 - c. Offline Analysis
 - d. Utilities
- 2. COSMOS can interface with many different kinds of targets. The examples shown in this diagram include Flight Software (FSW), Ground Support Equipment (GSE), Labview, and a Commercial Off-The-Shelf (COTS) target such as an Agilent Power Supply. Any embedded system that provides a communication interface can be connected to COSMOS.
- 3. COSMOS ships with interfaces for connecting over TCP/IP, UDP, and serial connections. COSMOS also supports custom interfaces to connect to anything that a computer can talk to.
- 4. All realtime communication with targets flows through the Command and Telemetry Server. This ensures all commands and telemetry are logged.
- 5. Every tool is configured with plain text configuration files (if any configuration is needed).
- 6. Project specific tools can be written using the COSMOS libraries that can interact with the realtime command and telemetry streams through the Command and Telemetry Server and can also do offline analysis of packet log files.
- 7. Cross Platform COSMOS supports Windows, Linux, and Mac OSX.

KEY BENEFITS TO SMALL SATELLITES

Full Lifecycle System

Supports board level test, box level test, I&T, and onorbit operations providing a consistent user interface throughout the full lifecycle of a product.

Logging

Everything is logged, and even more importantly, tools are provided to easily interpret and use the logs. Whenever an anomaly occurs there are tools already written that are ready to dig into the logs and help figure out what happened.

Superb Data Visualization

Anyone can create great telemetry displays, graph data in realtime, and provide an excellent sense of situational awareness – all without any programming required.

Powerful Scripting

COSMOS comes with a simple API that makes sending commands and checking telemetry easy. However, you are not constrained by your scripting language. COSMOS scripts are written in Ruby, a modern, fully functional scripting language. This allows you to read and write files, and perform live processing that most other systems force you to run offline.

Powerful Test Reporting and Organization

COSMOS Test Runner can produce very reliable test procedures that allow the user to easily execute the entire procedure or only a subset needed for a regression test. Automated test reports created at the end of every run make it very clear that everything passed successfully or where problems occurred.

HISTORY

COSMOS was first developed in 2006 and since then has been used to develop and test more than 30 flight programs at Ball including GMI, OLI, Kepler, WISE, OMPS, Ares, Orion, and numerous defense programs.

Since being open sourced in January 2015 it is now being used with at least 10 major corporations primarily for small satellite development and future operations.

SUMMARY

Ball Aerospace COSMOS is a free and open source command and control system that is immediately available for use. It provides a wealth of functionality much of which is not even available in expensive proprietary tools. For more information and to get started with Ball Aerospace COSMOS please see http://cosmosrb.com.

REFERENCES

1. Melton, Ryan, "Ball Aerospace COSMOS" Retrieved from <u>http://cosmosrb.com</u> June 3rd, 2016