NATIONAL OPTICAL
ASTRONOMY
OBSERVATORY

950 N. Cherry Ave.
P. O. Box 26732
Tucson, Arizona 85726-6732
(520) 318-8000 FAX: (520) 318-8303

MONSOON/Torrent

Software Setup

PAN processes, Python tools,
mec and **mborg** engineering consoles

NOAO Document

MNSN-AD-08-0005

Revision: 2.1

Authored by:
Nick C. Buchholz
10/25/2004

Please send comments:
nbuchholz@noao.edu

Revision History

Version	Date Approved	sections Affected	Remarks
0.0.1	10/25/2004	All	First release draft
0.0.2	12/20/2004	All	General update. Added info disk partitions and system configuration.
0.0.3	3/7/2005	All	Added steps in setting up a new MONSOON PAN. Rearranged the document to support the list.
0.0.4	10/31/2005	Appendix III, IV and V	Added additional information on focal plane configuration, descrambling and required software attributes.
1.0	5/31/2006	All	Reformatted and edited.
1.1 2.1 (sic)	20120716 20120726	All All	Added information for Torrent systems Major corrections! (pnd)

Table of Contents

Table of Contents (Cont.)

List of Figures

List of Tables

1.0 Introduction

1.1. Scope

The MONSOON and Torrent base software consists of four processes that run on the PAN and a number of shared libraries that are used by these processes. In addition, the normal operating mode is for these processes to be controlled from a remote machine running a GUI process called a “Science” or ”Engineering” client. MONSOON/Torrent systems can be controlled by the **mec**, NOCS, mop or **mborg**. In general Orange systems are setup to be run from the **mec**, NOCS or mop and Torrent systems are setup to run from the **mborg**, NOCS or mop. In addition, the Torrent systems require the installation of python2.5, Pmw1.3, numpy2.0.0, pyserial2.5, pyfits2.3.1 and all of the NOAO Monsoon/torrent python packages.

The purpose of this document is to explain how NOAO configures PAN and client systems to run the MONSOON/Torrent software and to give the new MONSOON/Torrent user instructions for setting up a MONSOON PAN system. In addition, section 2.0 contains a set of steps to be followed when using a binary or source distribution of the MONSOON software. The intended audience for this document is: all those with a requirement to configure or use a MONSOON system.

1.2. Acronyms and Glossary

1.2.1. Abbreviations and Acronyms

AC	Acquisition Camera
ADC	Analog to Digital Converter
AFE	Analog Front End
DAC	Digital to Analog Converter
DCS	Detector Controller System (software)
DHE	Detector Head Electronics
DHS	Data Handling System
ECS	Enclosure Control System
ES	Embedded System
FITS	Flexible Image Transport System
FP	Focal Plane
FPA	Focal Plane Array
GPX	Generic Pixel Server
mborg	MONSOON basic operator response GUI
mec	MONSOON engineering console (for Orange systems)
MONSOON	Not an acronym
NICD	NOAO Interface Control Document
IAS	Image Analysis System
ICS	Instrument Control System
IDPS	Image Data Preprocessor System
ID	Identifier
IR	Infrared
LAN	Local Area Network
N/A	Not Applicable
NOCS	NOAO Observation Control System
PDF	Parameter Description File
PSM	Power Supply Module

ROI	Region of Interest
SSH	Secure shell, used to connect with a MONSOON PAN system
SUS	Status Update System
TBD	To Be Decided
Tcl/Tk	Scripting language and GUI used by the **mec**
Tkinter	Python interface to TK
Torrent	not an acronym
TSM	Transition Module

1.2.2. Glossary

Attribute	An entity that describes some aspect of the configuration of a pixel server system. Examples are: the level of a voltage or the state of a shutter. The pixel server system will use some attributes as command parameters. The NOCS communicates with a science instrument by sending it sets of attributes and values.
Byte	8 bits.
Command	An instruction requiring a system to start some action. The action may result in a voltage changing or some internal parameters being set to particular values. A command may have command parameters (arguments) that contain the details of the instruction to be obeyed.
Data Array	The data, while it is stored in data processing memory, which resulted from one or more readouts of an IR array or CCD detector.
Data Set	A self-contained collection of data generated as a result of a pixel server obeying a *gpxStartExp* command. Each *gpxStartExp* command results in one and only one data set.
Detector Head Electronics	The lowest level hardware system, normally closely connected to the detector and the dewar in which the detector resides. Sometimes referred to as the detector controller.
Exposure	The process and the data resulting from the process of resetting or clearing a detector, exposing it to photons and then reading one or more frames to determine the photon levels. These frames are processed into a data array, called an exposure, which may be further processed. For example, an exposure would be the data array that results when a single Reset-Readout-Integrate-Readout cycle is performed on an IR detector or a single CCD Clear-Integrate Readout cycle.
Exposure Sequence	The process by which valid data is produced. Various levels of exposure sequencing occur during an observing run. At the lowest level there are the Reset-Readout-Integrate-Readout or Clear-Integrate-Readout cycles that result in a single IR or OUV exposure. At the highest level are the observing sequences, which move the telescope, configure the instrument and take a series of exposures that create an observation.
Frame	The result of a single readout of an array. Each frame represents the signal values obtained from reading the entire ROI being read out of the detector. Multiple frames may be processed into a single exposure.
Generic Pixel Server (GPX)	A set of software routines that implement a pixel server. A generic pixel server does not require any particular set of proprietary hardware or software.
Image	The array of detector pixel and description data representing a science or

	diagnostic exposure or combined set of exposures. An image is capable of being displayed or processed as a discrete entity. The values in the array may be stored in memory or on disk and are related to the data taken by the detector by some processing algorithm. For example, an image may consist of all the co-added and averaged exposures in one beam of a chop mode *gpxStartExp* command.
Image Acquisition System	A system of software and hardware capable of producing images from a detector on command.
Image Server	A system of software and hardware capable of producing images from a detector on command.
Instrument Control System	A set of software routines designed to control and configure a science instrument to take science observations.
Observation	The process of exposing the detector to photons through the telescope in one or more exposures. The result of an observation is an image.
Pixel Acquisition Node	A component of a generic pixel server, this is the computer that handles the interface to the detector head electronics and the image pre-processing of the data stream from the detector head electronics.
Pixel Server	A set of software and hardware that accepts commands and produces a set of pixels (an image) related to those commands.
Read	When used as a noun to describe instrument data, a single read of a pixel on the detector. A read may consist of several A/D conversions of the pixel data that are averaged or processed in some other way to produce a single integer output value for the pixel. A readout is made up of one read of each pixel in the detector ROI being read.
Readout	When used as a noun to describe instrument data, a single read of every pixel on the detector. A frame is made up of one or more readouts averaged pixel by pixel.
Region of Interest (ROI)	A sub-array of the available detector area. There are two types of sub-arrays that can be defined. The *sequence ROI* is an ROI on the active surface of the array used to increase the frequency of the array readout. The *data reduction ROI* is an arbitrary rectangle of any size that fits on the array. Data reduction ROIs are defined to reduce the volume of data sent to the disk or DHS even when the entire array is being read out.
Supervisory Node	A computer capable of controlling multiple image acquisition systems. The computer that runs the software that conforms to the GPS interface.
Value	The value associated with an attribute.
Word	Four bytes or 32 bits.
MONSOON Image Acquisition System	A generic pixel server. An extensible, modular image acquisition system. The design of the system is, to the extent possible, independent of the hardware being used in a particular implementation. Each component of the system should be capable of replacement by a similar component without having to redesign the rest of the system. Each component of the software is, as far as possible, independent of the underlying hardware and as modular as possible.

1.2.3. Reference Documents

SPE-C-G0037, “Software Design Description”, Gemini 8m Telescopes Project.

NOST 100-1.0, “Definition of the Flexible Image Transport System (FITS)”, NASA Office of Standards and Technology

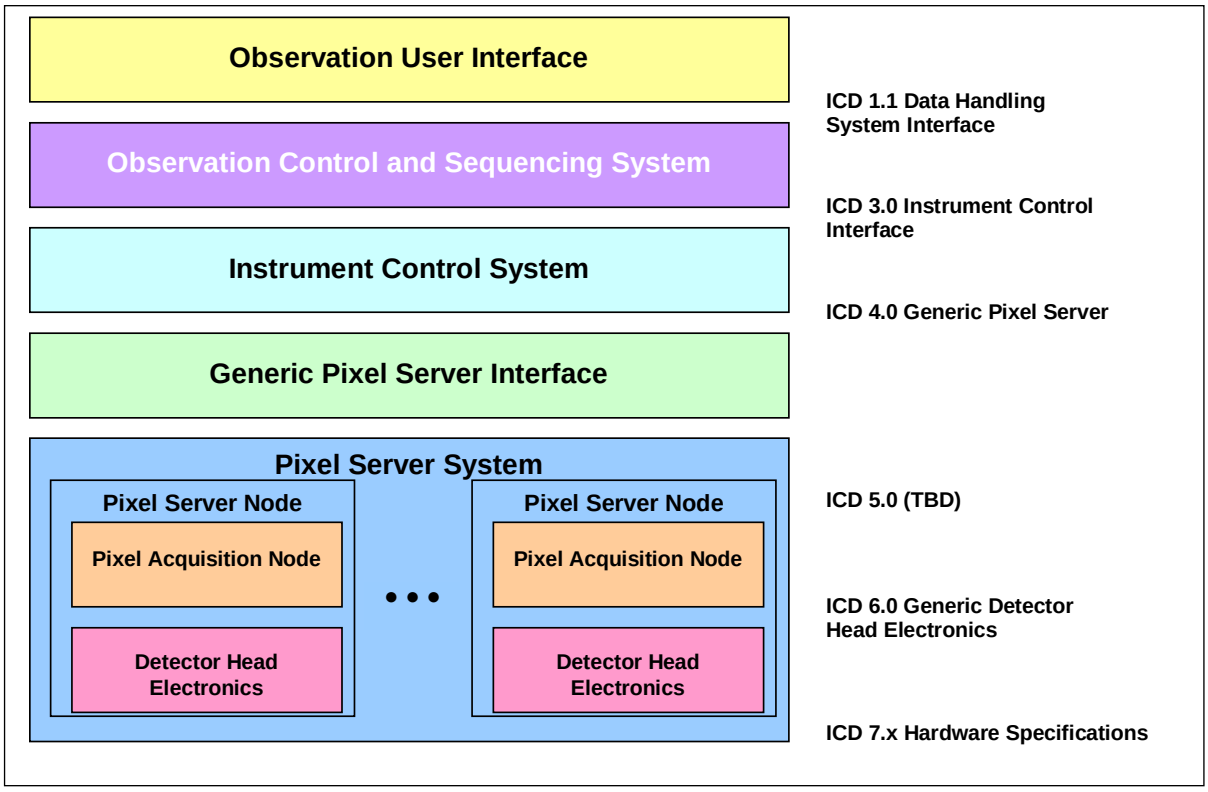
IEEE Std. 610.12-1990 - “IEEE Standard Glossary of Software Engineering Terminology”, Standards Coordinating Committee of the IEEE Computer Society, USA, 19901210

ANSI/IEEE Std 754-1985 - “IEEE Standard for Binary Floating-point Arithmetic” - Standards Committee of the IEEE Computer Society, USA 19850812

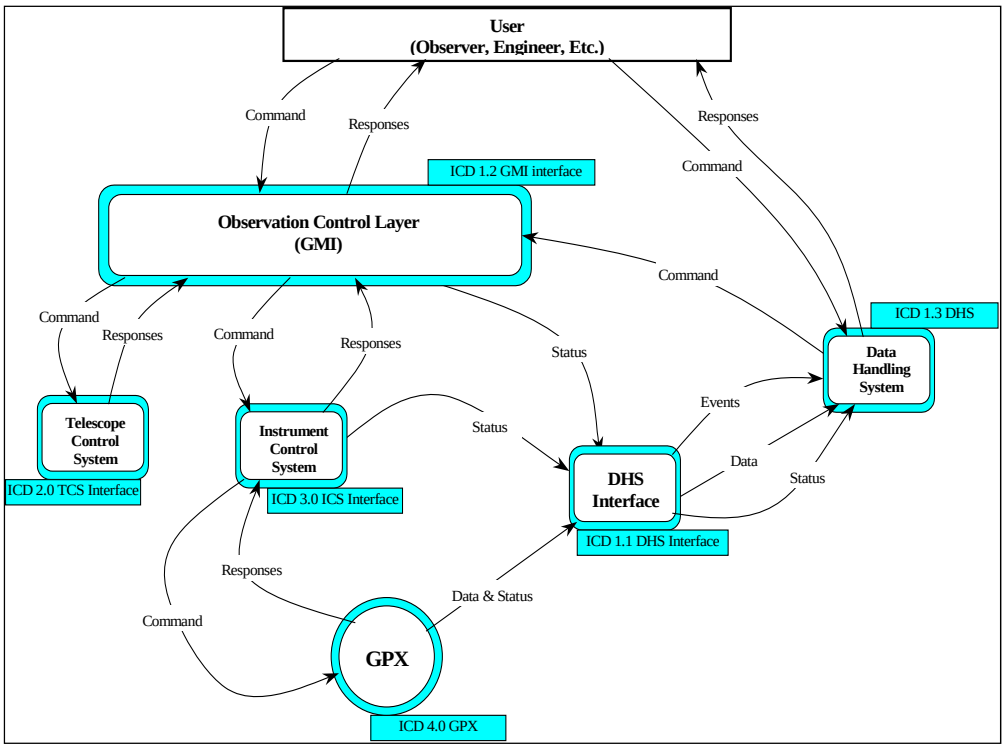
ICD 4.0- “Generic Pixel Server (GPX) Interface - Generic Pixel Server Communications, Command/Response and Data Stream Interface Description”, Nick C. Buchholz (NOAO), Barry M. Starr (NOAO), Version 0.1.2, dated: 20020311, NOAO Document Number MNSN-AD-01-0002

ICD 6.0- “Generic Detector Controller - DHE Command and Data Stream Interface Description”, Nick C. Buchholz (NOAO), Barry M. Starr (NOAO), Version 0.1.2, dated: 20020311, NOAO Document Number MNSN-AD-01-0004

