Created: February 19, 1999
Modified: March 1, 1999

Gemini Wavefront Sensing System Report

ospLib- Optical Signal Processing Library for Wavefront Sensing

Steven Heddle

wfs_sbh_004/01

This document is the reference manual for ospLib

1.0 Introduction

The signal processing library for the wavefront sensors is contained in three main files, ospLib.c, ospInvertFuncs.c and the header file osp.h. The bulk of the functions are contained in ospLib.c, with ospInvertFuncs.c containing only the SVD related functions, and functions to perform matrix and vector operations. The functions in ospInvertFuncs.c without an osp prefix have been taken from *Numerical Recipes in C*, and the necessary header file support for these functions has been incorporated into osp.h. The signal processing library makes use of the cfitsio library for input and output of test and calibration data in FITS format. The library may be run under VxWorks or Unix.

The library has been designed to support the operation of peripheral and on-instrument wavefront sensors to provide fast guide data (tip, tilt and focus) to the reflective memory bus, or active optics data (includes higher order Zernike coefficients) at a slower rate to the telescope control system. The raw data frames of Shack-Hartmann spots to be centroided are supplied by an SDSU-2 CCD controller, which allows the readout geometry of the CCD to be specified by the user, subject to the symmetry constraints imposed by the individual sectors of the CCD array. The operation of the library functions is dependent on a number of mode (i.e. fast guide or aO) dependent variables, and a number of variables describing the readout geometry, so in order to manage function arguments, and ensure reentrancy of shared code used by a number of wavefront sensors simultaneously, we invoke the concept of a context structure which allows the elimination of global and static variables which might be modified. A structure of this type (struct OSP_CONTEXT) is created for each mode of each wavefront sensor, and its pointer passed as an argument to the ospLib functions where appropriate. A more detailed description of the context structure occurs at the end of this document, but we note that the structure also stores calibration data and results from application of the library functions, including ultimately Zernike coefficients and their standard errors. An additional structure (struct OSP_GEOM) is used to implement changes of CCD readout geometry.

2.0 Summary of ospLib functions

The ospLib functions are listed here in approximate functional groupings. The source code is in ospLib.c unless stated otherwise.

2.1 Initialisation

ospInit : To create and initialise a wfs context structure

ospNewReadCentres : To calculate coordinate data for the subaperture null positions from data stored in file

ospReadNulls : To read the coordinates of the null positions from a file

ospTidyUp : To free all the memory allocated from the corresponding ospInit()

2.2 Measurement

ospFGMeasure : To calculate Zernike coefficients from a frame of SH spots, and write them to the synchro bus

ospFGCentroidWrapper : To perform offset correction, thresholding and centroiding on a data frame quickly for fast guiding

ospMeasure : To accept an input frame of SH spots and calculate the Zernike coefficients

ospThreshold : To estimate and/or impose a threshold for each subaperture to reduce the effects of noise

ospCentroidWrapper : To threshold the frame and repeatedly apply the ospCentroid() function to every subaperture with a specified centre

ospCentroid : To calculate the centroid position and its standard deviation for a subaperture

ospCoAdd : To co-add frames, either a specified number or timeout period's worth

ospGlobalThreshold : To apply a common threshold to an entire frame

ospGlobalCentroid : To determine the centroid of an entire frame, in pixels

ospMeasVars : To estimate the measurement variances of the calculated Zernike coefficients

ospAddFrameToFrame : To add a frame to an existing frame of the same size, pixel by pixel

ospSubtractFrameFromFrame : To subtract a frame from an existing frame of the same size, pixel by pixel

ospMultiplyFrameByFrame : To multiply a frame by an existing frame of the same size, pixel by pixel

ospAddConstantToFrame : To add to each pixel in a frame the same specified amount

ospMultiplyFrameByConstant : To multiply each pixel in a frame by the same specified amount

2.3 Readout geometry change

ospCalculateSubaps : To calculate the coordinates of the subapertures and null positions wrt to the readout geometry

ospChangeGeometry : To adapt the wfs context to changes in detector readout geometry

ospReduceFrame : To reduce the frame size to the minimum containing all subaperture pixels

2.4 Calibration

ospCalibrate : To populate a reconstructor matrix and invert to obtain control matrix

ospSaveNullPositions : To create or update null positions data from a calibration file

ospFitVars : To estimate the fitting variances of the calculated Zernike coefficients

ospGetBasisFunction : To populate the matrix which defines an arbitrary basis set in terms of Zernike polynomials and their coefficients

ospNewCalibrate : To populate a reconstructor matrix and invert to obtain control matrix

ospNullCorrection : To provide a simple interface by which the null positions file can be corrected using a FITS image of nulled spots

