Visible & Infrared Survey Telescope for Astronomy

Data Flow System

Document Title: **VIRCAM DFS needs in QFITS & CPL**

Document Number: **VIS-SPE-IOA-20000-0008**
Issue: **0.5**

Date: **2004-09-17**

Document Prepared by:	**Peter Bunclark** (CASU)	**Signature And date:**	
Document Reviewed by:	**Jim Lewis** (CASU Software engineer)	**Signature And date:**	
Document Approved by:	**Mike Irwin** (CASU Manager)	**Signature And date:**	
Document Approved by:	**Malcolm Stewart** (VDFS Manager)	**Signature And date:**	
Document Released by:	**Jim Emerson** (VDFS Project leader)	**Signature And date:**	

Change Record

Issue	Date	Sections Affected	Remarks
0.5	2004-09-17	All	New Document

Contents

1 Introduction

1.1 Scope

At the VISTA IR Camera DFS PDR the team was encouraged to consider using the QFITS [RD3] and the Common Pipeline Library (CPL) [RD5] infrastructure, and to indicate any areas where their functionality was insufficient for VDFS purposes. This discussion notes those areas where QFITS and CPL do not currently meet VDFS needs or expectations, and suggests potential solutions for discussion with ESO prior to submission of any formal SPRs.

1.2 Acronyms and Abbreviations

CCD	Charge-Coupled Device
CFITSIO	C (language) FITS Input/Output
CPL	Common Pipeline Library
FITS	Flexible Image Transport System
IR	Infra Red
MEF	Multi-Extension FITS file
QE	Quantum Efficiency
QFITS	Quick FITS library
SPR	Software Problem Report
VIRCAM	VISTA IR Camera
VISTA	Visible and Infrared Telescope for Astronomy

1.3 Reference Documents

[RD1] *VISTA Infra Red Camera DFS Data-Reduction Specifications, VIS-SPE-IOA-20000-0003, issue 0.5, 2004-04-08*

[RD2] *WCSLIB web site,* http://www.atnf.csiro.au/people/mcalabre/WCS

[RD3] *QFITS Reference Manual, http://www.eso.org/projects/aot/qfits/html, issue 5.0.0* (VLT-MAN-ESO-19500-2722).

[RD4] *CFITSIO – A FITS File Subroutine Library, http://heasarc.gsfc.nasa.gov/fitsio, version 2.490*

[RD5] *The Common Pipeline Library, http://www.eso.org/observing/cpl*, version 1.0.1.

[RD6]

2 Overview

The design of the VISTA Data-Flow System pipeline modules, set out in the VISTA Data-Reduction Specification [RD1], assumes a sophisticated middle-layer of infrastructure software. The PDR design stores data at all stages as Multi-Extension FITS files, and the current pipeline design expects to access the pixel data itself, and use the FITS header records as a dynamic state database, using CFITSIO [RD4] for file-handling.

It was suggested that we might adopt ESO's QFITS and CPL as middle layer infrastructure, and so we discuss here what is needed to remedy those problems we forsee in using the current QFITS and CPL for the VISTA IR Camera DFS.

For each perceived QFITS or CPL problem we suggest a remedy, and consider these under the headings of

- new features
- enhancements to existing features
- software engineering and documentation issues.

3 QFITS - New Features

3.1 Tile Compression

Proposal: Implement 'tile compression' in QFITS

Reason: The term **tile compression** refers to a data compression scheme that is currently awaiting ratification by the FITS community into a true FITS standard. In it the data array is split into small pieces (**tiles**) and each tile is compressed using a standard entropy-based encoding algorithm. The result is stored in a binary FITS table with the appropriate indexing information, so that, when required, the relevant part of the FITS image can be accessed, decompressed and presented to the user. It is expected, based on tests with similar data, that tile-compression using the Rice algorithm will losslessly compress the 32-bit VISTA IR data by roughly a factor of 4. This compression ratio translates into a huge cost and efficiency saving; archiving is cheaper, data-transfer is faster, and even processing is faster as the on-the-fly decompression is faster than the I/O time for uncompressed data.

Tile compression has a further advantage over standard file-compression algorithms (zip etc) in that the original FITS header is unchanged and accessible in the usual way.

