Design page for DM-1112

Target release	Winter 2015
Epic	⚠ DM-1112 - Jira project doesn't exist or you don't have permission to view it.
Document status	DRAFT
Document owner	Simon Krughoff
Designer	Simon Krughoff
Developers	Simon Krughoff
QA	

Goals

- Develop a tool to construct Camera objects from collections of fits files (or MEFs).

Background and strategic fit

The camera team would like to use the DM stack to analyze their test data. They can do this using obs_file, or by defining Camera objects by hand, but it would be much nicer to have a tool to create the appropriate Camera object from the data directly.

Requirements

#	Title	User Story	Importance	Notes
1	Support both amp per file and sensor per file with amps in extensions. There are actually 4 cases: 1. amp per file 2. chip per file amp per extension 3. chip per file raw mosaic 4. camera per file chip per extension	The data are currently in MEFs, one per sensor. This could change so should be flexible.	Must have	
2	Should use a minimum of external ancillary information.	The goal will be to create the Camera only using primarily information in the FITS headers. Amp and Detector information will be taken from the header with some values possibly given global defaults in the config. Camera information will be supplied in a config to the task.	Must have	• The camera team are using the NOAO mosaic keywords: `http://iraf.noao.edu/projects/ccdmosaic/imagedef/fitsdic.html`

User interaction and design

The current design of the image format from the camera team is here: https://confluence.slac.stanford.edu/display/LSSTCAM/Draft+File+Specification+for+EO+Test+Images

Also see the notes from an Oct 8 meeting

This could be informed by the DM effort to implement this tool. I'm first going to design the chip per file with an amp per extension. We can see how that fits with a more general design.

Design

- Inherit from CmdLineTask. This provides --config and --configfile command line options, but will also be callable
  - --showCamera: will display the camera in ds9
  - --writeCamera=destination: will save the camera description so it can be reused later.
  - It may not be possible to inherit directly from CmdLineTask since it requires a repository. In that case, we will add a slimmed down ArgumentParser that doesn't require a mapper. This may be generally useful anyway.
- Per chip info will be in the header.
  - The assumption is that the chip level info will be in the Primary Header
  - Unfortunately this will require some custom keys (indicated with *)
  - We need to make sure we are not colliding with accepted standards.
- Camera level info will come from a config (or specified with the --config option on the command line).
- Mapping between key names will happen via a key map taken from the config. This will simply by a dictionary mapping default name (key) to name in the data (value).
- There will also be a preprocessing step that will allow for more complicated mapping as well as filling the custom keys defined here at runtime.
- This utility will not produce a valid repository. I think it should be the job of an ingest.py script to do that.

Amp info mapping

I don't completely understand the different coordinate systems available. See: http://iraf.noao.edu/projects/ccdmosaic/imagedef/imagedef.html

I believe DTM/DTV are the ones we should assume are default, but I'm happy to be corrected.

Header Key	AmpInfo	Description	Default
EXTNAME	name	Name of Amp: '0,1'	
DETSEC	BBox	Bounding box of physical pixels in in assembled coordinates	
GAIN	Gain	Gain value of this amp e-/count	1.
RDNOISE	ReadNoise	Read noise in counts	0.
SATURATE	Saturation	Value of saturation threshold in counts	
DTM[1-4] check mod 90 rotation	ReadCorner	Location of first pixel read in assembled coordinates	LLC
LINCOEFF	LinearityCoeffs	Coefficients of linearity fit	0., 1.
LINTYPE	LinearityType	Type of linearity: This could map to a method for applying non-linearity correction	POLY
NAXIS1, NAXIS2	RawBBox	Bounding box of raw data (including prescan, overscan regions) in raw coordinates	
DATASEC	RawDataBBox	Bounding box of raw data in the raw frame	
DTM[1-4]	FlipX	Flip x axis when assembling?	False
DTM[1-4]	FlipY	Flip y axis when assembling?	False
DTV1, DTV2	RawXYOffset	Offset of to apply to assemble raw frames in a mosaic	0,0
BIASSEC[1]	HOverscan	Bounding box of horizontal overscan in raw coordinates	
BIASSEC[3]	VOverscan	Bounding box of vertical overscan in raw coordinates	Empty BBox
BIASSEC[2]	Prescan	Bounding box of prescan region in raw coordinates	Empty BBox

Detector info mapping

Header Key	Detector Config	Description	Default
CCDNAME	Name	Name of detector slot: R:22, S:11	
DETSIZE/CCDSIZE	BBox	Bounding box of physical pixels	guess from amps?
OBSTYPE	detectorType	Type of detector: SCIENCE, GUDER This can be extended.	SCIENCE
SERSTR*	serial	String serial identifier for the installed device	'none'
XPOS*, YPOS*	offset_[xy]	Offset of the chip from the origin in physical coordinates (mm)	0.0
XPIX*, YPIX*	refpos_[xy]	Position on the chip to which the offset refers	LLC
YAWDEG*	yawDeg	rotation of the detector about z axis	0.0
PITCHDEG*	pitchDeg	rotation of the detector about y axis	0.0
ROLLDEG*	rollDeg	rotation of the detector about y axis	0.0
XPIXSIZE*, YPIXSIZE*	pixelSize_[xy]	Size of a nominal pixel in physical coordinates (mm)	
TRNSPOSE*	transposeDetector	Transpose the pixel grid before orienting in the focal plane?	False