2.5 Testing

ospTestData : To generate a known test frame of data

ospSimulateCentroids : To generate centroid data for a specified combination of Zernike functions

ospFrameScramble : Scramble an entire frame of data

ospFrameTypeConvert : To convert a frame of floats to a frame of unsigned short ints, or vice-versa

ospConvertAndScrambleFITS : To convert a float FITS image to ushort int, scramble it and write it to file

2.6 FITS application code

The following functions require the cfitsio library available from

http://heasarc.gsfc.nasa.gov/fitsio

ospReduceFits : To generate a reduced frame FITS file from a full frame FITS file

ospReadHeaderInt : To read the integer values of an array of keywords from a FITS header

ospAddContextToHeader : To write ccd readout geometry to a FITS header

ospPrintHeaders : To print out all the header keywords in all extensions of a FITS file

ospReadFloatImage : To read a float image into a buffer from a FITS file

ospReadUShortImage : To read an unsigned short int image into a buffer from a FITS file

ospPrintError : To report cfitsio errors, as related to the status codes in fitsio.h

ospWriteFloatImage : To write a float image to a FITS file

ospWriteUShortImage : To write an unsigned short int image to a FITS file

2.7 Matrix and vector handling

These are a number of general functions to ease the manipulation of matrices and vectors, and allow reading and writing from file. The source code is in ospInvertFuncs.c. These functions are not documented in the next section, but are commented in their source.

ospReadVectorFromFile : To read a vector of specified size from a file

ospWriteVectorToFile : To write a vector of specified size to a file

ospReadMatrixFromFile : To read a matrix of appropriate format from a file

ospWriteMatrixToFile : To write a matrix to a file

ospApplyControlMatrix : To perform the matrix-vector multiplication that yields the Zernike coefficients

ospCalculateInverse : To calculate the inverse matrix from the matrices resulting from singular valued decomposition

ospConditionOfW : To calculate the condition number of the singular valued decomposition

ospMatrixProduct : To form a matrix product

ospShowMatrix : To display the elements of a matrix

ospShowVector : To display the elements of a vector

2.8 Functions to communicate data to TCS and synchro bus

Functions written by Sean Prior to communicate the Zernike coefficents and their uncertainties to the TCS or reflective memory bus. The source is in writeZernikes.c, but the functions are prototyped in osp.h and are thus listed here for completeness. They are not documented in the next section.

writeWfsToTcs : To write Zernike data to the TCS

writeWfsToSynchro : To write Zernike data to the reflective memory bus

2.9 Functions adapted/adopted from Numerical Recipes

These functions are described fully in Numerical Recipes in C (ISBN 0-521-43108-5), and are not documented in the next section.

The first two functions are subject to copyright. The last four functions are part of the Numerical Recipes Utility Routines which have been placed in the public domain and are freely usable. The memory allocation in **vector** and **matrix** has been changed to use calloc() rather than malloc().

pythag : calculate the square root of (a squared + b squared)

svdcmp : singular valued decomposition of a matrix

vector : allocate memory for a vector

matrix : allocate memory for a matrix

freeVector : deallocate memory for a vector

freeMatrix : deallocate memory for a matric

2.10 Other

ospShow : To display the current values of the wfs context

The following simple functions are used by the Numerical Recipes routines, and were formerly implemented as global declarations. They are not documented in the next section, and their source is in ospInvertFuncs.c

SQR : To form the square of its floating point argument

IMIN : To establish the minimum of its two integer arguments

FMAX : To establish the maximum of its two floating point arguments

3.0 Description of ospLib functions

The functions are listed here in alphabetical order.

3.1 ospAddConstantToFrame - To add to each pixel in a frame the same specified amount

INVOCATION :

ospAddConstantToFrame(buffp1,constname,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp1	float *	pointer to buffer to be added to
>	constname	float	additive constant
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To add to each pixel in a frame the same specified amount

DESCRIPTION : Adds to each pixel in an existing frame buffer pointed to by buffp1 the floating point value constname. The frame is assumed to be wfsSpecific->xframesize by wfsSpecific->yframesize No checking of the frame size is done, due to the function's intended use of making corrections to fast guide data frames, which requires low latency.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the pointer points to a buffer which is already allocated and large enough.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.2 ospAddContextToHeader - To write ccd readout geometry to a FITS header

INVOCATION :

ospAddContextToHeader (filename,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	filename	char *	name of FITS file to which we add header info
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To write ccd readout geometry to a FITS header