4 QFITS - Enhancements to existing features

4.1 Common Syntax for accessing Extension and Subsets

Proposal: Allow access to subsets of FITS images using filename extensions

Reasons: It is common practice to allow access to subsets of FITS images using filename extensions as follows:

- fff.fits - address the whole file
- fff.fits[1] - address extension 1 of a MEF
- fff.fits[100:150,200:237] - address area within a non-extended file
- fff.fits[1] [100:150,200:237] - address area within a certain extension

At present QFITS only allows you to address an extension by explicitly setting the **qfitsloader** structure variable **xtnum**. There is no true support for subset access. QFITS can tell you the location of the data segment and its size, so if the user wishes to read a data subset it is up to the programmer to work out the offsets and lengths required and to read them explicitly. This is clumsy and the package would benefit enormously from the ability to do both extension and subset access internally. Using the sort of notation above, a simple parser could be written to access the information required. Ideally this would be available to both the loader and the dumper routines, although the task would be significantly more difficult for the latter.

Examples of the convenience of this utility are handling individual readout channels in the case of IR detectors, and in bias-strip analysis in the case of CCD reduction.

4.2 Short Integer Data Type

Proposal: Allow short integer type

Reason: Currently QFITS only allows float, double and integer data types to be retrieved from the data loading routines. Confidence maps require short integer type, and while this could quite easily be accomodated with the current integer data type, it is wasteful of memory and doubles I/O, and this could be a real issue in (for example) deep stacking software, where many images and their associated confidence maps would need to reside in memory simultaneously.

5 QFITS – Software Engineering

5.1 Error Status Handling

Proposal: Add inherited status pointer as last parameter in every call to QFITS

Reason: The error handling in QFITS is weak. The onus is very much on the application programmer to track errors. The situation could be remedied by consistently using an inherited status pointer as the last parameter in every call. Each procedure should check the current status immediately on entry, and if non-zero, return. In this way a) the high-level programmer always can rely on a status being the final parameter, and b) several (perhaps simple) calls can be run in succession before checking the status for possible action, confident in the knowledge that the error value contained in the variable reflects the original problem that the package encountered.

5.2 User's Guide

Proposal: Produce a QFITS programmer's guide

Reason: The QFITS reference manual [RD3] is sufficient as a reference, but not as a programmer's guide. Far too little information is included in section 2.6 in particular, regarding the routines themselves and their arguments. At the moment, searching through QFITS code is the only way of answering many questions and this is extremely inefficient (not to mention frustrating). A much fuller guide in the same mode as the CFITSIO User's Reference Guide [RD4] would make a huge difference, considerably shortening the software life-cycle of new projects.

6 CPL – New Features

6.3 World Coordinate System

Proposal: Allow use of WCSLIB with CPL and check compatibility.

Reason: Production pipelines will need to use World-Coordinate System handling code. The most cost-effective way to implement WCS in FITS is by using the excellent WCSLIB routines written by Marc Calabretta [RD2]. This is well written code which covers all the conventions and projections that are part of the FITS standard. This in itself does not need an actual enhancement for CPL, but rather needs verification of no incompatibilities or side effects.

6.1 Catalogue Interface

Proposal: Enable access to standard catalogues, e.g. 2MASS and USNO-B, in CPL.

Reason: VDFS will use 2MASS and USNOB catalogues extensively. The functionality will be provided by an 'extract_cat' procedure, which will take the catalogue required, and a coordinate range, as parameters, and return a FITS table of results in a fixed format. This interface could either get built in to CPL or be provided as one of the VDFS modules. The catalogues themselves should be stored locally in their published form, allowing multi-use by other software which may be able to understand their format.

7 CPL – Enhancements to existing features

7.1 Confidence Maps (vs bad-pixel maps)

Proposal: Allow use of (16-bit) confidence maps in image combining routines instead of the current (1-bit) bad pixel maps.

Reason: CPL has image-combining routines which use an effectively 1-bit (sparsely-stored) bad-pixel map. When images are combined, any input pixel which was marked bad causes a corresponding bad pixel in the output image.

Because infra-red processing inherently requires combining multiple exposures (jitters) for the very purpose of eliminating bad pixels, the pipeline requires the use of a 'confidence map'. Each good pixel is initially given a confidence value of ~100%, modulated by the relative qe of the pixel plus telescope plus camera modulation function with respect to the median, and each bad pixel 0%. For example, when a good pixel from one image is combined with a bad pixel from another, the normalised confidence value of the output pixel will be 50%. During image-combination operations, a weight-map is generated by scaling the confidence map by the background variance, stored in the header. A confidence map is thus a superset of a traditional bad-pixel map.

8 CPL – Software Engineering

No changes proposed.

_o☺o_