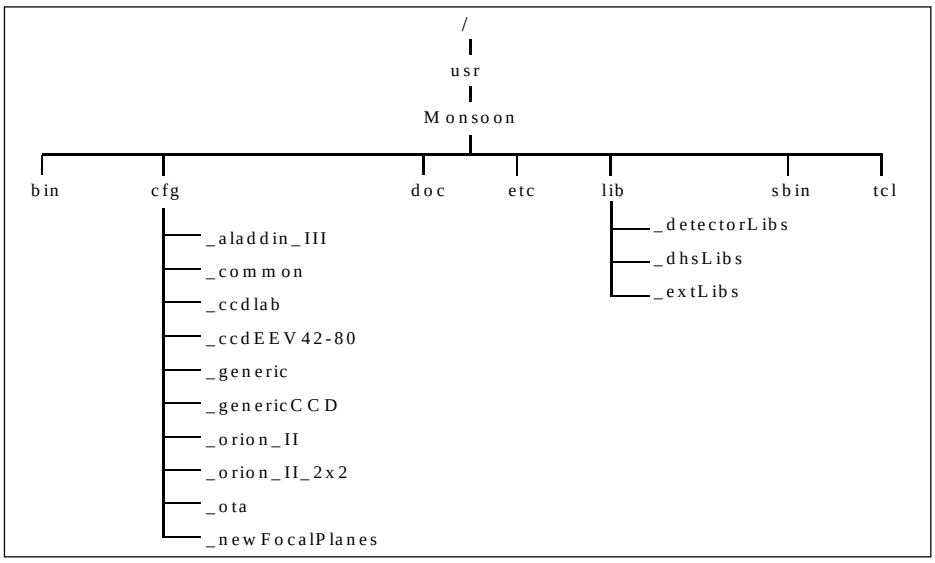
TRNT-AD-08-0001 - TORRENT Software System Description - Nick C. Buchholz (NOAO), Version 1.0, dated: 20090709 NOAO Document TRNT-AD-08-0001-R1.0-SDD



Observatory System Reference Model
Figure 1



System Context Diagram
Figure 2



MONSOON Production Directory Structure
Figure 3

2.0 MONSOON PAN Software Installation Steps

2.1. Binary Installation of the MONSOON PAN Software

1. Following the guidelines in sections 4.1 through 4.3. Install the appropriate Linux version for the current MONSOON release. This is currently CentOS 5.3 however, while the software has not been tested on later versions, there are no known version dependencies.
2. Install the Systran software distribution. Build the applications and build and load the sl240 driver. See section 4.4. Follow the instructions provided in the Systran software documentation.
3. Install all required packages. Refer to section 4.5 for guidelines. These include python2.5 and the required packages plus the cfitsio, TCL8.x, Tk8.x, 'C' libraries. You will need the development packages for Tcl/Tk (as these contain the correct header files). While there are no known version dependencies, cfitsio v3.08, tcl v8.5 and tk v8.5 are currently being used.
4. Modify the shared memory system in accordance with section 4.7.3.
5. Modify the /etc/ld.so.conf file as directed in section 4.7.4 to add the /usr/Monsoon/lib and /opt/sl240/bin directories to the list of directories which are automatically searched by ld for required shared libraries. Run /sbin/ldconfig to make these take effect.
6. Set the boot parameters and processes as directed in section 4.6. That is, set the boot level to 3 or 5 and remove unused process scripts from the rc3.d and rc5.d directories.
7. Create a user account called *monsoon*. Copy the homeDirs tarball and untar in /home/monsoon.
8. Source the new /home/monsoon/.cshrc with:

   source /home/monsoon/.cshrc
9. Decide where to put the binaries. If you have followed the partitioning recommendations in section 4.2, then the binaries will reside in /Monsoon. This directory should be owned by user monsoon with group users. The permissions should be rwxrwxr-x.
10. Do an su to root to create symbolic links from /usr/Monsoon to /Monsoon or wherever the Monsoon software tree will reside. Change the owner and group of the link to monsoon and users:

    ln -sf /Monsoon /usr/Monsoon

    chown monsoon:users /usr/Monsoon
11. Download and untar the binary tarball as monsoon. The current version can be found in ftp://ftp.noao.edu/MONSOON/software/PAN-linux2.6 and will be named (something like) currentBinaryDirs.tgz or yyyymmddBinaryDirs.tgz. The yyyymmdd will be the release date of the version desired. If no such file exists, please ask a member of the MONSOON team to create one for you. The tarball is created with a relative directory naming structure so change directory to /usr or / to do the untar.
12. Edit the .cshrc file in the monsoon home directory to set MONSOON_HOME to /usr/Monsoon and source $(MONSOON_HOME)/etc/monsoon.csh.
13. Do an su root and change directory to /usr or / and:
    a. Change the owner and group of the MONSOON files in **/usr/Monsoon** or **/Monsoon** to monsoon and users. Use the commands:

cd /usr

chown -R monsoon:users Monsoon

b. In Monsoon/bin, do:

chown root shmNuke semNuke shmUNuke semUNuke

c. Change the permissions of shmNuke, semNuke shmUNuke and semUNuke to rwsr-sr-x:

chmod ug+rws shmNuke semNuke shmUNuke semUNuke

14. Exit the su shell and as monsoon create the ssh key as instructed in section 4.9.

15. Execute the command:

setenv MONSOON_HOME /usr/Monsoon

source /usr/Monsoon/etc/monsoon.csh

16. Try the system for Orange systems by entering:

mecStart generic myMachineName localFITS 65

or for Torrent systems enter:

mborg –sysName basicCCD –panName myMachineName –dhsName localFITS

myMachineName should be the name of the PAN obtained from hostname or an alias for that name (eg decapod.tuc.noao.edu or decapod).

17. Create and set up a configuration directory for your focal plane using the guidelines in section 6.0. The systems generic and basciCCD are examples for Orange and Torrent systems respectively

2.2. Source Code Install of the MONSOON PAN Software

1. Follow steps 1 through 10 in section 2.1.

2. Make a link from /MNSN to the development directory location. This will prevent problems with different development directory locations:

ln -sf /MyMonsoonWorkingDir /MNSN

3. Download and untar the workingDirs or cvsRepository tarball. NOAO keeps the main working directory and the cvsRepository in the directories /MNSN/soft_dev and /MNSN/cvsRepository.

The current version can be found in ftp://ftp.noao.edu/MONSOON/software/PAN-linux2.6/ and will be named currentWorkingDirs.tgz or yyyymmddWorkingDirs.tgz. The yyyymmdd will be the release date of the version desired. The tarball is made with a relative directory naming structure so change directory to /MyMonsoonWorkingDir. Do the untar.

If starting from the cvsRepository tarball, decide where to install the repository. This does not have to be on a PAN machine. The software can be built on a compatible machine and distributed to as many PANs as desired. The remaining steps assume the working directory will reside on the PAN. If using a separate development machine, review Appendix VI, “Code Development on a Non-PAN Machine”. Once the cvs repository is untarred, log in as the MONSOON user on the machine containing the working directory:

cd /MyMonsoonWorkingDir

mkdir soft_dev

cd soft_dev

cvs checkout bin cfg doc etc inc lib sbin src tcl

This should result in a set of working directories ready to build new MONSOON software.

4. Type the alias dvlpMonsoon. This should set all the MONSOON environment variables to /MNSN/soft_dev/ etc. It is possible to modify the dvlpMonsoon alias to point directly to a specific working directory and eliminate the /MNSN link.

5. The next group of steps will build the PAN software. At each step, once there are no errors, proceed to the next step. There should be no errors in the build process other than a missing or misplaced file or library. If at any time in this process unresolvable errors are discovered, please contact NOAO to report a problem.

a. Change directory to **$(MONSOON_HOME)/src/Hdw/SystranHdw** and execute:
make everything

Proceed when there are no errors.

b. Change directory to **$(MONSOON_HOME)/src/Hdw/monsoonHdw** and execute:
make everything

Proceed when there are no errors.

c. Change directory to **$(MONSOON_HOME)/src/Hdw/Detectors** and execute:
make DETNAME=generic
make DETNAME=basicCCD

Proceed when there are no errors.

d. Change directory to **$(MONSOON_HOME)/src/Util** and execute:
make everything

Proceed when there are no errors.

e. Change directory to $(MONSOON_HOME)/src/Apps/Tools/uCodeAssembler5.0 and execute:
make everything

Proceed when there are no errors.

f. Change directory to **$(MONSOON_HOME)/src/Apps** and execute:
make everything

Proceed when there are no errors.

6. Do an su root. Change directory to /MyMonsoonWorkingDir/soft_dev

a. In bin, do:

chown root shmNuke semNuke shmUNuke semUNuke

b. Change the permissions of shmNuke, semNuke shmUNuke and semUNuke to rwsr-sr-x

chmod ug+rws shmNuke semNuke shmUNuke semUNuke

7. Exit the su shell and, as monsoon, create and distribute the ssh key as instructed in section 4.9.

8. At this point the working directory distribution can be tested by typing:

mecStart generic panMachine localFITS 65

or

mborg –sysName basicCCD –panName myMachineName –dhsName localFITS

9. If the system works, it is possible to distribute a production version of the system by executing the following commands:

   cd $(MONSOON_HOME)/sbin

   ./distPanSW YourMachineName

   ./cfgDist.sh YourMachineName

10. Test the production version by typing useMonsoon and then

    mecStart generic panMachine /data 65

    or

    mborg –sysName basicCCD –panName myMachineName –dhsName localFITS

11. Once the generic system is tested, create and/or modify other focal planes by creating and modifying a _focalplane directory in $(MONSOON_HOME)/src/Detectors and creating or modifying a _focalplane directory in $(MONSOON_HOME)/cfg. A full description of how to modify or extend MONSOON for additional focal planes is included in the MONSOON software manual and in the Configuring Monsoon Systems Manual. Both of these publications are currently in development.

    Focal plane configurations currently exist for the following:

    generic - a single detector focal plane with no frame processing (loosely based on an aladdin_III detector) for Orange.

    generic_CCD - a single CCD detector focal plane with versions for EEV42-80 and STIS ccd's in one two and four output versions for Orange.

    aladdin_III - a single aladdin_III detector focal plane for Orange.

    basicCCD - a single e2v 44-82 detector for Torrent.

    orion_II - a single orion_II detector focal plane for Orange.

    NEWFIRM - a 2 by 2 array of orion_II detectors focal plane. configured for 2 detectors in each PAN, so a single PAN would look like a 1 by 2 array of detectors for Orange.

    CCD Board Test – a configuration to test MONSOON Orange CCD Acq boards.

    MNSN Board Test – a configuration to test MONSOON Orange MCB and Clock & Bias boards.

    chile1, chile2 - a single e2v 44-82 for CTIO lab development for Torrent.

    chiron - a single e2v231-84 four output detector for Torrent.

    kosmos_e2v, cosmos_e2v - a single e2v 44-82 detector for the Kosmos and Cosmos instruments for Torrent.

    mosaic1 – eight e2v 44-82 detectors for the mosaic instrument for Orange.

    sta1 - a single STA1042 detector for the Bench Spectrograph for Orange.

    sta2 – a single STA1042 detector for the sta2 instrument for Orange.

    manualAfeTest, Calib, tucsonLab - single e2v 44-82 for Tucson lab development and AFE calibration for Torrent.

    whirc – unused.

2.3. Installing the MONSOON Torrent Python Code

MONSOON Torrent systems use a number of python tools. Most of these tools are used by Torrent systems and engineers developing or optimizeing Torrent based focal planes. The **csvEdit** tool was developed specifically to allow the safe editing of MONSOON Orange ".csv" files. The Torrent tools, enable the automatic configuration of systems at run-time, assist the detector or system engineer in creating and optimizing a new focal plane and automate the testing and calibration of Torrent boards. These tools are:

collector – a python program run automatically by the system startup script. This program reads the Torrent ".cfg" files, the EEPROM mirror files and the DHE EEPROMS, to create the ".ini", ".csv" and ".mod" files needed by the pan processes to setup and run the focal plane and take data.

assimilate – The python program that reads the ".vhd" source code files that are used to compile the FPGA code. **assimilate** creates the Torrent_XXXV###.cfg files for each FPGA code module, the EEPROM mirror files for each EEPROM in the DHE and a sysName_ConfigTmplt.csv file for a Torrent system.

sysConfig – A python based engineer's assistant that tracks and documents the decisions the engineer makes while designing a new focal plane. Several auxiliary tools assist in documenting the various parts of the system and tracking the configuration decisions. These are **arrayDesc**, **fclPlnDesc**, **dewarDesc** and **dheDesc.** A version of these tools could be developed for MONSOON Orange systems if desired.

eepStor – a program that allows hand editting of the data stored in the DHE EEPROMs (although this is *not recommended*).

mborg, **borg** – the basic operator response gui tools. The **mborg** is a fully functional GUI that allows the detector engineer near complete freedom to modify voltages and attribute settings, the **borg** is a similar tool but is designed to limit the access to attributes. In general the setting of attributes in the borg is limited to the observation parameters and loading ".mod" files from the front panel. The user can display all of the attributes in the system but cannot access them directly.

csvEdit – a tool that allows the safe editing of MONSOON Orange ".csv" files.

2.3.1. Installation

The Torrent tools should be installed on every PAN computer. If desired the **borg** and **mborg** can also be installed on machines that will be used to access the PAN remotely. There are also a number of auxiliary files that must be installed on the PAN to successfully use the tools:

1. – First insure that python 2.5 is installed. NOAO installs this in /usr/local/[lib|bin]. The Torrent python tools have been developed and tested with python2.5. It is likely they will also work with python2.6 and python2.7. They will almost surely fail on python 3.0 or later.
2. – cd to /usr/local and untar the Torrent Python packages, do this as root or insure that the directories: /usr/local/bin, /usr/local/lib/python2.5/site-packages are writable by the user doing the install.
3. – If you wish to allow modifications to the basic Torrent tools make the directories above accessable to the *monsoon* user on the PAN machine.

2.3.2. Auxiliary Files

Several directories and files are used by the Torrent tools when running a system these are installed in the ${MONSOON_HOME}/cfg directory. They include the following:

._testFiles - this directory contains all of the files describing the test routines for testing Torrent boards and systems. They are read and used by the testFrame, PsmTest, MezTest and Calibration

objects that control and provide the functionality for the Torrent testing procedures. Provisions exist for adding LCBTest, AFETest and TsmTest objects to the testing but these have been postponed.