DESCRIPTION : Opens FITS file for writing, and writes the following keywords and comments into the header (values included by way of example):

XSTART = 11 / [pixels] ospxstart
YSTART = 11 / [pixels] ospystart
XBIN = 1 /pixel binning factor in x dirn.
YBIN = 1 / pixel binning factor in y dirn.
XRASTER = 8 / number of x superpixels across subap.
YRASTER = 8 / number of y superpixels across subap.
XSPACE = 2 / [pixels] number in x between subaps.
YSPACE = 2 / [pixels] number in y betweem subaps.
XSUBAP = 3 / number of x subaps. per output
YSUBAP = 3 / number of y subaps. per output
OSP_FSZ = 1 / framesizeflag (0:reduced,1:fullframe)
OSP_NP = 14 / no. of Zernike coeffs. calculated
OSP_MP = 64 / 2 x total number of subaps used in array
OSP_SIDE= 8 / [pixels] side length of subap
OSP_NSIG = 3.000000E+00 / nsigma
OSP_RDSQ = 3.600000E+01 / [e- **2] read noise per pixel sqared
OSP_WT = 0 / weighting applied? (0:no, 1:yes)
OSP_THR = -1.000000E+00 / argument supplied to ospThreshold

The FITS file is then closed.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : None known

3.3 ospAddFrameToFrame - To add a frame to an existing frame of the same size, pixel by pixel

INVOCATION :

ospAddFrameToFrame(buffp1,buffp2,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp1	float *	pointer to buffer to which frame is to be added
>	buffp2	float *	pointer to buffer containing frame to be added
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To add a frame to an existing frame of the same size, pixel by pixel

DESCRIPTION : Adds a frame pointed to by buffp2 to an existing frame buffer pointed to by buffp1. The frames are both assumed to be wfsSpecific->xframesize by wfsSpecific->yframesize. The addition is done pixel by pixel. No checking of the frame size is done, due to the function's intended use of making corrections to fast guide data frames, which requires low latency.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the pointers point to buffers which are already allocated and large enough.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.4 ospCalculateSubaps - To calculate the coordinates of the subapertures and null positions wrt to the readout geometry

INVOCATION :

ospCalculateSubaps(wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To calculate the coordinates of the subapertures and null positions wrt to the readout geometry

DESCRIPTION : The positions of the bottom left hand corners of the subapertures (in pixels) are calculated using the ccd readout geometry stored in the context structure, with respect to either a full frame or reduced frame as appropriate, and the null positions of each subaperture which is to be used are calculated relative to these coordinates. This data is then stored in wfsSpecific->centres[], with centres[0] storing the number of data items to follow (which is four times the number of subapertures for which the null positions are positive values, which is clearly not necessarily the same as four times the number of subapertures).The subsequent entries are centres[1] = x coordinate (in pixels) of bottom left hand pixel of first subaperture with positive coordinates for its null position. centres[2] = offset in x (in pixels) of null position from bottom left hand pixel of first subaperture... centres[3] = y coordinate (in pixels) of bottom left hand pixel of first subaperture with positive coordinates for its null position. centres[4] = offset in y (in pixels) of null position from bottom left hand pixel of first subaperture... and so on in groups of 4 for subsequent subapertures with positive coordinates for their null positions, i.e. centres[5] to centres[8] are the corresponding data for the second subaperture with positive coordinates for its null position. The null positions used in the calculation are those stored in wfsSpecific->nulls[], which use negative coordinates to indicate that a subaperture is not to be used. The ccd readout geometry defines the positions of the subapertures on the array, and the coordinates of the bottom left hand pixel of each subaperture are calculated with respect to an x-y coordinate system which applies to the whole array, and is not redefined for each sector. The calculation of the subaperture positions in each sector however takes account of the various symmetry properties of the sectors with respect to each other. Note that the definition of the centre of the bottom left hand pixel of the whole area of the array which is read out as (1,1) has been chosen for convenience when calculating the subaperture positions in the various sectors. The ccd readout geometry is defined in the following elements of the context structure:

sectors - number of

ospxstart - pixels in x skipped before first subap

ospystart - pixels in y skipped before first subap

ospxbin - binning factor in x for the pixels read out

ospybin - binning factor in y for the pixels read out

ospxraster - x size of the subapertures in binned pixels

ospyraster - y size of the subapertures in binned pixels

ospxspace - pixels in x between subapertures

ospyspace - pixels in y between subapertures

ospxsubap - number of subapertures readout PER SECTOR in x

ospysubap - number of subapertures readout PER SECTOR in y

xarraysize - x size of the frame supplied to the centroiding func.