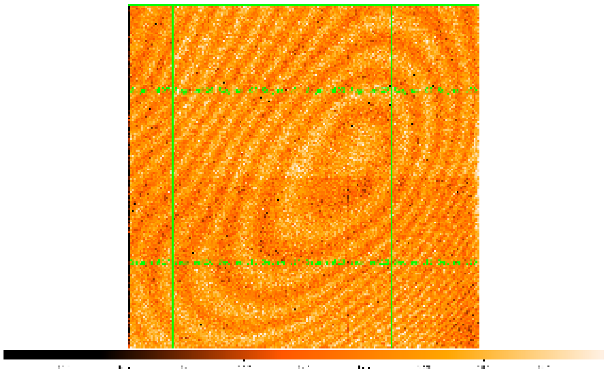
These could mostly be handled with a WCS, but it is fairly rare to have a WCS that goes to focal plane coordinates and if it exists, it can be used to fill in these key words.

Camera info mapping

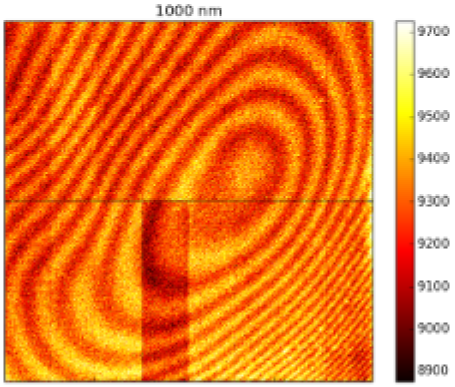
Camera Config	Description	Default
name	Name of the camera	'FileCamera'
plateScale	plate scale at the focal plane (arcsec/mm)	1.
radialCoeffs	radial coefficents that describe a radial polynomial distortion	I think the following produces no distortion [0, 1, 0]

Implementation of use case

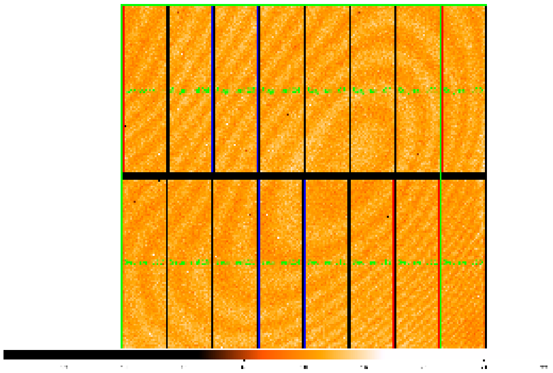
I implemented the plotting use case outlined above. The code can be found here. The results from commit 102320d are all shown below.



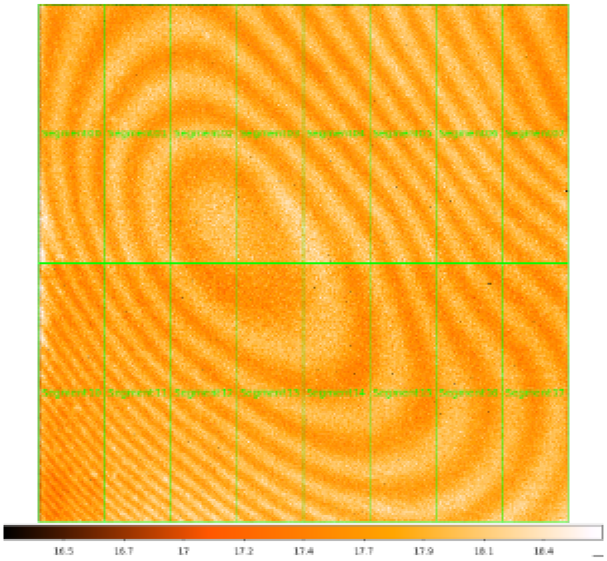
This shows an assembled flat provided by Jim C. The segments are labeled. Note that a bias correction and rudimentary gain correction have been applied. This agrees with an assembled flat he provided shown here:



Below is plotted an untrimmed version. The green box are raw data boundaries, the red box is the overscan region and the blue box is the data region.



The above images were produced using bounding boxes and offsets calculated in the code. This makes the plotted images match the orientation from Jim's example. If I use the bounding boxes in the header (commit 280194e), I get a good assembly, but it is flipped about the y-axis. I'm not sure how important that is or what I'm missing in the headers, but I believe that the above orientation does not imply that the serial direction is +ve in the +ve x direction.



Questions

Below is a list of questions to be addressed as a result of this requirements document. These are just here for historical reasons. These answers will be fleshed out in the design above.

Question	Outcome

Some of the necessary data will have to be input by the user. How should we do that? • config file • command line • subclass • something else	The result will be that we take any camera level info as config params. The rest will come from image headers. We are defining some custom header keys but they will be defaulted. Any key can be remapped using a mapping dictionary from a config file. More complicated mapping will be done through a processing step.

Not Doing