__CfgFiles – The TORRENT ".cfg" files for active FPGA versions are stored here. These files are used by the **collector** at runtime to determine the layout of each EEPROM in the DHE. The files includes are: Torrent_CCD#V###.cfg, Torrent_CCD1V###.cfg, Torrent_CCD2V###.cfg, Torrent_CFGV###.cfg, Torrent_CLKV###.cfg, Torrent_LCBV###.cfg, Torrent_PIXV###.cfg, Torrent_PSMV###.cfg, Torrent_SFTWV###.cfg, Torrent_SYSV###.cfg and Torrent_TSMV###.cfg

__Common – the directory holding the system database directory and files.

__DHEs - The description files for the generic detector head electronics units currently described. The only active file is the generic2AFE_trntDhe.dsc file.

__Detectors - descriptions of generic detectors that can be included in specific focal planes.

__Dewars - descriptions of generic Dewars that can be included in specific focal planes.

3.0 Using MONSOON/Torrent Scripts and Programs

A great deal of the MONSOON/Torrent functionality is provided in the shell scripts and programs that are provided with the PAN computer Appendix II outline some of the scripts used in configuring and running MONSOON/Torrent systems. This sections will discuss the other scripts provided with the systems.

3.1. Process and File Clean-up Scripts

fc0 – (alias sl_mon 0 crlf) clears the the Systran fiber link FIFO. The output will be something like this

SL_MON 0: 0 bytes or
SL_MON: 0: 131 Bytes 1048 bytes left

If the output that looks like this:

SL_MON: 0: 131,068 Bytes 1048892 bytes left or this
SL_MON: 0: 131,068 Bytes 1048892 bytes left (10492 Phanton)

it indicates that the DHE is powered down or the fiber is disconnected or the Systran driver is hosed. Use the slmon routine to fix the problem (see below).

fs0 – (ailas sl_mon 0) displays the status of the Systran Fiberlink (see also slmon). The output will look something like this on PCI based machines:

```
FibreXtreme (SL) Monitor (sl_mon) rev. 3.02 (2003/10/06)
Driver:  rev. b2-945465:776765 for Linux with API rev. 2.1
Hardware: unit/bus/slot 0/14/4 - SL100x (D64) Firm. 1C.6a (5C.6a) for 3.3V
PCI
    Link Control Register  (CSR 0x08) = 0x37
    Link Status Register  (CSR 0x0c) = 0x100   Link is DOWN
    FPDP Flags Register   (CSR 0x10) = 0x65800  NR.D.P2.P1.S: i=11000 o=00000
    FIFO Threshold Register (CSR 0x14) = 0x24004f  Int.thr. = 0x0
        Data count          = 0x10013c (1,048,892) bytes
    Link (and other) Errors      = 0
Configurable parameters:
    Loop Configuration:  0  (Point-to-Point)
    Max Timeout:     600000 (6000000 ms)
    Flow Control:     0 (NO)  Halt on link error:      1 (YES)
    CRC generate/check:  1 (YES)  Allow Queuing on link error: 1 (YES)
    Data Byte Swapping:  0 (NO)  Receive SYNC with DVALID:   0 (NO)
```

On PCIe based machines the output will be something like this:

```
NSL Monitor (nslmon) rev. 1.03 (2009/09/22)
Driver 1.1lnx-D31:766:853034 - API 1.0-674061 - Firmware 4.34
```

```
NSL unit 0 bus 3 slot 0 chan 0 - HW V1PC SFPDP - PCIe-x8
Memory 256MB - Link 1.06 Gbps - Bus 125 MHz - DDR 166 MHz
======================== CONTROL/STATUS ================================
Link    : 0  Signal   : 0  FPDP in  : 0x8 (NRDY       )
Laser   : 1  Flow Ctrl : 1  FPDP out  : 0x0 (         )
Receive  : 1  CRC Enable : 1  Remote Tx : 1
Retransmit : 0  DMA64   : 0  Frame Size : 512
Transmit  : 1  Byte Swap : 0  Gap    : 1
E-Wrap   : 0  Word Swap : 0  Max Timeout: 6000
StopLinkErr: 1  Test Mode : 0  LSF    : ...LD..I....E
========================== STATISTICS ====================================
TX Bytes  : 0      (     +0)  MB/s: 0.00  ( +0.00)
RX Bytes  : 0      (     +0)  MB/s: 0.00  ( +0.00)
Interrupts : 0      (     +0)  I/s : 0.00  ( +0.00)
Link Down : 0      (     +0)  E/s : 0.00  ( +0.00)
Link Err  : 0      (     +0)  E/s : 0.00  ( +0.00)
Decoder Err: 3346120253 (     +0)  E/s : 0.00  ( +0.00)
```

The `Link` entry will be "`Link is UP`" or "`link  : 1`" if the system is ready to go.

ff0 – (alias sl_mon 0 fpdp) allows the user to reset the DHE link from the PAN shell command line. `ff0 6` puts the Systran fiber signals into the reset DHE state. `ff0 0` returns the fiber to normal operations. The use of this sequence will leave random data on the link which must be cleared and it will leave the DHE in the asyncResp state that indicates the DHE was reset and the PAN computer has not acknowledged that reset. The DHE will respond to all commands sent to it with an asyncMsg until the PAN sends an asyncResp message to the DHE (seel also sls and slr).

slmon, nslmon – (scripts in /Monsoon/bin/[n]slmon) perform a number of functions on the Systran Fiber link that require root access to the Linux machine:

```
/Monsoon/bin/[n]slmon: {clrf|load|reload|reset|status|unload} <pan>\n
Eg: /Monsoon/bin/slmon clrf  140.252.31.172
Eg: /Monsoon/bin/slmon load  140.252.31.172
Eg: /Monsoon/bin/slmon reload 140.252.31.172
Eg: /Monsoon/bin/slmon reset 140.252.31.172
Eg: /Monsoon/bin/slmon status 140.252.31.172
Eg: /Monsoon/bin/slmon unload 140.252.31.172
```

panTools – (script in /Monsoon/bin/panTools) performs a number of useful cleanup functions on the PAN processes and configuration (also see panTools under section 3.2)

```
Usage: /Monsoon/bin/panTools {nuke|ping|rmlog|status|procs} <pan>\n
Eg: /Monsoon/bin/panTools lnkclr 140.252.31.172
Eg: /Monsoon/bin/panTools nuke  140.252.31.172
Eg: /Monsoon/bin/panTools rmlog 140.252.31.172
```

semNuke, semUNuke – (program and script) remove unused semaphores from the Linux system. semUNuke only removes semaphores belonging to the current user. semNuke removes items based on the uid entered on the command line.

shmNuke, shmUNuke – (program and script) remove unused shared memory segments from the Linux system. shmUNuke only removes segments belonging to the current user. shmNuke removes items based on the uid entered on the command line.

3.2. Information scripts

SemShow – (script) show information on currently allocated semaphores.

shmShow – (script) show information on currently allocated shared memory segments.

panTools – (script in /Monsoon/bin/panTools) performs a number of useful information functions on the PAN Processes (also see panTools under section 3.1).

```
Usage: /Monsoon/bin/panTools {nuke|ping|rmlog|status|procs} <pan>\n
```

```
Eg: /Monsoon/bin/panTools ping  140.252.31.172
Eg: /Monsoon/bin/panTools status 140.252.31.172
Eg: /Monsoon/bin/panTools procs 140.252.31.172
```

3.3. Fiber Link Scripts

sls – (program) sends a set of 10 safe commands to the DHE displays a list of the commands sent or times out with a return value of 7 (4 on PCIe systems):

```
0: ret = 0 buffer=0xc0010001
1: ret = 0 buffer=0x80010100
2: ret = 0 buffer=0x00004242
3: ret = 0 buffer=0x4001fffe
4: ret = 0 buffer=0x4001ffff
5: ret = 0 buffer=0xc0020002
6: ret = 0 buffer=0xc0030003
7: ret = 0 buffer=0x80010000
8: ret = 0 buffer=0x30302a2a
9: ret = 0 buffer=0x40010100
10: ret = 0 buffer=0x40010000
11: ret = 0 buffer=0xc0040004
```

Or on failure:

```
0: ret = 7 buffer=0xc0010001
```

slr – (program) reads data from the Systran fiber link until the link is empty. If all is well with the system and slr is run immediately after sls the output will look like this:

```
1: ret = 0 buffer=0xc0010001
2: ret = 0 buffer=0x80010100
3: ret = 0 buffer=0x00004242
4: ret = 0 buffer=0x4001fffe
5: ret = 0 buffer=0x000000c9
6: ret = 0 buffer=0x4001ffff
7: ret = 0 buffer=0x000000de
8: ret = 0 buffer=0xc0020002
9: ret = 0 buffer=0xc0030003
10: ret = 0 buffer=0x80010000
11: ret = 0 buffer=0x30302a2a
12: ret = 0 buffer=0x40010100
13: ret = 0 buffer=0x00000000
14: ret = 0 buffer=0x40010000
15: ret = 0 buffer=0x00000000
16: ret = 0 buffer=0xc0040004
17: ret = 7 buffer=0xc0040004
```

If the link is down slr will return

```
1: ret = 7 buffer=0x00000000
```

In rare cases when the DHE is in a strange state slr can appear to run forever, use `^C' to halt the program.

slmon, nslmon – scripts see slmon,nslmon under section 3.1

3.4. Housekeeping Scripts

These are mainly unused: pchkcfg.sh, pclean.sh, pdiff.sh, pincs.sh, pmd5.sh, pmkmf.sh, psed.sh, psync.sh.

3.5. Configuration Control Scripts

mkNewCfg.sh – (script for MONSOON Orange) creates a new set of config files from an existing set. Called as:

```
mkNewCfg.sh oldName newName
```

updBBCfgArchive – (script) copies a set of production config files from the local PAN to the big-boy configuration archive location /MNSN/soft_dev/cfg. Called as:

```
updBBCfgArchive cfgName
```

updCfgOnMachine – (script) copies a production configuration file set to the production directory on the destination machine. Called as:

```
updCfgOnMachine cfgName destMachine
```

updConfig – (script) copies a z set of config files to the production directory on dest Machine (/Monsoon/cfg): Called as:

```
updConfig cfgName destMachine
```

updNEWFIRM – (script) updates the production directory on on NEWFIRM PAN from the production directory on the other NEWFIRM PAN called without arguments.

3.6. Code Distribution Scripts

mkTestPkg – (script) bundles images from a detector test and ships it to a destination directory:

```
mkTestpkg testDate(msd) srcDir destDir
```

mkWDtarball – (script) tars all files in the MONSOON working/development directory into a tar file. This includes source code, makefiles and include files. The final product is a compressed tarball file that when untar-ed creates a complete working directory/environment that can be used for further development of the PAN code.

mkPDtarball – (script) tars all files in the MONSOON production directory into a tar file. This inculdes binary files, libraries and scripts. The final product is a compressed tarball file that when untar-ed creates a complete production directory/environment that can be used to run a focalplane from the PAN.

mkPYtarball – (script) tars all files required by the Torrent python code into a tar file. This includes startup code, objects and auxiliary routines needed to use the Torrent python tools.

DistTrntSys – (script) run from a Torrent archive machine or functional PAN, the script updates or creates all of the Torrent files needed to run torrent systems.

allDist.sh – (script) run from a MONSOON/Torrent archive machine or PAN the script updates or creates all of the files (except the Tcl and python files) needed to run a MONSOON/Torrent system on the machine.

cfgDist.sh – (script) run from a MONSOON/Torrent archive machine or PAN, the script updates or creates all of the active configurations on the destination machine.

distPanSW – (script) run from a MONSOON/Torrent PAN machine, the script updates or creates a set of production code on the destination machine.

4.0 Configuring a MONSOON PAN or Client System

4.1. Operating System Considerations

At this time MONSOON runs on LINUX systems running Centos 5.3 or later. The MONSOON software does not contain any known operating system dependencies and should run on any flavor of LINUX system for which a Systran SL100/SL240 driver exists. However, be aware that the gcc 4.x compiler has a known bug when creating shared libraries that badly effect the MONSOON Torrent run-time code. Therefore, we use the 3.x compiler series on all our machines.

4.2. Disk Partitioning

The partitioning of a MONSOON PAN is not critical, in that, any reasonable partitioning scheme that will run LINUX will allow the user to run MONSOON as well. NOAO uses the following partitioning on its PAN systems. The configured PANs contain two 32-Gb disks. One disk is used for the system and the other is reserved as a single 32-Gb /data partition. The PAN system disk is partitioned as follows:

File system	1M-blocks	Comment
/	17512	root partition includes /usr, /opt, /var, /etc
/boot	494	partition for boot code and kernel images
/MONSOON	4031	MONSOON binary distribution partition
/home	4031	home directories for users normally just MONSOON on PAN
/src	4031	area for new local software package development/building
swap partition	4031	for a 2Gb memory machine with no file system mount point

4.3. Directory Structure

The current version of the MONSOON software assumes that an environment variable "MONSOON_HOME" contains the path to the base of the MONSOON software tree. On NOAO systems this is "/usr/Monsoon" or "/Monsoon" for the distributed production version and "/MNSN/soft_dev" for the development working directory. In fact, symbolic links are used to ensure that all systems can use /usr/ Monsoon as the base of the MONSOON production directory tree. See Figure 3 for the full structure.

In addition, the CVS repository for MONSOON development is in "/MNSN/cvsRepository" located on "big-boy.tuc.noao.edu".

On systems that will run an engineering or science client only, the /usr/Monsoon/tcl directory is needed. This directory may be kept anywhere on the system as long as MONSOON_HOME and MONSOON_TCL point to the correct location.

4.4. Systran Driver and Utilities

The NOAO MONSOON PANs have the Systran driver installed in the directory /opt/sl240. Follow the procedures in the Systran software documentation and load the code from the Systran CD-ROM. Alternatively the drivers and applications can be downloaded in RedHat 9.0 systems by getting the sl240All.tar file from:

http://ftp.noao.edu/MONSOON/software/sl240All.tar

Then untar the sl240All tar file as follows:

cd /opt

tar xvf /temp/sl240All.tar

The sl240 driver will have to be built and installed and /etc/rc.local modified to load the sl240 driver at boot time. See the rc.local file in the systemFiles.tar file. Additionally, occasionally the Systran utilities, sl_mon, sltp, etc will require rebuilding on the local machine as follows:

cd /opt/sl240/examples

make -f apps.linux.mak

Once this is done, change directory to the /opt/sl240/driver directory and as root, type:

./fxinst

This will load the driver and make it accessible. To test the driver type:

/opt/sl240/bin/sl_mon 0

A status screen will be displayed if the driver loaded properly. Note that the dirver must be made with the compiler that the system was created with. Invariably, this is now gcc 4.x so you will have to swap to the new compiler to build the Systran driver before reverting back to 3.x to build or re-build the any of the MONSOON software.