yarraysize - y size of the frame supplied to the centroiding func.

framesizeflag - flag for full frame (1) or reduced frame (0) image

which correspond with the geometry shown most clearly in Gemini Newsletter #16, June 1998, p12. Currently there is no provision for binning (i.e. xbin=ybin=1), and as the null positions in wfsSpecific->nulls are relative to the same full frame coordinate system, the calculation of the offsets in x and y are simple differences. If binning is employed, some scaling will also be necessary when calculating the offsets in x and y of the null positions. The calculation of the centres[] data proceeds in one of two distinct ways dependent on whether wfsSpecific->framesizeflag is 1 (indicating that a full frame is used) or 0 (indicating that a reduced frame of only subaperture pixels is used). If framesizeflag = 1, our calculations are complete and we have populated the centres[] array. If framesizeflag = 0, we need to recalculate the subaperture coordinates to reflect that only the pixels which constitute the subapertures (including the unused subaps) make up the reduced frame. The offsets of the null positions in x and y are unchanged. The offsets in x and y of the null positions for each subaperture are checked to ensure that they are less than a quarter of the width of the subaperture from its geometric centroid, and a warning flagged if this is not the case- the function continues but exits with an ERROR value. An error value is also returned if the number of subapertures which are to be used (which have positive coordinates for their null positions) does not equal (wfsSpecific->mp/2), which is specified when the context structure is initialised. If a subaperture has negative coordinates for its null position this is reported- this is not an error. The data thus calculated is used by the thresholding and centroiding functions.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the null positions have been read in using ospReadNulls, called either by itself or as a part of ospInit or ospCalibrate.

INCLUDE FILES : osp.h

DEFICIENCIES : No provision yet for binning of pixels

3.5 ospCalibrate - To populate a reconstructor matrix and invert to obtain control matrix

INVOCATION :

ospCalibrate(calibpath, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	calibpath	char *	path to directory of calibration data
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To populate a reconstructor matrix and invert to obtain control matrix

DESCRIPTION : Calibration frames of displaced SH spots corresponding to linearly ide-pendent combinations of known Zernike aberrations are used to form a reconstructor matrix with respect to the basis functions defined by the linear combinations. This reconstructor matrix is transformed into one with respect to the individual Zernike polynomials of unit magnitude, and further inverted to provide the necessary control matrix which is applied to the SH spot displacements in an arbitrary input frame to yield the Zernike coefficients. The control matrix is written to a file specified by the character string wfsSpecific->controlfile, and the calculated fitting variances written to a file specified by the character string wfsSpe-cific->fvarsfile. If either of these files exist, they are copied to a file with a .bak extension added, befeore the new version is created.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : Not general enough-need to prompt for data input, basis functions etc.

Superceded by ospNewCalibrate, which prompts for necessary values.

3.6 ospCentroid - To calculate the centroid position and its standard deviation for a subaperture

INVOCATION :

ospCentroid(buffp,x0,xdiff,y0,ydiff,disp,mean,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	float *	pointer to input frame buffer
>	x0	int	x coord of bottom left pixel of subap
>	xdiff	float	x offset of null position from x0

>	y0	int	y coord of bottom left pixel of subap
>	ydiff	float	y offset of null position from y0
<	disp	float *	4 element array returning centroid isplacement and variance
>	mean	float	value subtracted from above threshold pixels
>	wfsSpe-cific	struct OSP_CONTEXT *	wfs context structure

FUNCTION VALUE : void

PURPOSE : To calculate the centroid position and its standard deviation for a subaperture

DESCRIPTION : The centroid position in a subaperture is calculated using moments, and the standard deviation estimated, taking into consideration the uncertainty estimated in the choice of threshold value for that subaperture. x0,xdiff,y0,ydiff may be obtained as four consecutive elements of the wfsSpecific->centres[] array, taking care to cast x0 and y0 as ints. Knowledge of (x0,y0) allows the function to index to the correct start point in the frame for the subaperture, and thence to calculate the moments for each pixel relative to the null position as specified by xdiff and ydiff in the subaperture, given knowledge of the subaperture geometry (e.g. size) from wfsSpecific. The displacements of the centroid in x and y are given by nomalising the x and y sums of the moments by the total intensity in the subaperture. If, for whatever reason, the total intensity in a subaperture should be zero, the displacements are set to zero and the standard deviations set to 99. The displacements and variances of the spot centroid are returned through the disp[] array: disp[0] = x displacement disp[1] = variance of disp[0] disp[2] = y displacement disp[3] = variance of disp[2] N.B. if a subaperture has no light in it, its centroid x and y displacements are set to zero, and the variances are returned as 99.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.7 ospCentroidWrapper - To threshold the frame and repeatedly apply the ospCentroid() function to every subaperture with a specified centre

