XMM Optical Monitor

U. C. SANTA BARBARA– LOS ALAMOS & SANDIA NATIONAL LABORATORIES

Authors: Tim Sasseen and Cheng Ho

DPU Flight Software Testing

DOCUMENT: XMM-OM/UCSB/TR/0002

Signatures:

Author:	Date: Dec. 22, 1998
OM Project Office	Date:
Distributed:	Date:

TABLE OF CONTENTS

Section 1: Hardware/Software Test Environment 3

Section 2: DPU Flight Software Testing 4

Section 2.1 Overview 4

Section 2.2 Diagnostic Output 5

Section 2.3 Ancillary Programs 6
Section 2.3.1 DPU Simulator 6
Section 2.3.2 Split Archive 6
Section 2.3.3 Data Display Software 6

Section: 2.4 Version Control 7

Section 3: Test Plan for Current Campaign 7

3.1 Current Testing Exposure Script Example 8
3.1.1 Expose.tiny_all.e0_a 8
3.1.2 Expose.everything.e0_a 10
3.1.3 Expose.e00011_a 11

3.2 Explanation of Recent Changes 15

3.3 Test Reports for Particular NCR/ECRs 15
3.3.1 Test report for NCR 111 15
3.3.2 Test report for NCRs 114/115/119/125/126: 17
3.3.3. Test report for NCR 120 19
3.3.4 Test Report for NCR 123 20
3.3.5 Test Report for NCR 131: 21
3.3.6 Test Report for ECR 70: 22
3.3.6 Test Report for ECR 73: 23

Appendix 1: List of Acronyms 24

Section 1: Hardware/Software Test Environment

We describe here the software/hardware and GSE test environment used in testing flight software for the Digital Processing Units (DPU) on the Optical Monitor. The Digital Electronics Module (DEM) also contains the Instrument Control Unit (ICU) that is being supplied by the Mullard Space Sciences Laboratory (MSSL). The ICU software testing are described in a document generated by MSSL, document XMM-OM/MSSL/SP/0207, ‘‘OM FM Software Testing at MSSL.’

The DPU flight software test environment has evolved over the course of the instrument development project. We concentrate here on the more recent configurations. Originally, two electro-optical breadboards (EOB 1 and EOB 2) were built as prototypes for the DPU hardware. This hardware was functionally similar to the eventual flight models. There were minor hardware implementation differences, but the more important difference was that it was not until fairly late in the project (1996) that the EOB units were connected up to a flight like ICU card set. Prior to this time, a simple interface card (synchronous serial interface or SSI card) was used to buffer data between the GSE and the DPU. In this configuration, the software algorithms and numerical computations could be tested on simulated input data, but the detailed timing interaction between the DPU and ICU commanding and data exchange could not be tested. Nonetheless, it was in this form that most of the DPU higher level science operations, such as finding guide stars, tracking, shift and add, etc. calculations were tested.

Since the first operational ICU card set were constructed near the end of 1996, the DEM team has attempted to test the ICU/DPU unit together as a unit as much as possible. A typical configuration is illustrated in Figure 1, which shows the Electronic Ground Support Equipment (EGSE) connected to a flight like DEM hardware system. The hardware configuration includes a Sun workstation (Sun OS 4.1) networked via a 100 Mb/s Ethernet link to a dedicated Force data acquisition and control computer running VxWorks housed in a VME bus card cage. Also in this card cage are cards to simulate the telescope systems and a line driver card to power cables to the DEM hardware.

As of June 1998, DPU software testing has taken place in the flight spare DEM. This unit has been built to be identical to flight units, although the ICU components are not flight quality. We believe the flight spare DEM behaves essentially identically to the flight units and provides a virtually identical test configuration for flight software. Naturally, there are and will be environmental differences between the flight DEM’s mounted on the spacecraft and the flight spare sitting on a bench connected with our GSE. As of this writing, we are aware of no differences in performance of the flight and flight spare DEM’s because of their different environments.

DPU Software Testing Hardware Suite

J02

J03

J04

Sparc Station

Containing:
Force Computer
Tel-Sim Card
(SSI card)

J01

Figure 1: Typical hardware setup for flight software testing

Section 2: DPU Flight Software Testing

Section 2.1 Overview

Initial testing of software begins by loading codes into the ICU and DPU via code-loading scripts and verifying the boot sequence. Following successful loading of codes, a pre-defined set of simulated exposures is run through the system via exposure scripts. Simulated data of sky images from the Optical Monitor detectors reside on disk in the Sun Workstation. This data is sent from the Sparcstation through the telescope simulator card (tel-sim card) and buffered to simulate an incoming photon stream. Spacecraft jitter is included as a default. The data is passed through the ICU to the DPU as would be real data from the Optical Monitor detectors. Window configurations and memory windows are established as they would be at the start of each exposure. The DPU processes each photon datum as it would in flight, building up a flight image, including shift and add, for the specified number of frames or exposure time for each exposure. Finally, the image is compressed by the DPU and sent back through the ICU/Spacecraft interface. Typically, a sequence of exposures lasting several hours is run, testing all expected acquisition, compression and interrupts simultaneously.