4.5. Required Libraries and Tools

MONSOON is built and maintained using xemacs 21.7, gcc 3.46 and cvs 1.11.

To start the MONSOON PAN processes, the PAN system must have the LINUX standard libraries, the MONSOON libraries and the following external libraries located on the LD_LIBRARY_PATH:

libcfitsio.so, libfitstcl.so, libfxsl.a, libfxslapi.so, libkcore.a

See section 4.7.4 for a description of how this can be accomplished.

To start and run the **mec** interface, a PAN or remote machine must have the following libraries and packages installed: libtcl8.3.so, libtk8.3.so, tcl8.3, tk8.3 and BWidget-1.7.0.

4.6. Boot Time Run-Level Processes

PAN systems should be as “stripped down” as possible. The PAN can boot into run-level 3 or 5, All unneeded processes should be removed from the startup files and rc3.d and rc5.d directories. These unneeded processes include cups, kudzu, sendmail, cron, at, canna (unless the Japanese font set is required), xfs, all but one getty, samba (unless the PAN disks will be viewed on a Windows machine). The processes for automatically sensing CDROM or DVD disk changes and the sound system drivers can also be shut down.

4.7. PAN System Setup

PAN systems require some modifications from the standard LINUX setup.

4.7.1. Shells

The MONSOON user is set up to use /bin/tcsh as the login shell. Any login shell can be used at the cost of having to redo aliases and start-up files and such. Most MONSOON shell scripts are set to use /bin/sh but they can be rewritten to suit the local preferences.

4.7.2. Secure Shell Daemon

The MONSOON system requires ssh (secure shell) for remote startup. “telnet” and “rlogin” are disabled by default on MONSOON PANs. A remote machine ssh daemon must be set up to properly connect to a MONSOON PAN. The file /etc/ssh/ssh_config should have the following two lines near the end:

HOST *

ForwardX11 yes

Some additional setup will be required on a per user basis on the client. This is outlined in section 4.9.

4.7.3. Shared Memory

The MONSOON PAN software uses a system of shared memory buffers to store data and pass information between the various processes. A minimum of two image buffers is also needed. These image buffers must be large enough to store a complete image of the focal plane in memory. To allocate large shared memory buffers, the Linux kernel must be configured at boot time for the large buffers. This is done by adding the following two lines to the end of the file /etc/sysctl.conf:

 # set maximum shared memory segment

kernel.shmmax = 167772160

The number on the second line is the value of the largest single image memory buffer that will be required. Thus if the focal plane is made up of four detectors each with four outputs, with active image areas of 4096 x 4096 and a column pre-scan of 8 and a column post-scan of 16 and a row post-scan of 12 is being used, then the value of shmmax must be at least:

4 * ((4096 + (2 * 8) + (2 * 16)) * (4096 + (2 * 12))) * number of bytes per pixel (2 or 4 in general)

which is 136586240 (2 bytes/pixel) or 273172480 (4 bytes/pixel). Note that this number is the maximum size of an image in memory. If in the example above, a column post-scan of 32 pixels is used, the system must be reconfigured to accommodate that value. Also at this time the size of the PAN memory must be capable of dealing with two entire images in memory simultaneously.

4.7.4. Setting The Load Path

The shared libraries required for the MONSOON PAN software are located in several places on the disk. To ensure that the programs search the correct directories to find the libraries, the configuration of the “ld” loader is modified by completing the following steps:

1. Log in to the PAN machine as root.
2. Go to the /etc directory

   cd /etc
3. Edit the file ld.so.conf with a chosen editor and add the following three lines to the file:

   /usr/local/lib

   /opt/sl240/bin

   /usr/Monsoon/lib
4. Update the ld.so.cache file by running the /sbin/ldconfig program to rebuild the cache.

   /sbin/ldconfig -v
5. If there is still a problem of missing libraries when the PAN processes are started, one or more of the required files may be missing or in the wrong place. Enter:

   cd /usr/Monsoon/bin

   ldd panDaemon

This will display a list of all the libraries required by *panDaemon* and their locations. Look for the ones that are missing and provide it by either rebuilding that library or down loading a version off the web. Repeat the last command for *panCapture*, *panProcAlg* and *panSaver*. The most common missing file is probably the cfitsio library which can be down loaded from the following web site: http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html. Follow the instructions provided and build and install the library.The compiled library is in /usr/lib/libcfitsio.so.

4.8. Client System Setup

Client systems that only run the MONSOON Engineering Console (**mec**) or science clients may require some modifications from the standard LINUX setup. These are minor and outlined below.

4.8.1. Shells

The MONSOON user is set up to use /bin/tcsh as the login shell. Any login shell can be used at the cost of having to redo aliases, and start-up files, etc. Most MONSOON shell scripts are set to use /bin/sh. Again, they can be rewritten to suit the local preferences.

4.8.2. Secure Shell Daemon

MONSOON systems require ssh (secure shell) for remote startup. “telnet” and “rlogin” are disabled by default on MONSOON PANs. A remote machine ssh daemon must be set up to properly connect to a MONSOON PAN. The file /etc/ssh/ssh_config should have the following two lines near the end:

HOST *

ForwardX11 yes

Some additional setup will be required on a per user basis on the client. This is outlined in section 4.9.

4.8.3. Tcl/Tk

This section can be ignored if the MONSOON Engineering Console (**mec**) is not used.

The standard setup for Tcl/Tk on Linux systems changed between version RedHat 7.3 and RedHat 9.0. Earlier systems put the Tcl/Tk libraries in /usr/lib/. Newer systems use /usr/share/. To ensure compatibility, MONSOON currently uses /usr/lib/tcl8.3 as its base library path. On newer systems, make a link to /usr/ share/tcl8.3 with the following command:

ln -sf ../share/tcl8.3 /usr/lib/tcl8.3

Connect in tk8.3 as well with the command:

ln -sf ../share/tcl8.3 /usr/lib/tk8.3

MONSOON uses some non-standard packages when running the **mec** interface, namely BWidget. To install Bwidget as root, untar the BWidget tar file into /usr/lib/tcl8.3, or whatever the most recently installed version of Tcl/Tk might be.

4.9. Remote Startup

To properly start a MONSOON PAN system from any machine, the monsoon user’s ~/.ssh directory on the PAN must be setup to accept ssh connections without requesting a password. This is accomplished as follows:

1 On any remote machine any user that wishes to start up a MONSOON PAN must create a public key to be used for the ssh login. The command is:

ssh-keygen – t dsa.

Do not enter a pass phrase. Use <return> instead. The output will look similar to the following:

ssh-keygen -t dsa

Generating public/private dsa key pair.
Enter file in which to save the key (/home/userName/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/userName/.ssh/id_dsa.

Your public key has been saved in /home/userName/.ssh/id_dsa.pub.
he key fingerprint is:
85:66:d0:19:d7:2a:67:46:63:b2:e9:2c:66:3a:5e:bc userName@machine

2. Copy the ~/.ssh/id_dsa.pub to each system that requires access:

   scp ~/.ssh/id_dsa.pub
   monsoon@panMachine:/home/monsoon/.ssh/id_dsa.pub.machine

3. Now use ssh to login to the PAN and add the recently copied key file to the authorized_keys2 file on the PAN machine with the following:

   ssh monsoon@panMachine

   The authenticity of host 'panMachine (140.252.99.254)' can't be established.
   RSA key fingerprint is 86:e7:18:7c:7d:f2:d1:4a:81:58:03:ed:5d:8a:d0:17.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added 'panMachine,140.252.99.254' (RSA) to the list of known hosts.
   monsoon@panMachine's password: *********

   cd .ssh

   cat id_dsa.pub.machine >> authorized_keys2

4. Repeat Steps 1 through 3 for each machine and user who will want to access the MONSOON PAN system.

The directories /home/monsoon and /home/monsoon/.ssh and the file /home/monsoon/.ssh/authorized_keys2 must be writable only by the owner of the directories and files. Use the following command as monsoon:

chmod go-w ~ ~/.ssh ~/.ssh/authorized_keys2

5. To test the ssh installation do the following:

   ssh monsoon@panMachine

This may ask if you want to continue connecting but it should not request a password. If it requests a password, check the permissions on the files mentioned in Step 4. If they are correct, check the ssh configuration to be sure it is not set to “always request a password”.

4.10. Using a MONSOON System

There are three ways to use a MONSOON system. First, log in to the MONSOON PAN as monsoon and run the PAN processes directly by typing ppx commands directly into the *panDaemon* process command line interface. This is neither convenient nor safe. It is difficult to remember and type all of the commands required to set up and use a MONSOON system.

The second and third methods involve using a science or engineering client. These client systems may be run on the PAN or remotely. Most science clients will most likely be run on a machine other than the PAN. The engineering clients (**mec** or **mborg**) can be run on the PAN or from a remote machine. See section 4.9 for the requirements for remote start-up.

4.10.1. Running the panDaemon

The PAN processes can be run directly for hardware or software testing and debugging. In particular *panDaemon* can be run alone or as part of the suite of PAN processes used to take data. Running the processes directly requires the user to remember and type the commands needed to run the PAN system. The recommended procedure is to run the *panDaemon* alone only to make

brief tests of the configuration files or individual hardware/software changes relating to the attribute setting routines.

4.10.2. Running the MONSOON Engineering Console (mec)

Running the **mec** uses the same steps regardless of whether it is being run from a remote machine or the PAN. The command to start the system has the following syntax:

Syntax: mecStart {focalPlaneName} {panName} [dhs machine name | localFITS]

In the example below the user buchholz starts the orion_II_2x2 or NEWFIRM focal plane software on nfpan-a, one of the NEWFIRM PANs, from the remote machine decapod. The default data handling is set to local FITS that writes images in FITS format to the local disk. If the site is running a DHS system, this would be replaced with the name of the machine running the DHS server. The debug level is set to 65. Note that the higher the debug level the more debugging text is generated. Levels above 100 become intolerable and should only be used when tracking a specific low-level bug. A debug level of 66 will lock in the xterms on the screen even if an error caused the process to shut down. Levels above 1000 start the systems in simulation mode.

buchholz@decapod> mecStart orion_II_2x2 nfpan-a localFITS

The steps required to start the system are outlined as follows:

1. Log in to the machine being used as a user who has been set up according to section 4.9.
2. Type the mecStart command syntax outlined previously. Four xterms and the **mec** GUI should pop up. If this does not occur, some error messages may appear on the terminal where the mecStart command was typed. The four xterms can also be checked to see if there are error messages.
3. Follow the steps outlined on the **mec** main screen. Check the hostname and port entries for correctness and then press the Step 3. Connect button. If this is the first connection since a power down or reset of the DHE, press the AsyncResp button at the bottom center of the **mec** GUI. Generally pressing the Reset and Asyncresp buttons in sequence after a restart is safe and ensures consistent startup conditions.
4. Continue to follow the instructions on the **mec** GUI main screen and press the System Setup button. The configuration file screen will appear. Follow the instructions on that screen.

From this point on, the attributes page screen that popped up after the connection was made can be reviewed for attributes changes and data retrieval.

4.10.3. Simulation Mode

The DHE and PAN sometimes have communication problems. When this happens the *panDaemon* and other PAN processes come up in simulation mode. If the system is in simulation mode, every command is successful. A command and response on the **mec** screen will then look like this:

s> ppxSetAVP AttributeName=3.4

r> OK[SIM]: AttributeName set to 3.4

Note the [SIM]. To exit simulation mode at the current time, the system must be shut down and restarted. Common causes for going into simulation mode are:

1. A Systran fiber link that is down. The DHE power may be off. A fiber or fibers between the PAN and DHE may be broken or disconnected. The Systran boards in the DHE or PAN may be faulty (unlikely).

2. A DHE being held in reset mode by the PIO lines. Use sl_mon or ff0 to set the pio lines to 0x00.

3. A locked up Systran driver. This may require a shutdown and restart of the Systran driver or a reboot of the PAN machine.

4.10.4. Microcode Sequencer

The control signal timing for the focal plane is generated by a microcoded sequencer in the DHE. This sequencer is disabled while attributes are being changed and must be enabled before the system can generate image data. The *startExp* button and the attribute screens change color to indicate the system state. If the *startExp* button is red (attribute page list has a green background), the attributes can be changed but a *startExp* command will time out because no data will be generated by the DHE. If the *startExp* button is green (attribute page list has a red background), the attributes cannot be changed and a *startExp* command should complete correctly.

The bug that caused this behavior has been fixed and the sequencer may be left enabled while changing attributes. The sequencer should still be disabled when loading a microcode sequence.

5.0 Important Orange Attributes

The following are attributes whose required settings are not obvious because of the generic nature of the interface and may only be needed in Orange systems.

procMode - short for processing mode. For IR systems a 0 means process the frames. A 1 means send all frames to storage. In CCD systems this attribute is probably not used.

seqPreScaler - slows the sequencer, so for most systems this should be a 1 giving 100 ns resolution for the sequencer clock states.

expVector - determines the code the sequencer will run for the exposure. This should be 1 for default data mode.

imagefile - a string prefix prepended to each local image file.

imageDir - the full path to the directory on local disk where the images will be placed.

CLK0_BIAS[] - array attributes are created at configuration time. The user can set all of the attributes in an array by setting the DHE name variable corresponding to the PAN name. To get a DHE name, type ppxDump clkBias[] into the **mec** command line and read the DHE name from the screen (in this case CLK0_BIAS[]). Setting this name to a value will set all of the attributes in the array to the value, even those that have been aliased:

ppxSetAVP CLK0_BIAS[]=0.0

will set all of the bias voltages on the clock and bias board to zero.

6.0 Setting Up MONSOON Focal Plane Configuration Files

A MONSOON configuration is made up of several files that describe the focal plane and DHE hardware to be controlled by the system. The main configuration files are the focalPlane.arr file, the focalPlane.csv file and the focalPlane_guiCategories.txt file. These three files are read into the system once at system start-up and control the functionality and shared memory configuration of the PAN system. Several set-up files are loaded into the system when requested by the user: focalPlane.ini and focalPlane_DefaultSetup.mod set up the system to an initial state and default mode for this focal plane and PAN DHE pair. A sequencer microcode file is required to load the correct waveforms and timing patterns into the sequencer to run and read out the focal plane.

Finally, several default, (usually empty) mode files are provided to set up different detector, exposure and data processing modes as required. Some of these modes can be created and modified by the user during an observation run

6.1. csv File structure

The MONSOON configuration file is a .csv file originally constructed from an Excel document. Unfortunately the complexity of the configuration process rapidly out-stripped the ability of Excel to handle it and the .csv file is now created by hand. Future plans are for development of a semi-automated system for generating a MONSOON configuration.