INVOCATION :

ospCentroidWrapper(buffp,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to input buffer containing image of SH spots
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To threshold the frame and repeatedly apply the ospCentroid() function to every subaperture with a specified centre

DESCRIPTION : Reads the array of centres and invokes ospCentroid() for each complete set of x0, xoffset, y0, yoffset, and stores all the centroid positions as a vector wfsSpecific->s of x and y displacements of the centroid and a vector wfsSpecific->dssq of the x and y variances. If the weight argument is set to 1, the x and y displacements in s are replaced by values weighted by the reciprocal of the respective standard deviations. NB this weighting will be changed to something more...correct.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999 The weighting (in this sense) has been removed, but not yet replaced. (27/1/99, SBH)

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That ospCalculateSubaps has generated the centres[] array.

INCLUDE FILES : osp.h

DEFICIENCIES : Need to read the return values from ospCentroid and ospThreshold

3.8 ospChangeGeometry - To adapt the wfs context to changes in detector readout geometry

INVOCATION :

ospChangeGeometry(ospGeom,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	ospGeom	struct OSP_GEOMETRY *	structure containing modifications to the detector readout geometry
!	wfsSpecific	struct OSP_CONTEXT *	wavefront sensor context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To adapt the wfs context to changes in detector readout geometry To allow the wfs software to adapt correctly to changes in the detector readout geometry, to the values specified in ospGeom.

DESCRIPTION : Updates the wfs context structure, checks for consistency of the new parameters, recalculates subtractive and multiplicative subframes to correspond with new readout geometry. If wfsSpecific->framesizeflag = 1 the ccd readout geometry reverts to the default values, i.e. the values read in from the appropriate .ini file when the wfs context structure was created by ospInit- with the exception of the value of framesizeflag which may have been set to zero in the .ini file, but now will be 1.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known.

3.9 ospCoAdd - To co-add frames, either a specified number or timeout period's worth

INVOCATION :

ospCoAdd (buffp, N, deltaT,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	float *	pointer to buffer for frame to be added
>	N	int	Number of frames to be co-added
>	deltaT	float	Timeout period for summation of frames in seconds
!	wfsSpe-cific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To co-add frames, either a specified number or timeout period's worth

DESCRIPTION : Is called repeatedly to coadd N frames, with the first call (as indicated by a frame counter, which is part of the wfs context structure, being set to zero) initialising a frame buffer with the input frame, starting a clock and incrementing the frame counter. The frame counter is initially set to zero either by ospInit or the successful conclusion of a series of coadds. Repeated calls check the clock, and if the timeout period has not been reached,

the current input frame is added to the coadded frame buffer and the frame counter incremented, until the specified number of frames is reached. The start time and counter are stored as part of the wfs context structure. When either the specified number of frames or timeout is reached, the coadded frame buffer has each pixel divided by the number of frames, and ospMeasure is called to calculate Zernikes etc. The Zernike coefficients and their estimated errors are written to the TCS, and the frame counter is set to zero so the next cycle can begin.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h time.h

DEFICIENCIES : Need to get the return values of ospMeasure and writeWfsToTcs! Also the clock() function may not be defined in VxWorks, and thus the timeout will not work. In VxWorks the time should be derived from the Bancomm card.

3.10 ospConvertAndScrambleFITS - To convert a float FITS image to ushort int, scramble it and write it to file

INVOCATION :

ospConvertAndScrambleFITS(infile, outfile, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

infile	char *	name of input file
outfile	char *	name of output file
wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR.

PURPOSE : To convert a float FITS image to ushort int, scramble it and write it to file This was written initially to generate test data of the same format as would be expected from a ccd after a frame callback, from our readily available unscrambled float images.

DESCRIPTION : The input file is opened and the NAXIS1 and NAXIS2 keywordss read from the FITS header to check if the dimensions are consistent with the current detector readout geometry as stored in the context structure. If not, the function exits with a warning; if so, two buffers of the appropriate size are allocated, one for the input float image, the other for the output scrambled unsigned short int image. The input image is read into its buffer from infile, and then both scrambled and converted to unsigned short int by a call to ospFrameScramble. The resultant image is written to a FITS file, and the current detector readout geometry added to the header. The buffers are freed.