Over the course of years, many new routines have been written and added to the DPU flight software suite. These range in complexity from single purpose library routines (e.g. pad a sequence of 24 bit words to 32 bits) to a more complex package containing several different programs, such as the data compression task. Simple routines are tested individually with sample input data and the output data compared with expected output. Once verified with sample data individually, routines will then be added into their parent package and testing continued by running exposures.

The philosophy for DPU testing has been to run a fixed set simulated detector data through the flight software such that changes in the output data are immediately recognizable. This philosophy is different than what one might do if random photon data were generated with each run, and is a more stringent test of software operation.

Naturally, some software changes will cause output to be different than previous versions; when this occurs the new output is checked in detail for accuracy and for agreement with expected results.

Section 2.2 Diagnostic Output

There are two types of data output from the exposure runs on the DEM that may be examined to investigate the success of a particular software run. The first is the log file generated by the Unix ’tee’ logging process. The second is the collected data archive.

In the log file, there are two types messages generated by the GSE codes. The first are the commands send into the DEM by the GSE. These have a one-to-one correspondence with the exposure script. The second are the messages generated by the GSE in response to output from the DEM. Again, there are two general types of DEM-induced messages. The ICU generic messages look like

```
ver=4 type=0 headflag=1 apid=1024 segflags=3 count=2613
len=111 spare=0 checkflag=3 type=1 subtype=1
timecoarse=113105 timefine=5d9b
```

The meaning of these terms and the contents of a packet are described in RS-PX-0032 ‘‘Packet Structure Definition’ and the Telemetry and Telecommand Specification XMM-OM/MSSL/SP/0061. These messages are turned on or off by the set_dump_tm command.

The second type of DEM-induced messages are DPU-specific. A few examples of these are:

```
Heartbeat 2829 0070 0000 2710
done SAA taken
done switch frame taken
```

Some of the DPU messages are always passed out from ICU. Some have to be turned on. See the Telemetry and Telecommand Specification (XMM-OM/MSSL/SP/0061) for more details. In testing, we selective turn on/off the DPU messages to balance diagnostics information content and the TM bandwidth limitation.

The second type of data output that are used for diagnostics is the collected data archive. These are in the form of concactnated TM packets. We use the split_archive utility to break them up and specifically extract the different DPU generated data sets. At the moment, the DPU generated data sets are examined using binary comparison. In addition, the split_arhive also generates a text file *_alerts. These are the collection of all DPU alerts that were passed out by the ICU. Examination of the time sequences of these alerts and their relation to those in the log file in some occasions plays an important role in the diagnosing or debugging a test run.

Section 2.3 Ancillary Programs

A number of non-flight support programs and scripts have been developed to create, separate and examine DPU data. We discuss these here.

Section 2.3.1 DPU Simulator

The simulator is the basis for testing and operation the DPU during testing. It is described in detail in document XMMOM/PENN/ML/0002 and is available from the UCSB web site (http://xmmom.physics.ucsb.edu). We have used to simulator to create input data files based on a ray trace of astronomical sources through the telescope, filter and detector systems. These data files can be used as inputs to either a DPU simulator that runs entirely on the Sun workstation, or put into the flight spare DEM.

Section 2.3.2 Split Archive

This command script calls several routines designed to decompress telemetry archive files and restore the data contained within into separate files corresponding to each exposure. Because data from different exposures may be contained in different telemetry output files, care must be taken to correctly assemble data from a given exposure. Sample routines are:

depacket_dpu_only
mem_depacket
dissect_dpu_data
decmprss

Section 2.3.3 Data Display Software

A number of routines have been written in IDL to display image and fast mode exposure data from the DPU. These are used internally by the DPU software development team during flight software verification. Sample routines are:

fast_display.pro
display_image.pro

Section: 2.4 Version Control

The DPU software has been developed in the Concurrent Version System (CVS) software control environment. CVS is a Unix–based facility to provide archiving of accepted code and logging of changes. Each developer checks out the current version of a code module from a central repository, works on it and tests it, then only after the code is verified is the modified code module checked back into the central repository. A given code module is checked out and in with changes that result from one NCR and each check-in logged with reference to a that NCR.