The configuration and setup of a MONSOON system is currently done using a text editor to modify the focal plane .csv file. An explanation of the .csv file entries is provided in Appendix III. A full explanation of the configuration process for new focal planes is given in "MONSOON User's Guide" which is in development. Appendix III will give a quick explanation of the .csv file and its structure and content.

6.1.1. csv File Block Structure

The .csv configuration files consist of configuration lines that describe each of the attributes in a MONSOON system. The lines are arranged in blocks to ease the editing task.

Block 1 -	MCB Board Attribute Description	
Block 2 -	Sequencer Attribute Description	
Block 3 -	IRAcq board 1 thru N Descriptions	-or-
	CCDAcq Board 1 thru N Descriptions	
Block 4-	Software Attributes and Client Level Attribute Aliases	

The organization may be changed as desired for your system. For example the orion_II_2x2 configuration has four IRAcq boards and the configuration is blocked partially by attributes within block 3 as in:

attriubteABd1,BD1Attribute,...
attriubteABd2,BD2Attribute,...
attriubteABd3,BD3Attribute,...
attriubteABd4,BD4Attribute,...

6.1.2. csv File Line Structure

The lines in the .csv file contain 16(18) columns of data separated by commas. Empty fields must contain at least a single space character and there can be *no* embedded space characters in any field except the help field. Use the '_' to aid readability. Each field can contain up to 160 characters, and the entire line may not exceed 1024 characters. In actuality 160 characters for an entire line has not been exceeded in testing. The fields in the configuration line are as follows:

PAN/Client Attribute Name - the attribute as it will be known to the average user of a MONSOON system. Familiar names like *IntTime*, *binning*, *digitalAvgs*, *VddUc* or *detBias* are included here.

DHE Attribute Name - the attribute name from the board function point of view, that is, the name given to the attribute or function by the system hardware engineer. Names like CLKBD1_BIAS, MCB_CONTROL_REG, SEQ_LOOP_REG or ACQBD1_VOFFSET[25] are used here.

DHE Hardware Base Address - this field contains the 32-bit hex base address of the function in the DHE. This address is broken into two parts with bits 23 thru 16 giving the board address. This is given as a bit field with a one in the bit corresponding to the slot the board resides in (slot 0 is bit 16). At the current time the software will not accept a multiple board address for an address. The

lower 16 bits contain the address on the board of the attribute or the first attribute in an array of attributes. These two parts may soon be divided into two separate fields.

Number of elements - the number of elements in the attribute array. Enter 1 for single value attributes, N, where N is the number of elements for arrays of identical attributes. There is no limit on the size of N except that the total number of attribute slots in the entire system is restricted to 4000 at this time.

Creg - a 32-bit control register used by the software for various purposes. In particular, the upper byte is used to group attributes for GUI interfaces the other uses are outlined in Appendix III.iii.

Set Method - contains the symbolic name for the method used to set the attribute. This is the method different hardware procedures can used to set individual attributes. The methods are compiled in the *libdetCmnds.so* library for each focal plane type. See Appendix III.iv for a list of all available set methods and what they do.

Read Method - contains the symbolic name for the method used to read the attribute. This is the way that different hardware procedures can be used to read individual attributes. The methods are compiled in to the *libdetCmnds.so* library for each focal plane type. See Appendix III.iv for a list of all available read methods and what they do.

PANType - the data type of the attribute as stored in the PAN. Currently all PAN data types are represented as either doubles (FLOAT) or strings (STRING). See Appendix III.v for the complete list of available data types.

NOTE: The current configuration software assumes that a PAN type which is not STRING is a FLOAT represented by a 'C' double internally.

DHEType - the data type of the attribute used in the DHE. See Appendix III.v for the complete list of available data types.

Coef1 - The first coefficient (slope) in use in the conversion function.

Coef2 - The second coefficient (intercept) in use in the conversion function.

Function ID - the name of the function used to convert from PAN values to DHE values and from DHE values to PAN values. This usually converts from some range of values (see Min. Value and Max. Value) to a bit pattern in the DHE or from a bit pattern in the DHE to a value in the PAN. A list of the available conversion methods is available in Appendix III.vi.

Min. Value - the minimum value the attribute may take on.

Max Value - the maximum value the attribute may take on. This field is also currently used as the mask pattern for the RDMSKWRT set and read methods for DHE data. However, this will change shortly as another field to contain the mask value is being added. See Mask Value.

Mask Value - NOT CURRENTLY implemented. See Max Value. This field will be added to the configurations at a later date.

Attrib. Units Name - The name of the units of the attribute. (volts, bit_Field, Flag, etc.).

HelpText - consists of the remainder of the text on the line after the comma ending the Attrib. Units Name field. If the Help is to contain a ',' the entire help text should be enclosed in "quotes, just like this."

6.2. Example Configuration Entries

These examples will appear on a single line terminated by a new line in the actual .csv file.

```
intTime,MCB_SEQITR,0x0010000,1,0x02000000,SIMPLE,SIMPLE,FLOAT,ULONG,1000,0,LINEAR,0,4294967
.296,SECONDS,Required integration. 1 millisec resolution
mcbSysClkEnables,MCB_CLKENABLE,0x0010100,1,0x05000040,SIMPLE,SIMPLE,FLOAT,UINT,1,0,LINEAR,0
,0xFFFF,FLAGS,Bit register to enable SYSCLK signal slots
mcbEnClk2,MCB_ENCLK2,0x0010100,1,0x05000040,RDMSKWRT,RDMSKWRT,FLOAT,UINT,2,0,LINEAR,0,1,FLA
GS,Defined to enable/ disable SYSCLK to slot 2
```

Appendix I MONSOON User Home Directory Files

The monsoon user on NOAO's PAN systems is set up to use the tcsh shell. If an organization is more comfortable with another shell, it is possible to reconfigure by modifying the startup scripts for the other shell.

Appendix I.1 Example .login File

```
# .login file for monsoon on lechat
/bin/stty crt kill ^U intr ^C
unset noglob
set cdpath=~
set time=120
set history=100 savehist=20
set histsave=50
set filec
set ignoreeof
umask 02
setenv LC_ALL C
set noeof
unset autologout
set prompt="%m <%t> {%h} "
setenv SHELL /bin/tcsh
```

Appendix I.2 Example cshrc File

```
# .cshrc file for monsoon on lechat
setenv LC_ALL C
set ignoreeof
alias addenv 'if (:${\!:1}\: !~ *:\!{:2}\:*) setenv \!:1 ${\!:1}\:\!:2'

et path=(. ~/bin /bin /usr/bin /sbin /usr/X11R6/bin )
setenv MANPATH /usr/man
addenv MANPATH /usr/X11R6/man/
umask 2
set noeof
set prompt="`whoami`@%m <%t> {%h}"
set filec
setenv CVSROOT /MNSN/cvsRepository
addenv PATH /opt/sl240/bin
alias fc0 "sl_mon 0 clrf"
alias ff0 "sl_mon 0 fpdp"
alias fs0 "sl_mon 0"
alias ls 'ls -F'
alias d dirs
alias to pushd
alias useMonsoon 'setenv MONSOON_HOME /usr/Monsoon; source /usr/Monsoon/etc/monsoon.csh'
alias dvlpMNSN 'setenv MONSOON_HOME /MNSN/soft_dev; source /MNSN/soft_dev/etc/monsoon.csh'
unalias vi
if (! $?MONSOON_HOME) then
 setenv MONSOON_HOME /usr/Monsoon
endif
source $MONSOON_HOME/etc/monsoon.csh
```

Appendix I.3 Example MONSOON Setup Script - monsoon.csh

```
#!/bin/csh -f
# alias(es)
alias addenv 'if (:${\!:1}\: !~ *:\!{:2}\:*) setenv \!:1 ${\!:1}\:\!:2'
alias fc0 "/opt/sl240/bin/sl_mon 0 clrf"
alias ff0 "/opt/sl240/bin/sl_mon 0 fpdp"
alias fs0 "/opt/sl240/bin/sl_mon 0"
```

```
alias useMonsoon‘setenv MONSOON_HOME /usr/Monsoon; source /usr/Monsoon/etc/monsoon.csh’
alias dvlpMNSN ‘setenv MONSOON_HOME /MNSN/soft_dev; source /MNSN/soft_dev/etc/monsoon.csh’
# path(s)
if (! $?MONSOON_HOME) then
 setenv MONSOON_HOME "/MNSN/soft_dev"
endif
echo MONSOON_HOME = ${MONSOON_HOME}
setenv MONSOON_BIN "${MONSOON_HOME}/bin"
setenv MONSOON_CFG "${MONSOON_HOME}/cfg"
setenv MONSOON_DOC "${MONSOON_HOME}/doc"
setenv MONSOON_DATA "${MONSOON_HOME}/data"
setenv MONSOON_ETC "${MONSOON_HOME}/etc"
setenv MONSOON_INC "${MONSOON_HOME}/inc"
setenv MONSOON_LIB "${MONSOON_HOME}/lib"
setenv MONSOON_LOG "${MONSOON_HOME}/log"
setenv MONSOON_SRC "${MONSOON_HOME}/src"
setenv MONSOON_SBIN "${MONSOON_HOME}/sbin"
setenv MONSOON_TCL "${MONSOON_HOME}/tcl"
setenv MEC_TCL_PATH "${MONSOON_HOME}/tcl"
setenv MONSOON_DATA_RHOST "chive.tuc.noao.edu"
# default(s)
setenv MONSOON_UTC "`/bin/date --utc +%Y%m%d`"
setenv MONSOON_FXSL "SystranBd.cfg"
setenv MONSOON_CLI "monsoon_Config.csv"
setenv MONSOON_ARR "monsoon.arr"
setenv MONSOON_DEF "monsoon_DefaultSetup.mod"
setenv MONSOON_GUI "monsoon_guiCategories.txt"
setenv MONSOON_SUP "monsoonClients.cfg"
setenv MONSOON_AUTH "monsoonAuth.cfg"
setenv MONSOON_LFD "monsoon.log"
# system
if (! $?CVSROOT) then
 setenv CVSROOT /MNSN/cvsRepository
endif
if (! $?LD_LIBRARY_PATH) then
 setenv LD_LIBRARY_PATH ${MONSOON_LIB}
else
 addenv LD_LIBRARY_PATH ${MONSOON_LIB}
endif
addenv LD_LIBRARY_PATH /usr/local/cfitsio/lib
addenv LD_LIBRARY_PATH /opt/sl240/bin
addenv PATH ${MONSOON_BIN}
```

Appendix II

Startup and Configuration Scripts

A MONSOON PAN system is started by a set of scripts that set up file links and start the appropriate processes on the PAN computer. Each script is designed to use the environment variables set up by the monsoon.csh script and based from the MONSOON_HOME directory variable.

Appendix II.1 FPSetup Setup Script for a Focal Plane

```
#!/bin/sh
# usage FPsetup focalPlaneName - This routine sets up the $MONSOONHOME/cfg
#directory on a pan to suppoort operations using a particular named focal
# plane by selecting and making symbolic links to the correct focal plane
# configuration files
#
if [ $# == 0 ]; then
  echo "usage: FPsetup fpConfigName"
  echo "      fpConfigName is the focal plane configuration name"
  exit -1
fi
echo $MONSOON_CFG
#Remove any old files from the default configuration directory. (NOTE this
#step may be uneccessary if a PAN is only used to support an single focal plane
cd $MONSOON_CFG
/bin/rm -rf monsoon* SystranBd.cfg
#These are links to the configuration files read by pan Daemon at config time
/bin/ln -s ../cfg/_$1/$1.arr monsoon.arr
/bin/ln -s ../cfg/_$1/$1_Config.csv monsoon_Config.csv
/bin/ln -s ../cfg/_$1/$1_guiCategories.txt monsoon_guiCategories.txt
/bin/ln -s ../cfg/_$1/$1_attributeLogList.txt monsoon_attributeLogList.txt
# files read by ppxSetMode
/bin/ln -s ../cfg/_$1/$1.ini monsoon.ini
/bin/ln -s ../cfg/_$1/$1_DefaultSetup.mod monsoon_DefaultSetup.mod
/bin/ln -s ../cfg/_$1/$1_DefaultExp.mod monsoon_DefaultExp.mod
/bin/ln -s ../cfg/_$1/$1_DefaultIdp.mod monsoon_DefaultIdp.mod
#default empty ucode file for sequencer
/bin/ln -s ../cfg/_$1/$1_DefaultSeq.ucd monsoon_DefaultSeq.ucd
#default client/auth
/bin/ln -s ../cfg/_common/monsoonAuth.cfg monsoonAuth.cfg
/bin/ln -s ../cfg/_common/monsoonLocalClients.cfg monsoonClients.cfg
/bin/ln -s ../cfg/_common/SystranBd.cfg SystranBd.cfg
#default library links
/bin/rm -f ../lib/libdetCmnds.so
/bin/ln -sf ../lib/detectorLibs/lib$1.so ../lib/libdetCmnds.so
# DHS library should be set up for local DHS routines ONCE.
# libdhsFITS writes MEF files to local disk and should only be used when no
# DHS library is available which sends image data to a remote machine over
# a gigabit ethernet or otherfast link.
#
/bin/rm -f ../lib/libdhsUtil.so
/bin/ln -sf ../lib/dhsLibs/libdhsFITS.so ../lib/libdhsUtil.so
```

Appendix II.2 runPAN Startup Script

This script starts the PAN processes. Each process is started in an xterm for debug purposes. This should become unnecessary in production situations.

```
#!/bin/sh

if [ $# == 0 ]; then
 echo "Usage: $0 fp [host [DBG [sleep]]]"
 echo " fp   focal plane configuration <generic>"
 echo " host  machine running the DHS server <localhost>"
 echo " DBG DBG number for the PAN processes <100>"
 echo " sleep wait time between process activation <5>"
 exit -1
elif [ $# == 1 ]; then
 fp=$1
 host="localhost"
 MONSOON_DATA=/home/monsoon/data
 DBG=100
elif [ $# == 2 ]; then
 fp=$1
 host=$2
 MONSOON_DATA=/home/monsoon/data
 DBG=100
elif [ $# == 3 ]; then
 fp=$1
 host=$2
 MONSOON_DATA=$3
 DBG=$4
else
 fp=$1
 host=$2
 MONSOON_DATA=$3
 DBG=$4
fi

MONSOON_DATA_RHOST=${host}
export MONSOON_DATA_RHOST
export MONSOON_DATA

cd $MONSOON_CFG
$MONSOON_BIN/FPsetup ${fp}
X11BIN = /usr/X11R6/bin

${X11BIN}/xterm -title panDaemon -geometry +2+2 -e $MONSOON_BIN/panDaemon -u0 -D${DBG} &
sleep 1
${X11BIN}xterm -title panCapture -geometry +570+2 -e $MONSOON_BIN/panCapture -u0 -D${DBG} &
${X11BIN}/xterm -title panProcAlg -geometry +2+396 -e $MONSOON_BIN/panProcAlg -u0 -D${DBG}
&
${X11BIN}/xterm -title panSaver -geometry +570+396 -e $MONSOON_BIN/panSaver -u0 -D${DBG} &
```

Appendix II.3 Focal Plane Scripts

Each MONSOON focal plane configuration has its own executable script. These scripts have the name of the focal plane and take care of the clean-up and start-up tasks for that focal plane. These scripts eventually run the "runPAN script that starts the processes required by the focal plane. Below are examples of the focal plane start-up scripts and the runPAN script.