Author : Steven Heddle, UKATC, Edinbu>rgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : none known

3.11 ospFGCentroidWrapper - To perform offset correction, thresholding and centroiding on a data frame quickly for fast guiding

INVOCATION :

ospFGCentroidWrapper(buffp,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to input buffer containing image of SH spots
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To perform offset correction, thresholding and centroiding on a data frame quickly for fast guiding

DESCRIPTION : Performs subtraction and multiplication of offset frames, thresholding and centroiding of the resultant frame, and estimation of the variances of the centroid positions. These calculated values are stored directly in the wfsSpecific->s[] and wfsSpecific->dssq[] arrays of the context structure. NB if a subaperture has no light in it, the centroid position is not updated, which may be advantageous for minor glitches, but the variances are returned as 99 for the subaperture in this case.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That ospCalculateSubaps has generated the centres[] array.

INCLUDE FILES : osp.h

DEFICIENCIES : Corner averaging of subap pixels for thresholding is not available, error estimation is simplified

3.12 ospFGMeasure - To calculate Zernike coefficients from a frame of SH spots, and write them to the synchro bus

INVOCATION :

ospFGMeasure(buffp,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	float *	input frame buffer
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To calculate Zernike coefficients from a frame of SH spots, and write them to the synchro bus

DESCRIPTION : Subtractive and multiplicative offset frames are applied to the input pointed to by buffp, and thresholding and centroiding takes place, through a call to ospFG-CentroidWrapper. The Zernike coefficients are calculated and stored in the wfs context structure. Measurement variances are calculated and added to the fitting variances and the square root taken to give the error estimates for the Zernike coefficients, which are also stored in the wfs context structure. Under vxWorks, the Zernike coefficients and their standard deviations are written to the synchro bus. The input frame buffer is unmodified.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Need to check the return values of the functions called. Corner averaging for thresholding is not available. Error estimation is simplified, compared with osp-Measure.

3.13 ospFitVars - To estimate the fitting variances of the calculated Zernike coefficients

INVOCATION :

ospFitVars(v, w, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	v	float **	matrix from SVD of reconstructor
>	w	float *	vector of diagonal elements of matrix from SVD of reconstructor
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK

PURPOSE : To estimate the fitting variances of the calculated Zernike coefficients

DESCRIPTION : Variance derived from the fitting error is derived from matrices [v] and [w]. The equation used may be found in 'Numerical recipes in C' page 677, equation 15.4.19. Equations and page numbers are wrt the second edition- in all editions the equation may be found under the 'Solution by use of Singular Value Decomposition' subsection of the 'General Linear Least Squares' section of the 'Modeling (sic) of Data' chapter. The values calculated are stored in the context structure. As the fitting variances are only calculated at the same time as the control matrix, ospInit will generally just read the fvars values from file, rather than apply this function. The values as calculated here may be written to such a file using ospWriteVectorToFile, as is done in ospCalibrate.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Could benefit from checking for divide by zero from w[]. However the calculation of the control matrix by ospCalibrate ensures that w[] is real and positive.

3.14 ospFrameScramble - Scramble an entire frame of data

INVOCATION :

ospFrameScramble (xPixels, yPixels, outputs, inBuffer, outBuffer)

PARAMETERS : (">" input, "!" modified, "<" output)

>	xPixels	const int	Number of columns
>	yPixels	const int	Number of rows
>	outputs	const int	Number of detector outputs (2 or 4)

>	inFrame	float *	Pointer to input frame buffer
<	outBuffer	unsigned short int *	Pointer to output frame buffer

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : Scramble an entire frame of data

DESCRIPTION : This function takes a simulated frame of data and scrambles the pixels into the order they are read from the detector.

ACKNOWLEDGEMENTS : This function is almost identical to detFrameScramble, except that uint16 has been explicitly declared as unsigned short int, and the interim error handling is simpler. This function is based around the LeachDeScramble (lds) program provided by Les Saddlemyer and Tim Hardy, Hertzberg Institute of Astrophysics, Canada.

EXTERNAL VARIABLES : None. (The function needs to be reentrant)

PRIOR REQUIREMENTS : inBuffer must point to a buffer containing xPixels*yPixels floating point values. outBuffer must point to a buffer large enough to contain at least xPixels*yPixels unsigned short integer values.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.15 ospFrameTypeConvert - To convert a frame of floats to a frame of unsigned short ints, or vice-versa

INVOCATION :

ospFrameTypeConvert (usbuffp, fbuffp, mode, buffsize)

PARAMETERS : (">" input, "!" modified, "<" output)

> or <	usbuffp	char *	pointer to buffer of unsigned short ints
< or >	fbuffp	char *	pointer to buffer of floats
>	mode	char *	direction of type conversion
>	buffsize	int	size of frame

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To convert a frame of floats to a frame of unsigned short ints, or vice-versa

DESCRIPTION : Takes a buffer of unsigned short ints, and copies and casts it into a buffer of floats if mode is ">", or vice-versa if mode is "<".