Our current test requirement for new or modified codes to be checked into CVS is that they must have been part of a successful exposure sequence of duration more than 12 hours. Once the individual routines are checked into CVS, the whole code package is tested for a much longer duration. Since about mid-1998, the code has been stable enough that much longer exposure sequences, over 100 hours, have been achieved. (Our current record is 191 hours.) For routines that are not normal part of exposures, such as engineering modes or boot sequences, their operation has been verified by testing their operation multiple times, although no strict number of successful tests has been adopted as a requirement for acceptance. CVS provides a sequenced version number for each routine, which is the mechanism to alert developers who might be working on the same code module simultaneously. Thus far, the US group has been small enough that we could coordinate simultaneous development of codes by telling each other ‘‘I’m working on this module, don’t modify it until I’m done.’

The end result of our software development and testing efforts is that nearly all of the flight software has been tested for thousands of hours of running simulated exposures in the flight spare and EOB systems. Of course, the most recently changed codes will have less total running time.

Section 3: Test Plan for Current Campaign

It has always been the plan of the OM team to extensively test the ICU/DPU software both separately and together. Because the two software sets were developed at different institutions, we have historically had the least opportunity to test the way the ICU and DPU software work together and that is where we have been concentrating our efforts for the last year. This work has proceeded well and a number of fatal and non-fatal software errors have been detected and solved. The general test plan for each error/bug is summarized in Section 3 of XMM-OM/MSSL/SP/0207, ‘‘OM FM Software Testing at MSSL.’

The level of software changes that have occurred in the last several months in order to solve observed bugs are typically minor, often one line. During this time, we are at the level of detecting bugs that might only show up after one or several days of running the software. This not only makes identifying and solving them time consuming, but also indicates the level of software performance is already fairly robust and any remaining problems should be minor.

3.1 Current Testing Exposure Script Example

Here is a typical example of a set of exposure scripts used to run simulated exposures through the Flight Spare DEM. Editing of non-instructive lines has been performed and these scripts are not intended to be run verbatim.

3.1.1 Expose.tiny_all.e0_a

```
# $Id: expose.tiny_all.e0_a,v 1.1.1.1 1998/03/24 00:49:51 xmmom Exp $
#
#       This srcipt chould make a total of 325 * 3 exposures
#       each exposure should have a unique data output file.
#       CH 12/12/97
#
```

```
< tc_clean_slate

#
#           1
#
< expose.everything.e0_a

cd "/data/xmmom/archive"

save_archive "e00009","e01009"
save_archive "e00008","e01008"
...
transfer archived files to uniquely named archive
...

save_archive "e00prg","e01prg"
save_archive "e00lcl","e01lcl"

copy  "save_archive.sh","save_archive_now"
taskDelay (20*60)

cd "/home/xmmom/test_data"
#
#           2
#
# taskDelay 60*1

< expose.everything.e0_a

cd "/data/xmmom/archive"

save_archive "e00009","e02009"
save_archive "e00008","e02008"
save_archive "e00007","e02007"

etc.
```

Repeat each set of exposures about 50 times.

3.1.2 Expose.everything.e0_a

```
#
# $Id: expose.ev.e0_a,v 1.1 1998/05/28 20:25:53 xmmom Exp $
#
#
#  This script executes 3 representative exposure scripts
#
#  CH 09/22/98
#


tc_init_dpu
taskDelay 60*25
tm_wait_for_alert DA_EOT_INIT_DPU

#

< expose.e00011_10_a

#

< expose.e00012_40_a

#

taskDelay 30

close_packet_archive
open_packet_archive "/data/xmmom/archive/e000t0"

tc_send_command IC_FLUSH_CMPRS
tm_wait_for_alert DA_EOT_FLUSH_CMPRS

taskDelay(60*90)

close_packet_archive
```