"generic" focalplane script - a bare bones IR focal plane.

```
#!/bin/sh
#
userNum=`id -u`
mach=`hostname`

shmUNuke $userNum
semUNuke $userNum

runPAN generic ${mach} 100
```

"orion_II" focalplane script - used in the IR R&D lab for testing the ORION_II InSb detectors

```
#!/bin/sh
#
userNum=`id -u`
mach=`hostname`

/opt/sl240/bin/sl_mon 0 clrf

shmUNuke $userNum
semUNuke $userNum

runPAN orion_II ${mach} $1 $2
```

"orion_II_2x2" focalplane script - used in the NEWFIRM PANs

```
#!/bin/sh
#
userNum=`id -u`
mach=`hostname`

/opt/sl240/bin/sl_mon 0 clrf

shmUNuke $userNum
semUNuke $userNum

runPAN orion_II_2x2 ${mach} $1 $2
```

Appendix III Focal Plane Configuration Files

Appendix III.1 Attribute Arrays

A feature provided by the MONSOON configuration system is the ability to describe groups of similar DHE functions as an attribute array. For example, the NOAO Clock and Bias board contains 36 Bias voltage DACs. They can be described in the configuration file as follows:

Client Name, DHE Name, DHE Addr, Number in group,..,Help Text

```
clk0_Bias,CLK_BD0_BIAS,0x0080100,36,....,36 slow bias output channels on Clock/Bias Board 0
```

Once this has been done, setting CLK_BD0_BIAS[] = value will set all 36 Bias DACs to the requested value. Setting clk0_Bias[3] = value will set only the fourth Bias to the requested value. Note that all of the members of the array will have the same conversion factors, limits and help text. See Attribute Aliasing which follows.

Appendix III.2 Attribute Aliasing

The detector or system engineer uses the configuration file to describe the system functions in language and using names that are familiar to the engineers and the users. The engineering function names are generic and describe a function on one of the boards in the DHE or a software function in the PAN. Names like CLK_BD0_BIAS or SEQ_LOOPREG[2], should be associated with names which are useful to the users. There are several mechanisms that allow this to occur.

1. In the case of registers that are bit fields, the engineer can assign multiple names to the same register address to access each bit individually. Following is an example using the MCB_CLKENABLES register:

```
mcbClkEnables,MCB_CLKENABLES,0x0010100,.... # access to the entire register
clkBiasEnbl,MCB_CLK_ENABLE_3,0x0010100,..... # access to the bit controlling slot
3
AcqBd0Enbl,MCB_CLK_ENABLE_5,0x0010100,1,.. # access to the bit controlling slot 5
AcqBd1Enbl,MCB_CLK_ENABLE_6,0x0010100,1,....
```

The lines above provide a single name (mcbClkEnables), which accesses all the bits in the clock enable register as well as individual attributes which access only the bit controlling each board in the system. The RdMskWrt setting mechanism provides the proper control so only a single bit or group of bits is set.

2. Each entry in a DHE function array can be individually named using the alias facility in the MONSOON configuration file. Following is an example of this facility being used to give client level names to the sequencer registers and to the Bias voltage channels:

```
seqLoopReg,SEQ_LOOPREG,0x0010110,16,......,sequencerloop registers array 0-15
rows,SEQ_LOOPREG[0], ,1,0x06000000,...... ,number of rows to loop over- loop register 0
cols,SEQ_LOOPREG[1], ,1,0x06000000,........,number of columns to loop over - loop register
1
rowBin,SEQ_LOOPREG[2], ,1,0x06000000,...., number of rows to bin - loop register 2
colBin,SEQ_LOOPREG[3], ,1,0x06000000,....., number of cols to bin - register 3
```

As for the Bias voltages, note that the conversion factors and allowed voltage ranges for arrays of functions can be modified by using the aliasing technique so safe values for voltages can be maintained.

Client, DHEName, DHEAddr conversion Factor Min/Max allowed voltage

```
clkBias,CLK0_BIAS,0x0080100,36,....,17,175,LINEAR,  -10.24,4.76,  VOLTS,36 slow bias output
channels
vnrow,CLK0_BIAS[0], ,1,....     ,17,175,LINEAR,     -7.0,0.0,     VOLTS,vnrow bias voltage
VssExt,CLK0_BIAS[1], ,1,...     ,22,128,LINEAR,      0.0,2.4,     VOLTS,VssExt bias voltage
```

Notice in the Bias voltage example that both the conversion factors and the Max/Min allowed voltages can be changed from the defaults set by the original array descriptor. The voltage array can still be set using the CLK0_BIAS[]=value technique. However, if any of the voltages have been aliased in such a way that the requested voltage is no longer legal for any array member (i.e. if value = -1.0 in the above example), then the command will fail when it tries to set the illegal value (in this case when we try to set VssExt or CLK0_BIAS[1] to -1.0 volts. The command will return an out of range message after setting vnRow to -1.0

Appendix III.3 CReg Usage

The Command Classification register or CReg value is provided to give a single location where a number of pieces of information about an attribute can be stored. In use, the register is broken up into several fields described in the following. Not all of this functionality is implemented or completely tested. In particular, the restrictions inherent in ATT_NOSAVE and ATT_SYSONLY are not functional at this time.

```
/*****************************************************************************
 * Command classification register values - these classify the commands and attributes
 *  according to when and how they can be executed.
 *****************************************************************************/
/* Creg Field Masks */
#define ATTRIB_USAGE     0x00000FFF
#define ATTREADONLY      0x00000001   /* TRUE if attribute is Read only in the DHE */
#define ATT_NOSAVE 0x00000002   /* Attribute is not saved with ppxGetMode <SAVE>*/
#define EXEC_ANYTIME     0x00000004   /* attribute can be executed/accessed anytime */
#define ATT_SYSONLY      0x00000008   /* attribute modified only by system mode files */
#define READ_OK_DEXP     0x00000010   /* attribute may be read during DHE_EXP_ACTIVE */
#define WRITE_OK_DEXP    0x00000020   /* attribute may be written during DHE_EXP_ACTIVE */
#define RETDHETYPE 0x00000040   /* attribute Value return Type same as DHE type */
#define NOFLAG128  0x00000080   /* Not Used */
#define NOFLAG256  0x00000100   /* Not Used */
#define NOFLAG512  0x00000200   /* Not Used */
#define NOFLAG1024 0x00000400   /* Not Used */
#define NOFLAG2048 0x00000800   /* Not Used */

#define CONFIG_GROUP     0x0000F000
#define ARRCONFIG  0x00001000   /* TRUE if attribute is part of the array config*/
#define EXPCONFIG  0x00002000   /* TRUE if attribute is part of the Exposure config */
#define IDPCONFIG  0x00004000   /* TRUE if attribute is in the Data process config */
#define DLDCONFIG  0x00008000   /* obsolete? replaced by MEMCONFIG? */
#define MEMCONFIG  0x00008000   /* TRUE if attrib is in the downloaded code config */

#define GUICATEGORY      0xFF000000   /* the GUI class of the attribute 0-255 */
                                      /* see the guiCategories.txt file explanation */
#define UNUSED_CREG      0x00FF0000   /* These bits are currently unassigned */
```

Appendix III.4 Set and Read Methods

The base MONSOON system provides a number of standard methods to access the attributes in the DHE and PAN. These standard methods implement all of the access methods currently implemented in the MONSOON PAN software and DHE firmware. These methods are selected by NAME for each attribute in the configurations file. Each attribute has a SET and a GET method field in the configuration record. There are thirteen standard methods defined in cmdCfgUtil.h. These methods are coded as part of the generic detector library and are automatically included from that library when a new detector library is built. If new methods are required for a system, the routine to implement those methods should be included in the generic detector directory and their descriptors added to the setMethod/getMethod tables in detCmnds.h for inclusion in new systems.

NOMETHOD - NOMETHOD is used for read-only or write-only attributes. Examples are status registers that are filled in by the DHE firmware or trigger addresses which, when written to, start a DHE firmware process. If called, the method returns a SUCCESS message saying the attribute in question cannot be read or written to.

SIMPLE - On SET access, this method converts the value field to the DHE format as needed and then does a write of the value to the attribute DHE address or the PAN structure value field. On GET, this method does a read of the value from the attribute DHE address or the PAN structure value field and then converts from DHE values to PAN values as needed.

INTTIME - On SET, the requested integration time is compared to the minimum possible integration time and the minimum of the requested time and the minimum time is written to the DHE integration timer register after conversion from seconds to milliseconds. On GET, reads the requested integration time from the PAN structure.

NOTE: ROISETC and ROISETR are used when a single region of interest is read out on a PAN-DHE pair to speed up the readout. In this mode, the user may either read out the same single ROI on all chips controlled by a PAN-DHE or can elect to read out only a single chip with this ROI. The described ROI may not cross chip boundaries.

ROISETC - (NYI) On SET, the method will use spdRoiXLo from spdRoiXHi to determine the number of columns to skip over at high speed and the number of columns to read out for this exposure and load those values into the appropriate sequencer registers. On GET, the method will read back the values in the columns to skip and columns to read registers in the sequencer.

ROISETR - (NYI) On SET, the method will use spdRoiYLo from spdRoiYHi to determine the number of rows to skip over at high speed and the number of rows to read out for this exposure and load those values into the appropriate sequencer registers. On GET, the method will read back the values in the rows-to-skip and columns-to-read registers in the sequencer

FNAMESET - This method stores or reads back the file name the user requested with "%04d_%c.fits". Appended, this allows each exposure to be given a unique name on disk. Note that this is only used if no DHS library is available. (superseded by STRINGSET)

SETVOFF - On SET, this method converts the requested video offset voltage to a DHE bit value and writes the requested value to onE of the video offset banks. It then writes to the trigger address to load the requested bank into the video offset DACs. On GET, this methods reads the requested video offset bank register and converts to a voltage.

SETBIASV - Provided to set the bias for IR detectors. This method uses the requested bias to determine the value of the bias voltage by subtracting two bias voltages. Not yet implemented and currently calculated manually.

SETFBIAS - OBSOLETE - this method converts the requested voltage to a bit pattern. Shifts the 8-bit result to the correct byte and loads four fast biases at the same time.

WRT2READ - Used for reading housekeeping channels. The first write triggers the ADC, which is then read to determine the value on the housekeeping channel.

STRINGSET - Used by PAN-only string attributes to set or get string values. Mostly used to set the setup file and directory names and to designate sequencer ucode files to load.

RDMSKWRT - On SET, this method reads the DHE address for the attribute then creates a mask, clears the masked bits in the value read, then converts the requested value and ORs the converted bit pattern into the read value and writes the result back to the DHE. This is used to access bit fields in some registers. On GET, the method reads the vale from the register and masks off the unwanted bits and converts to a PAN value.

DIGAVGS - Only used in systems that use the MONSOON IR Acquisition board. On SET, the requested number of digital averages (1, 2, 4, 8, 16, 32 or 64) is converted to an index 0,1,2,3,4,5,6 and loaded into the digAvgCnt on each IRAcq board and then into the digAvg Delay register in the sequencer. Finally the digAvgEnbl bit is set in the sequencer and IRAcq boards.

Appendix III.5 PAN and DHE Attribute Data Types

These two columns of the configuration file describe the type of attributes in the PAN and in the DHE. Currently two types are permitted in the PAN; String and Float. Strings are represented by 'C' char arrays, Floats are represented by 'C' doubles. Both of these types are converted using the attribute conversion functions into the DHE Types. Types in the DHE are all integer types of various sizes. The currently implemented DHE register sizes are:

ONEBIT - a single bit, used for flags and triggers; (0 or 1)

BYTE, **CHAR** - aliases for a signed, two's compliment 8-bit value; (-128 - 127) or (0x80 - 0x7F)

UCHAR - an unsigned 8-bit value; (0 - 255) or (0x00 - 0xFF)

LONG - a signed, two's compliment 32 bit value: (-2147843648 - 2147843647) or (0x80000000-0x7FFFFFFF)

ULONG - an unsigned 32 bit value: (0x00000000 - 0xFFFFFFFF)

SHORT -a signed, two's compliment 16 bit value: (-32768 - 32767) or (ox8000 - 0x7FFF)

USHORT - an unsigned 16 bit value: (0x0000 - 0xFFFF) or (0 - 65365)

TWLVBIT - an unsigned 12 bit value: (0x000 - 0xFFF) or (0 - 4095). This is used for the 12 bit ADC used for housekeeping readback

TWNT4BIT - an unsigned 24 bit value: (0x000000 - 0xFFFFFF) or (0 - 16777215). Used for readback from 18-24 bit ADC's which may be used on new CCD readout boards.

NOTE: Most of the current hardware and firmware in the DHE uses unsigned values even when the output voltages are positive and negative. The conversion from signed voltage to unsigned bit pattern takes place in the PAN software

Appendix III.6 Attribute Conversion Functions

Provision is made in the PAN software for the implementation of any number of conversion functions to convert PAN or real world values to or from the bit patterns in the DHE. Currently three functions are defined. However, the only conversion actually being used and tested is the LINEAR function. Additional functions should be added to the generic detector library directory.

```
/*****************************************************************************
 * Conversion method ID values for _cnvrtToPAN and _cnvrtToDHE functions
 *****************************************************************************/
#define LINEAR    0      /* function is DHEValue = (coef1 * PANValue) + coef2 */
                         /*     or PANValue = (DHEValue - coef2) / coef1 */
#define POWER     1      /* function is DHEValue = coef1 * (PANValue ^ coef2) */
#define TABLE1    2      /* function is linear interpolation of table1 values */
```

Appendix IV Software Attribute Usage

The PAN software uses and depends on the existence of certain attributes in addition to those declared by the detector engineers. This section lists those attribute names that are compiled into and used by the PAN software. A complete list of these attributes is kept in attributeNames.h in the main detector library directory. The attributeNames.h file is used to give symbolic names to these attributes. This allows the actual text string name to change without having to make changes in every file that uses the attribute name. Changing the macro definition in attributeNames.h and recompiling the appropriate files and libraries is sufficient to make a change in the attributes text name.