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : That arrays of the required size (i.e. greater than or equal to buffsize) have been allocated .

INCLUDE FILES : osp.h

DEFICIENCIES : none known

3.16 ospGetBasisFunction - To populate the matrix which defines an arbitrary basis set in terms of their Zernike composition

INVOCATION :

ospGetBasisFunction (f,i,znum,mag)

PARAMETERS : (">" input, "!" modified, "<" output)

!	f	float **	output matrix showing basis functions in terms of Zernike coefficients
>	i	int	index of component, equal to index of actual Zernike function
>	znum	int	index of component, equal to index of arbitrary function
>	mag	float	size of component

FUNCTION VALUE :

void

PURPOSE : To populate the matrix which defines an arbitrary basis set in terms of Zernike polynomials and their coefficients

DESCRIPTION : Puts an element into a matrix. This process must be tied in to ospCalibrate and also linked with the vector (or file containing the vector) of the numbered basis function for which we are storing the Zernike coefficients.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : f[][] must be intialised as identically zero, and of the appropriate dimensions, as is done in ospCalibrate.

INCLUDE FILES : osp.h

DEFICIENCIES : Incomplete, never tested...

3.17 ospGlobalCentroid - To determine the centroid of an entire frame, in pixels

INVOCATION :

ospGlobalCentroid(buffp, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to input frame buffer
!	wfsSpecific	struct OSP_CONTEXT	pointer to wfs context structure

FUNCTION VALUE : void

PURPOSE : To determine the centroid of an entire frame, in pixels

DESCRIPTION : Uses moments to calculate the centroid of an entire frame. As ever, the coordinate system for the entire frame is x-y, in units of pixels, with the centre of the bottom left hand pixel in the frame (1,1).

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.18 ospGlobalThreshold - To apply a common threshold to an entire frame

INVOCATION :

ospGlobalThreshold(buffp, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to input frame buffer
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

void

PURPOSE : To apply a common threshold to an entire frame

DESCRIPTION : The threshold is supplied as wfsSpecific->thresh. Pixels above this threshold have it subtracted from their value. Pixels below this threshold have their values set to 0.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.19 ospInit - To create and initialise a wfs context structure

INVOCATION :

wfsSpecific = ospInit(wfsName,ospGeom)

PARAMETERS : (">" input, "!" modified, "<" output)

>	wfsName	char *	name of the .ini file to be read
>	ospGeom	struct OSP_GEOMETRY *	pointer to ccd readout geometry structure

FUNCTION VALUE :

struct OSP_CONTEXT *	pointer to wfs context structure

PURPOSE : To create and initialise a wfs context structure

DESCRIPTION : A wfs context structure is created, with its pointer the return value of the function. The .ini file named as wfsName is read, and the values read in either directly initialise elements of the context structure, or are names of files which in turn are read to initialise elements. The context structure is/will be documented in the osp.h header file, but includes the ccd readout geometry, the null positions of the SH spots, the control matrix, frames of subtractive and multiplicative offsets which are to be applied to subsequent frames of SH spots, measured spot displacements, fitting and measurement variances, and ultimately, calculated Zernike coefficients. The values for the ccd readout geometry read in from the .ini file are also stored in the structure as default values which will be reverted to should ospChangeGeometry be called with the framesizeflag element of its argument equal to 1 (full frame). This behaviour should perhaps be modified as it limits us to a single full frame readout geometry, unless we modify the .ini file and call ospInit again. The ospGeom structure allows the ccd readout geometry to be modified for reduced frames on calling ospInit, but as this is carried out by ospChangeGeometry the same restriction on the full

frame readout using the default values applies. Full frame versions of the offset frames are stored in the wfs context structure, from which any versions for reduced frame readout may be calculated, and also stored in the wfs context structure. If no ospGeom structure modifying the readout geometry is necessary on initialisation, supply NULL instead. Much memory allocation takes place during this function- remember to use the companion function ospTidyUp(wfsSpecific) to deallocate the memory and destroy the pointer to the structure. The pointer to the ospGeom structure is freed separately.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.20 ospMeasVars - To estimate the measurement variances of the calculated Zernike coefficients

INVOCATION :

ospMeasVars(wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK.