3.1.3 Expose.e00011_a

Notes:

wtb – ‘‘write to blue’ – streams archived photon data as input for exposure

```
#
# $Id: expose.e00011_a,v 1.1 1998/05/28 20:25:42 xmmom Exp $
#
# >>>>    Exposure e00011    <<<<
#
# CH03/18/98
#
# Full 100 frame exposure using e40011 simulation data set.
#
# Tracking by hand control, delayed compression
#

rm "/home/xmmom/archive/e00011"

close_packet_archive

open_packet_archive "/data/xmmom/archive/e00011"

#####################################################################
#
#   Set up some numbers first
#
#####################################################################

# ref frame exp
frame_time = 20

# wtb delay
wtb_delay = frame_time*60 - 5

# tracking exp
tracking_exp = 10

# tracking delay
tracking_delay = 30

# xytable delay
xytab_delay = 2

# exposure time. Be very careful here!!
# Make sure you have the right numbers

exposure_time = 10

tc_set_frame_time frame_time*1024

#--------------------------------------------------------------
# acquire field, turn the following statements on if FAQ
#--------------------------------------------------------------

# tc_send_command IC_INIT_EXP

tc_set_exp_no 0xe00011
```

```
tc_send_command IC_INIT_EXP
taskDelay 60*3

tc_prog_mem_wind "/home/xmmom/sim_data/e40011/mmw.cfg"

tc_prog_sci_wind "/home/xmmom/sim_data/e40011/scw.cfg"

tc_report_tracking 1

tc_enbl_verbose 0,1

tc_enbl_verbose 3,1

tc_set_exposure_time exposure_time

# choose_guide_stars

tc_send_command IC_CHOOSE_GS

taskDelay 5

###############################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.0_a"
tm_wait_for_alert 0xBEAD

wtb "/home/xmmom/sim_data/e40011/e40011.mic.0_b"
tm_wait_for_alert 0xBEAD

wtb "/home/xmmom/sim_data/e40011/e40011.mic.0_c"
tm_wait_for_alert 0xBEAD

###############################################

taskDelay 60*(frame_time)

# time allocated for guide star selection
taskDelay 60*(25)

# time allocated for window setup
taskDelay 60*(20)

#-----------------------------------------------------------------

wtc 0x00

# tc_enbl_events blue1,1
# tc_enbl_events blue2,1

# tc_enbl_blue_fast_mode

#------------------------------------------------------------------
# If auto-piloting, then use "tc_send_command IC_TRACK_GUIDE_STARS"
#

# tc_enbl_by_hand_track 1

# taskDelay 60*10

#------------------------------------------------------------------

tc_send_command IC_TRACK_GS
```

```
taskDelay 60
# tm_wait_for_alert DA_BEGOF_EXP

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.1"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.2"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.3"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.4"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.5"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.6"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.7"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.8"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.9"
taskDelay wtb_delay

######################################################

wtb "/home/xmmom/sim_data/e40011/e40011.mic.10"
taskDelay wtb_delay

tm_wait_for_alert DA_ENDOF_EXP
tm_wait_for_alert DA_COMPLETE_EXP


taskDelay 60*2

# close_packet_archive
```

tm_clear_sem

wtc 0x00

#---------------------------- done

3.2 Explanation of Recent Code Changes

A summary of software changes in the last several months of development are presented in documents XMMOM/UCSB/TC/0041 and XMMOM/UCSB/TC/0042. The first document covers changes are in reponse to DPU related NCR's 89 – 96 and DPU ECR's 62 – 68. The second document covers DPU NCR's 111 – 127 and DPU ECR's 70 – 77. These documents are available by anonymous ftp from *http://xmmom.physics.ucsb.edu*..
Below are test reports for software changes in response to recent NCR's. Testing for changes in response to earlier NCR's proceeded similarly.

3.3 Test Reports for Particular NCR/ECRs

3.3.1 Test report for NCR 111

Purpose of tests

To validate the code changes made for NCR 111

Test Location: LANL/UCSB

Test environment.

DEM/FS
DPU/ICU GSE Compliment
SNLA Telesim Card

Expected results.

The DPU boots up properly without error.

Tests performed.
----

1. Turn off DEM completely
2. Turn on DEM
3. Load the DPU and ICU codes.
4. Tune tc_clean_slate to bring up DPU properly.

Results

The DPU boots up properly without error.

Performed by

CH, 1998 ~ Oct. 1

3.3.2 Test report for NCRs 114/115/119/125/126:

Purpose of tests

To validate the code changes made for NCR114/115/119/125/126

Background

This is the so-called ’DPU Busy Spot’ problem. It is related to the internal timing and resource allocation among the four processors inside the DPU. NCR114/115 were early manifestation of the problem and ’thought’ corrected after a simplistic tuning of the DPU operation paramaters. NCR 125 was noted after 191 hours of execution, but the symptom is that of the busy spot problem. The problem was resolved after a concerted effort over a month.

Test Location.

LANL/SNLA

Test environment.

DEM/FS
DPU/ICU GSE Compliment
SNLA Telesim Card

Expected results.

DPU executes without extraneous reset or un-understood anomolies.