The user of a MONSOON system should note that many attributes are included for record keeping purposes. This allows the end user to review the attribute values to determine how the system was set up when an exposure was taken. The MONSOON software may not “use” an attribute like arrayID except to record its value. Likewise the settings for voltages and most DHE registers are unimportant for the MONSOON software to capture, process and archive exposure data. It is left to the client software and the detector engineer/instrument scientist to ensure that the MODE descriptions (files or database entries) are consistent with safe and effective operation of the focal plane detectors.

Appendix IV.1 Attributes Common to All Focal Planes

Table 1 - Attributes Common to All Focal Planes

Mode File and Directory Attributes (** = obsolete)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
arrFname	arrFname_S	Name of the array/detector setup file used to set up the system. Contains a subset of the attributes on the major mode file.	ppxSetArrcfg.c:58, ppxSetArrcfg.c:72, ppxSetArrCfg.c:142
arrFdir	ArrFdir_S	Directory to be searched for array/detector mode files.	ppxSetArrCfg.c:54, ppxSetArrCfg.c:71, ppxSetArrCfg.c:141, ppxSetArrCfg.c:154
dwnLdFname	memFname_s dwnLdFile_S**	Name of the memory file to be downloaded to system memory. Usually used to load the sequencer program and pattern memories.	ppxSetMemCfg.c:123,
dwnLdFdir	memFdir_S dwnLdDir_S**	Directory to be searched for download/memory mode files.	ppxSetMemCfg.c:84, ppxSetMemCfg.c:128
modeFname	modeFname_S	Name of the major mode file used to set up the system.	ppxSetMode.c:53, ppxSetMode.c:65, ppxSetMode.c:174
modeFdir	modeFdir_S	Directory to be searched for major mode files.	ppxSetMode.c:49, ppxSetMode.c:64, ppxSetMode.c:173, ppxSetMode.c:187
expFname	expFname_S	Name of the exposure setup file used to set up the system for an exposure. Contains a subset of the attributes on the major mode file.	ppxSetExpCfg.c:57, ppxSetExpCfg.c:70, ppxSetExpCfg.c:139,
expFdir	expFdir_S	Directory to be searched for major mode files.	ppxSetExpCfg.c:53, ppxSetExpCfg.c:69, ppxSetExpCfg.c:138, ppxSetExpCfg.c:151
idpFname	idpFname_S	Name of the array/detector setup file used to set up the system. Contains a subset of the attributes on the major mode file.	ppxSetIdpCfg.c:58, ppxSetIdpCfg.c:72, ppxSetIdpCfg.c:141
idpFdir	idpFdir_S	Directory to be searched for array/detector mode files.	ppxSetIdpCfg.c:54, ppxSetIdpCfg.c:71, ppxSetIdpCfg.c:140, ppxSetIdpCfg.c:154

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Data Disposition Attributes (= obsolete)**			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
fileName (**)	fileName_S (**)	Obsolete. Replaced by imageFile.	_testSaver/detStartExp.c:103
imageCount	imageCount_S	A count of the number of images saved since the last restart of the PAN software. Used to distinguish image files in lab systems with the MSD feature disabled.	panSaver/detSaveImage.c:84, panSaver/detSaveImage.c:88
imageDir	imageDir_S	The directory where the image files created by a start exposure command are stored.	panSaver/detSaveImage.c:91
imageFile	imageFile_S	The name of the image file to be used to store the image created by a start exposure command. Used by the FITS DHS library.	panSaver/detSaveImage.c:107
expEndTime	expEndTime_S	The MONSOON Star Date obtained after the capture of the last frame of an exposure. Based on the Linux system time/date.	_genCCD_Mosaic/detCapture.c:209, _generic/detCapture.c:227, _generic_CCD/detCapture.c:209, _orion_II/detCapture.c:219, _orion_II_2x2/detCapture.c:220, _orion_II_Fst/detCapture.c:221, _orion_II_Sml/detCapture.c:220
expID	expID_S	The MONSOON Star Date which identifies the current exposure. This is set by the Client software or, if not set, created by the PAN software.	ppxStartExp.c:102, ppxStartExp.c:116, Detectors/_*/ detExpParams.c:...
expStrtTime	expStrtTime_S	The MONSOON Star date obtained after the capture of the first frame of an exposure. Based on the Linux system time/date.	_genCCD_Mosaic/detCapture.c:207, _generic/detCapture.c:226, _generic_CCD/detCapture.c:207, _orion_II/detCapture.c:218, _orion_II_2x2/detCapture.c:219, _orion_II_Fst/detCapture.c:220, _orion_II_Sml/detCapture.c:218

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Detector/Focal Plane Description (= obsolete)**			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
arrayType	arrayType_S	String name of the array type in the focal plane, for example, aladdin_III, orion_II, ccd_EEV42, ccd_STIS	Detectors/_*/detGetState.c:27
detName	detName_S**	Obsolete. Replaced by arrayType	
arrayID	arrayID_S	String containing the ID strings (serial numbers) of all the detectors in the focal plane.	Detectors/_*/detGetState.c:28
mosaicCols	mosaicCols_S	The number of columns of detectors in the focal plane. The number of detectors in each row of the focal plane	panDaemon/panSysConfig.c:93, panSaver/detSaveImage.c:69, Detectors/_*/detCalcPixels.c:~31, Detectors/_*/detDescramble.c:111
mosaic rows	mosaic rows_S	The number of rows of detectors in the focal plane. The number of detectors in each column of the focal plane.	panDaemon/panSysConfig.c:91, panSaver/detSaveImage.c:68, Detectors/_*/detCalcPixels.c:33, Detectors/_*/detDescramble.c:110
pxlCols	pxlCols_S	The number of active (photon collecting) pixels in each detector in the x direction. The number of pixels in each row of the detector. This does not include prescan, postscan, overscan or reference pixels.	Detectors/_*/detCalcPixels.c:34, Detectors/_*/detGetState.c:31,
pxlRows	pxlRows_S	The number of active (photon collecting) pixels in each detector in the y direction. The number of pixels in each column of the detector. This does not include prescan, postscan, overscan or reference pixels.	Detectors/_*/detCalcPixels.c:35, Detectors/_*/detGetState.c:32

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Detector/Focal Plane Description (** = obsolete) (Cont.)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
imageCols	imageCols_S	The total number of pixels in each row of the focal plane. This includes pre-, post-, overscan and reference pixels.	panSaver/panGetImgParams.c:25, Detectors/_*/detCalcPixels.c:99, Detectors/_*/detDescramble.c:153,..., Detectors/_*/detCalcPixels.c:36, Detectors/_*/detCalcPixels.c:65
imageRows	imageRows_S	The total number of pixels in each column of the focal plane. This includes pre-, post-, overscan and reference pixels.	panSaver/panGetImgParams.c:25, Detectors/_*/detCalcPixels.c:99, Detectors/_*/detDescramble.c:153,..., Detectors/_*/detDescramble.c:208,..., Detectors/_*/detCalcPixels.c:65, Detectors/_*/detDescramble.c:73
pxlsPerImage	pxlsPerImage_S	The total number of pixels produced by each read of the focal plane. This may be smaller than the actual focal plane if ROI's are defined.	Detectors/_*/detExpParams.c:105
frmsPerRdOut	frmsPerRdOut_S	The number of frames of size pxlsPerImage produced by each start exposure for this detector type, expMode and readout scheme.	Detectors/_*/detExpParams.c:66
totFrames	totFrames_S	The total number of frames of size pxlsPerImage produced by each start exposure for the current configuration.	Detectors/_*/detExpParams.c:108
rdOutTime[%d]	rdOutTime_S	An attribute array indexed by the digital averages index which gives the time in milliseconds required for one read of the entire focal plane for the current configuration.	Detectors/_*/detCapture.c:57,85,93, Detectors/_*/detExpParams.c:81,83

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Detector/Focal Plane Description (** = obsolete) (Cont.)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
xStart	xStart_S	An attribute array indexed by the array number (N) in the focal plane which gives the x (column) coordinate in the total focal plane of the first pixel in detector N. Array numbers are assigned by the client software or, if not set, assigned arbitrarily by the PAN software.	panSaver/detSaveImage.c:172
yStart	yStart_S	Similar to xStart but gives the y (row) coordinate of the first pixel in detector N.	panSaver/detSaveImage.c:175
finDataType	finDataT_Sype	The type of the data produced by the processing algorithm. Currently this is set to 32 the FITS value for long integer data. This is determined by the detProcess routine written to support the detector type in the focal plane.	panSaver/panGetImgParams.c:39
finPxlSize	finPxlSize_S	Size of the pixels produced by the processing algorithm in bytes. Used by the detDescrambling algorithm to determine how to move pixels when descrambling images.	Detectors/_*/detExpParams.c:123, Detectors/_*/detProcess.c:48
rawPxlSize	rawPxlSize_S	Size of the pixels produced by the data capture routine in bytes. Used by the detProcess algorithm to determine how to process raw pixels into finished pixels ready for descrambling and archiving.	Detectors/_*/detExpParams.c:123, Detectors/_*/detProcess.c:48

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (= obsolete)**			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
actIntTime	actIntTime_S	The actual integration time to the nearest ms	
intTime	intTime_S	The integration time requested by the client software (namely, the observer) and loaded down to the DHE integration timer.	Detectors/_*/detExpParams.c:42
binning (**)	binning_S (**)	Obsolete. Replaced by xBinning and yBinning.	Detectors/_generic/detExpParams.c:72
coadds	coadds_S	The number of frames to coadd together to construct the final image. The raw data for a coadded image is discarded after being coadded.	Detectors/_generic/detExpParams.c:72
numOutputs	numOutputs_S	The number of output channels on each detector in the focal plane. Currently this must be the same for all detectors.	Detectors/_*/detCalcPixels.c:40, / Detectors/_*/detExpParams.c:78, Detectors/_*/detDescramble.c:68
minIntTime	minIntTime_S	The minimum integration time for the current configuration determined by readout time, shutter operation time or whatever physical or software parameter is the choke point for detector readout.	

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (** = obsolete)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
Attributes for Infrared Detectors (aladdin, orion, virgo, etc.)			
fSamples	fSamples_S	The number of Fowler samples (frames to average) for the current configuration. Currently set in all configurations to a maximum of 64 frames.	Detectors/_*(IR)/detProcess.c:51, Detectors/_*/detExpParams.c:58, Detectors/_*/detCapture.c:96
digAvgs	digAvgs_S	The number of CTC-data conversions to do on each pixel of the focal plane. The resulting data is summed and then divided by the number of samples. Values are restricted to powers of two between 1 and 64.	Detectors/_*(IR)/detExpParams.c:74
acq0DigAvg	acq0DigAvg_S	Each IR Acquisition board in the system has a register that contains the $\log_2$ of the number of digital averages for the current configuration. These values are used to do the averaging by shifting the summed values to the right by this value. These values are set by the setDigAvg routine in libdetCmnds.	Detectors/_*/digAvg.c:96,..., (IR)
acq1DigAvg	acq1DigAvg_S		
acq2DigAvg	acq2DigAvg_S		
acq3digAvg	acq3digAvg_S		
seqDigAvgCntr	seqDigAvgCntr_S	A sequencer loop counter that is used to count the number of digital averages to be done for the current exposure. This is set to the correct value by the setDigAvg routine in libdetCmnds.	Detectors/_*/digAvg.c:96,..., (IR)

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (= obsolete) (Cont.)**			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
Attributes for Infrared Detectors (aladdin, orion, virgo, etc.) (Cont.)			
mcbSeqDigAvgEn	mcbSeqDigAvgEn_S	A user flag bit in the sequencer control register which tells the sequencer that digital averaging is enabled. This is set to the correct value by the setDigAvg routine in libdetCmnds.	Detectors/_*/digAvg.c:96,..., (IR)
refCols	refCols_S	The number of reference columns that will be produced by the detector readout in the current configuration.	Detectors/_*/detExpParams.c:64, Detectors/_*/detCalcPixels.c:33, Detectors/_*/detDescramble.c:81, Detectors/_*/detExpParams.c:70
loopDelay	loopDelay_S	A sequencer loop counter that allows each readout sequence in a set of Fowler samples to be delayed by a determined amount. The minimum value of the delay is 5ms. This allows time for the PAN software to call the readData routine.	Detectors/_*/detExpParams.c:70
Attributes for OpticaL/CCD Detectors			
colBin	xBinning_S	The binning factor in the x (column) or y (row) directions. Regardless of the actual orientation of the detectors rows are always the slow shifts and columns the fast shifts.	NYI, Detectors/_*/detCalcPixels.c:33
rowBin	yBinning_S		NYI, Detectors/_*/detCalcPixels.c:33

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (= obsolete) (Cont.)**			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
Attributes for Optical/CCD Detectors (Cont.)			
xPostScan	xPostScan_S	The number of pixel values produced on each row after the readout of the active (photon collecting) pixels.	Detectors/_*/detCalcPixels.c:37, Detectors/_*/detExpParams.c:60
xPreScan	xPreScan_S	The number of pixel values produced on each row before the start of readout of the active (photon collecting) pixels.	Detectors/_*/detCalcPixels.c:36, Detectors/_*/detExpParams.c:57
yPostScan	ypostScan_S	The number of extra rows of pixel values produced for each detector after the readout of the active (photon collecting) pixel rows.	Detectors/_*/detCalcPixels.c:38, Detectors/_*/detExpParams.c:63
outputCfg	outputCfg_S	A description of the position of active output transistors on every detector in the focal plane. Used if there are two outputs. See Appendix V. The following are possibilities for output configurations: AB, CD, AD, BC, AC and BD. These are designated in the attribute as: AB=0, CD=1, AD=2, BC=3, AD=4 and BC = 5.	Detectors/_*/detCalcPixels.c:39, Detectors/_*/detDescramble.c:106, Detectors/_*/detExpParams.c:75
outputGain	outputGain_S	The CCD acquisition board register containing the gain setting for the current configuration.	
abort	abort_S	Designates the DHE address used when aborting an exposure. Abort means stop the integration and/or readout and dispose of the data collected so far.	Detectors/_*/detAbort.c:...

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (** = obsolete) (Cont.)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
Attributes for Optical/CCD Detectors (Cont.)			
pause	pause_S	Designates the DHE address used when pausing an exposure. IR systems and systems without shutters may not be able to pause.	Detectors/_*/detAbort.c:...
resume	resume_S	Designates the DHE address used when resuming an exposure.	Detectors/_*/detPause.c:...
shutterCntrl	shutterCntrl_S	Determines how the shutter should be handled. 0 is close and leave closed during exposures, 1 is open during the integration and close after and 2 is open and leave open.	
shutterState	shutterState_S	Gives the current state of the shutter. 0 for closed 1 for open.	Detectors/_*/detAbortExp.c:86, Detectors/_*/detPauseExp.c:97, Detectors/_*/detResumeExp.c:100
mcbSeqEnable	mcbSeqEnable_S	A BIT flag in the MCB sequencer control register that enables and disable the running of the detector clock sequencer.	Detectors/_*/detSeqSetup.c:59
captureMode	captureMode_S	Determines the mode/technique used to capture the data from the focal plane. Currently all modes are the same. Provided to allow future flexibility.	Detectors/_*/detExpParams.c:48

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Exposure Attributes (** = obsolete) (Cont.)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
Attributes for Optical/CCD Detectors (Cont.)			
processMode	processMode_S	Determines the mode/technique used to pre-process data from the focal plane. Currently two modes are used, STARE or NORMAL. (processMode=0) is the normal data taking mode for IR systems. It involves two reads of the detector subtracting the first frame the second pixel by pixel. SEP mode sends each frame to panSaver to be archived without processing. For CCD systems STARE and SEP are equivalent for coadds <=1.	Detectors/_*/detExpParams.c:54, Detectors/_*/detProcess.c:39
expMode	expMode_S	Determines the mode of the exposure. Provided for future flexibility to allow for coordinated handling of Darks vs. image data or other exposure technique differences.	Detectors/_*/detExpParams.c:45, Detectors/_*/detExpParams.c:45
expVector	expVector_S	Determines the value loaded into the Sequencer command register when a startExp command is issued. It is used to control the sequencer code in the MCB.	Detectors/_*/detExpParams.c:51

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Debugging Engineering Attributes (** = obsolete)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
labSystem	labSystem_S	Controls aspects of the system related to data taking. If set to 0, it indicates a science system and full functionality is used. If set to 1, the use of the MONSOON Star Date to create file names is used but the pre- and post-exposure attributes collection is suppressed. If set to 2, the MONSOON Star Date is suppressed. If set to 3, both the MSD and attribute collection are suppressed.	panCapture/panDataCapture.c:98, panSaver/panDataSaver.c:77, panSaver/detSaveImage.c:100
asyncVector	asyncVector_S	Value sent to the DHE when responding to an asyncMsg from the DHE. Currently ignored.	Detectors/_*/detAsyncResp.c:56
avCommentSize	avCommentSize_S	Size in bytes (characters) of the attribute-value table comment field (currently 64 characters).	panSaver/panGetImgParams.c:49
avNameSize	avNameSize_S	Size in bytes (characters) of the attribute-value table attribute name field (currently 32 characters).	panSaver/panGetImgParams.c:45
avValuseSize	avValuseSize_S	Size in bytes (characters) of the attribute-value table attribute value field (currently 32 characters).	panSaver/panGetImgParams.c:47
ctcPeriod	ctcPeriod_S	Test attribute to determine the time between CTC's in ADC FFT test mode sequences.	

Appendix IV.1 Attributes Common to All Focal Planes (cont.)

Debugging Engineering Attributes (** = obsolete) (Cont.)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
numTstRows	numTstRows_S	Test attribute to determine the number of rows of FFT test data to produce. (Each row is as wide as the number of ADCs in the system).	
panState	panState_S	Current state value of the PAN system. Possible states are: PAN_IDLE(0) PAN_EXP_ACTIVE(1) PAN_EXP_PAUSED(2) PAN_DATAXFR_LOCK(3) PAN_LASTXFR_LOCK(4) PAN_ABORTING(5) PAN_STOPPING(6) PAN_EXP_COMPLETE(7) PAN_ERROR(8) PAN_DATADONE(9)	Internal variable which is not yet implemented as an attribute

Region of Interest Attributes (** = obsolete)			
Text Attribute Name	**Internal Macro Name**	**Definition**	**Where Used**
drROIXHi	drROIXHi_S	Data reduction ROIs are intended to allow the transfer and archiving of only the areas of a focal plane that contain interesting data. The entire focal plane is read out but only the ROIs are preserved. The rest of the data is deleted.	Not yet implemented.
dROIXLo	dROIXLo_S		
drROIYHi	drROIYHi_S		
drROIYLo	drROIYLo_S		
spdROIXHi	spdROIXHi_S	Speed ROIs allow the capture of data from events whose timing exceeds the readout time of the focal plane. In this case, a single area of each detector, or a single detector, is read out and preserved. The remaining portion of the focal plane is not read out.	Not yet implemented.
spdROIXLo	spdROIXLo_S		
SpdROIYHi	spdROIYHi_S		
spdROIYLo	spdROIYLo_S		

Appendix IV.2 Focal Plane Specific Attributes – Generic

Appendix IV.3 Focal Plane Specific Attributes - orion_II

acq0digAvg
acq1digAvg
fSamples
refCols
refPixels
seqDigAvgCntr
seqDigAvgEn
"CLK0_TDATA[5]", (void *)&tmpVal);panSaver/detSaveImage.c:181:
"CLK0_TDATA[6]", (void *)&tmpVal);panSaver/detSaveImage.c:141:
"actualIntTime"

Appendix IV.4 Focal Plane Specific Attributes - orion_II_2x2

Appendix IV.5 Focal Plane Specific Attributes - generic_CCD

Appendix IV.6 Focal Plane Specific Attributes - genCCD_Mosaic

Appendix IV.7 Focal Plane Specific Attributes – ccdlabOTAtest

Appendix V

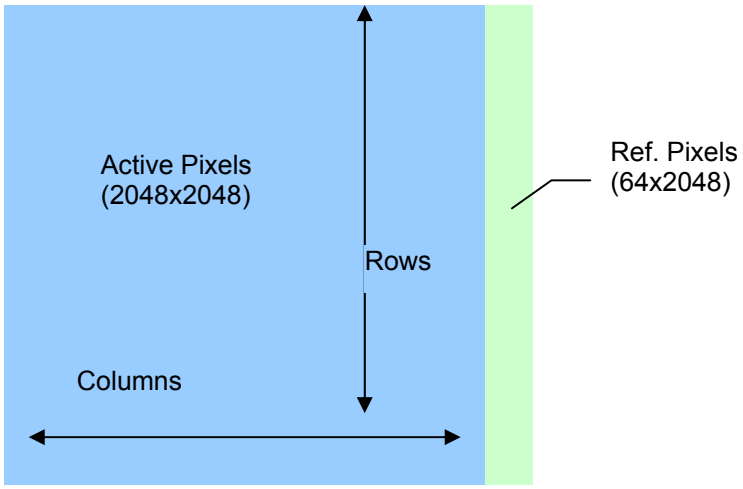
Descrambling Pixels in MONSOON

The MONSOON system is designed to deal with a wide variety of focal planes. So that the output of a MONSOON system is consistent from one system to another, a DHS API in draft format has been proposed. This API was designed so a DHS system would know how the data was organized based on the parameters sent when saving the data.

Regardless of the orientation of the detector on the focal plane, MONSOON considers the pixels to be generated by columns first, then by rows. In any detector the first pixel generated is considered pixel 1,1 and is placed in the lower left corner.

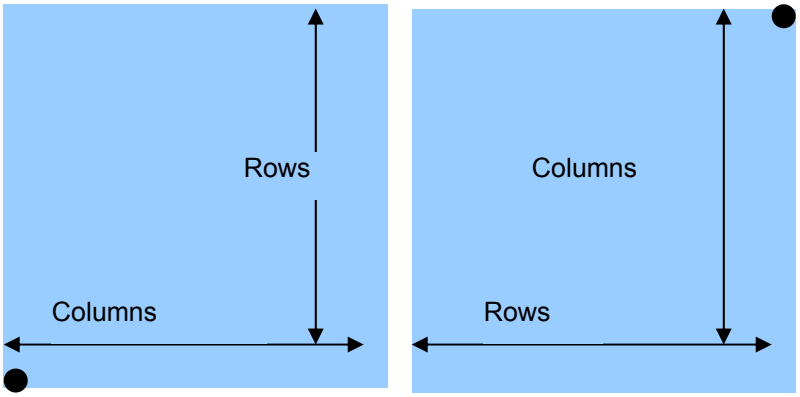
The data sent to the DHS consists of a pre-exposure block of metadata (temperatures, voltages), then N blocks of image data representing the data from each of the N detectors in the focal plane then a final block of post-exposure metadata.

Descrambling the image data provides the end user with a simple format for the image. For the IR instruments (NEWFIRM, ORION_II test systems) each detector is descrambled as shown in Figure 4.



IR Detector Descrambling
Figure 4

In the NEWFIRM focal plane each of two PANs handles the data for two detectors the detectors are arranged as shown in Figure 5. The black dots represent the first pixel produced in each detector.

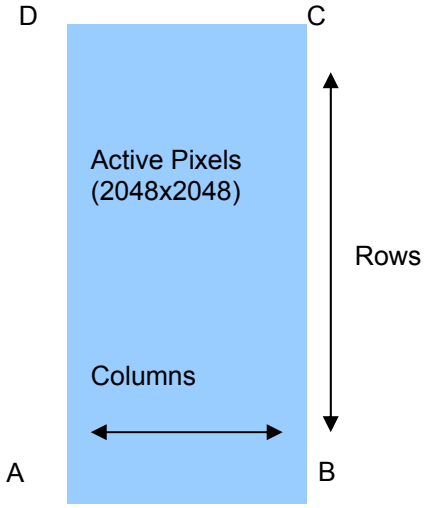


NEWFIRM Focal Plane
Figure 5

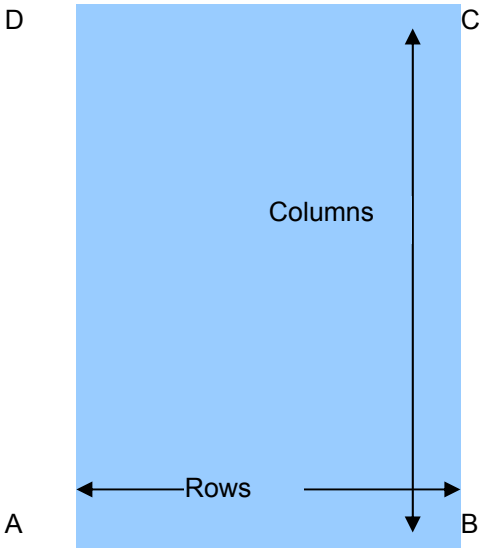
Each PAN in NEWFIRM sends four blocks of data to the DHS. When using the standalone FITS writer, the data from a PAN ends up in a single MEF file with the base header and four extensions. A DHS may do as it wishes with the four data blocks sent by means of the API.

The generic_CCD, ccdlab, genCCD_Mosaic and various specific ccd focal planes implemented for testing purposes are operating in the ccdlab in Tucson. These systems use the same DHS API to create images on the local disk or to send data to a DHS machine for archiving. To develop a consistent method for descrambling, which is consistent with the IR systems and allows the display of images from the focal plane, the CCD systems follow the descrambling technique outlined here.

The diagram in Figure 6 shows how MONSOON labels the outputs on standard 4-output CCD's. In MONSOON, rows are always in the parallel shift direction and columns are always the serial shift direction regardless of how the detector is arranged on the focal plane.

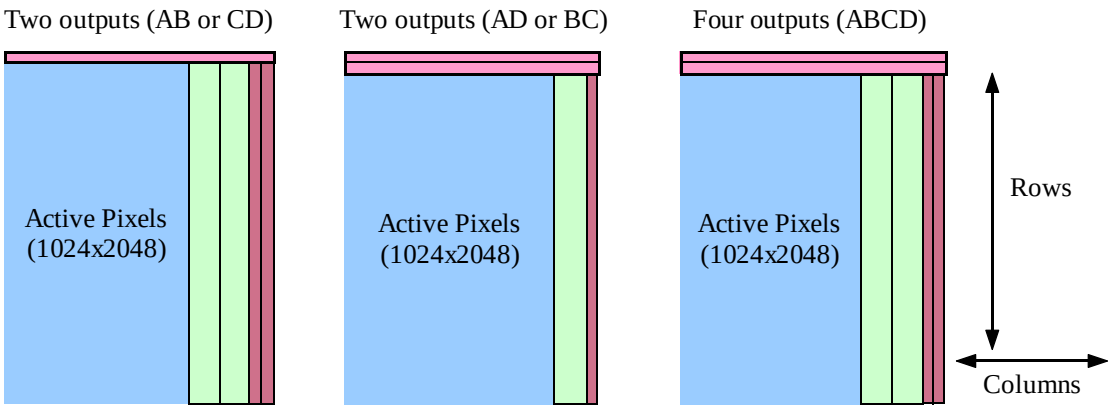


CCD Output Designation
Figure 6



Sample CCD Focal Plane
Figure 7

The archiving creates a separate extension for each CCD in the focal plane, that is, the focal plane in Figure 7 would result in Three blocks of data being sent to the DHS, the Before and After exposure meta-data and one image data extensions. During descrambling, the pre and post scan columns are moved to the right edge of the image and the post scan rows to the top of the image as shown in Figure 8. Assuming 40 column pre-scans, 20 column post-scans and 20 row post-scans, the final images would be 1144x2068 for the AB/CD case, 1084x2088 for the AD/BC case and 1144x2088 for the ABCD case.



Descrambling
Figure 8

Each focal plane has its own descrambling algorithm (detDescramble) included in the detector/focal plane library libdetCmnds.so. This allows the MONSOON system to handle arbitrary data streams from a focal plane. It is assumed that the order of pixel generation will not change from one exposure to the next. However, the captMode, procMode, expMode attributes have been provided to handle even varying pixel orders. In addition, the implementer of a MONSOON system can modify this scheme at the cost of writing new descrambling and image saver algorithms for the new descrambling technique.

Appendix VI Code Development on a Non-PAN Machine