Tests performed.
----

1. Reset DEM
2. Load the new DPU codes
3. Run the standard exposure script, expose.all.e0_b.
4. Examine DPU data output.
5. Perform binary comparison of DPU data products to confirm successful exposure and screen for anamolous results.

Results

The DPU data product does not exhibit un-understood anamoly after > 52 hours of continuous execution. Specifically, we have not seen any anamolous results at the DPU busy spot.

Special Note

Any future modification of the DPU codes will require a careful
assessment of the timing split among the processors to ensure that
each processor lives within its own allotment.

Performed by

CH, 1998 Dec 10-12.

3.3.3. Test report for NCR 120

Purpose of tests

To validate the code changes made for NCR120

Test Location.

LANL/UCSB

Test environment.

DEM/FS
DPU/ICU GSE Compliment
SNLA Telesim Card

Expected results.

The DPU boots up properly without error.

Tests performed.
----

1. Turn off DEM completely
2. Turn on DEM
3. Load the DPU and ICU codes.
4. Run tc_clean_slate to bring up DPU properly.

Results

The DPU boots up properly without error.

Performed by

CH, 1998 ~ Nov 10.

3.3.4 Test Report for NCR 123

Purpose of tests.

To validate the code changes made for NCR123

Test Location.

LANL/SNLA

Test environment.

DEM/FS
DPU/ICU GSE Compliment
SNLA Telesim Card

Expected results.

DPU heartbeat word 9-bit carry status of tracking on/off.

Tests performed.
----

1. Reset DEM
2. Load the new DPU codes
3. Run tc_clean_slate script, with DPU Heartbeat passing by ICU enabled
4. Turn on/off the DPU tracking by the tc_set_trk command
5. Confirm the toggling of the 9-th bit in the DPU status word

Results.

The DPU software performs as expected after the correction of NCR123.

Performed by

CH, 1998 Dec 10.

3.3.5 Test Report for NCR 131:

Problem statement:

The n_dtw field in the DP_WDW gives the valid detector window plus one. This is due to DPU internal allocation where dtw_0 has been designated as unsed by BPE but counted by the DPU. Following the agreement to conform to this convention between DPU and ICU, the ENG portion of the DPU codes need to be modified for compliance.

Severity: Minor.

Purpose of tests

To validate the code changes made for NCR131

Test Location.

LANL/MSSL

Test environment.

DEM/EOB1 at MSSL
DPU/ICU GSE Compliment

Expected results.

DP_WDW output from ENG modes conforms to the +1 convention, except for ENG4 and 5.

Tests performed.
----

1. Reset DEM
2. Load the new DPU codes
3. Run the standard exposure script for ENG mode 3-6 at MSSL.
4. Examine DPU data output at LANL.
5. Confirm that DP_WDW conforms to convention

Results

DP_WDW output from ENG modes conforms to the +1 convention.

Performed by

PJS at MSSL, CH at LANL, 1998 Dec 16-17.

3.3.6 Test Report for ECR 70:

Purpose of tests

To validate the code changes made for ECR70.

Test Location.

LANL/UCSB

Test environment.

DEM/FS
DPU/ICU GSE Standard Environment
SNLA Tel-sim card

Expected results.

DPU operates correctly; no changes to science and memory window behavior.

Tests performed.
----

1. Make the DPU codes.
2. Run successful exposure sequence with different window configurations.

Results

DPU operates as expected.

Performed by

Cheng Ho, 1998 Oct 7.

3.3.6 Test Report for ECR 73:

Purpose of tests

To validate the code changes made for ECR73. The changes are in the make utility scripts and not in the flight code proper.

Test Location.

LANL/UCSB

Test environment.

DPU code development environment

Expected results.

Compiled codes in .p and .cmd format co-located.

Tests performed.
----

1. Make the DPU codes.
2. Examine the DPU codes delivery directory: dpu/development/eeprom.

Results

Compiled codes in .p and .cmd format co-located.

Performed by

Cheng Ho, 1998 Nov 10.

Appendix 1: List of Acronyms

CPU	Central Processor Unit
CVS	Concurrent Version System
DEM	Digital Electronics Module (made up of DPU and ICU)
DPU	Digital Processing Unit (supplied by US)
ECR	Engineering Change Request
EOB	Electro-Optical Breadboard
ICU	Instrument Control Unit (supplied by MSSL)
LANL	Los Alamos National Laboratory
MSSL	Mullard Space Sciences Laboratory
NCR	Non-Conformance Report
SNLA	Sandia National Laboratory
UCSB	University of California at Santa Barbara