PURPOSE : To estimate the measurement variances of the calculated Zernike coefficients

DESCRIPTION : Variance derived from measurement errors in the vector ds of SH spot displacements is obtained from the sum of squares of the (elements of ds multiplied by the appropriate row of [c]) for each Zernike coefficient. The values calculated are stored in the context structure.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.21 ospMeasure - To accept an input frame of SH spots and calculate the Zernike coefficients

INVOCATION :

ospMeasure(buffp,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	input frame buffer
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To accept an input frame of SH spots and calculate the Zernike coefficients

DESCRIPTION : The subtractive and multiplicative offset frames are applied to the input pointed to by buffp. Thresholding and centroiding then takes place through a call to ospCentroidWrapper, and the Zernike coefficients are calculated and stored in the wfs context structure. Measurement variances are calculated and added to the fitting variances and the square root taken to give the error estimates for the Zernike coefficients, which are also stored in the wfs context structure. NB The input buffer, on exit has had the subtractive and multiplicative offsets applied, and has been thresholded.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Need to check the return values of the functions called. Slower than ospFGMeasure.

3.22 ospMultiplyFrameByConstant - To multiply each pixel in a frame by the same specified amount

INVOCATION :

ospMultiplyFrameByConstant(buffp1,constname,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp1	float *	pointer to buffer to be multiplied
>	constname	float	multiplying constant
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To multiply each pixel in a frame by the same specified amount

DESCRIPTION : Multiplies each pixel in an existing frame buffer pointed to by buffp1 by the floating point value constname. The frame is assumed to be wfsSpecific->xframesize by wfsSpecific->yframesize. No checking of the frame size is done, due to the function's intended use of making corrections to fast guide data frames, which requires low latency.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the pointer points to a buffer which is already allocated and large enough.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.23 ospMultiplyFrameByFrame - To multiply a frame by an existing frame of the same size, pixel by pixel

INVOCATION :

ospMultiplyFrameByFrame(buffp1,buffp2,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp1	float *	pointer to existing frame which is to be modified by multiplication
>	buffp2	float *	pointer to frame by which existing frame is to be multiplied
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To multiply a frame by an existing frame of the same size, pixel by pixel

DESCRIPTION : Multiplies an existing frame buffer pointed to by buffp1 by another, pointed to by buffp2. The frames are both assumed to be wfsSpecific->xframesize by wfsSpecific->yframesize. The multiplication is done pixel by pixel. No checking of the frame size is done, due to the function's intended use of making corrections to fast guide data frames, which requires low latency.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the pointers point to buffers which are already allocated and large enough.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.24 ospNewCalibrate - To populate a reconstructor matrix and invert to obtain control matrix

INVOCATION :

ospNewCalibrate(calibpath, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	calibpath	char *	path to directory of calibration data
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To populate a reconstructor matrix and invert to obtain control matrix

DESCRIPTION : This differs from ospCalibrate in that the files and information required are prompted for. Calibration frames of displaced SH spots corresponding to linearly independent combinations of known Zernike aberrations are used to form a reconstructor matrix with respect to the basis functions defined by the linear combinations. This reconstructor matrix is transformed into one with respect to the individual Zernike polynomials of unit magnitude, and further inverted to provide the necessary control matrix which is applied to the SH spot displacements in an arbitrary input frame to yield the Zernike coefficients. The control matrix is written to a file specified by the character string wfsSpecific->controlfile, and the calculated fitting variances written to a file specified by the character string wfsSpecific->fvarsfile. If either of these files exist, they are copied to a file with a .bak extension added, befeore the new version is created.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : Not general enough-need to prompt for data input, basis functions etc.

3.25 ospNewReadCentres - To calculate coordinate data for the subaperture null positions from data stored in file

INVOCATION :

ospNewReadCentres(nullfile,centres,side,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	nullfile	char *	name of file containing null positions of the SH spots in the subapertures
<	centres	float *	pointer to array containing the coordinates of the subapertures bottom left hand corners, and the relative offsets of the null positions (in pixels)
>	side	int	side length of subap, in pixels
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK

PURPOSE : To calculate coordinate data for the subaperture null positions from data stored in file

DESCRIPTION : This function is essentially a wrapper for the two functions ospReadNulls and ospCalculateSubaps, and was written as an intermediate stage in the evolution of these two functions from the previous ospReadCentres, which read from a file containing different data in another format. Essentially, the null positions are read from a file, coordinates relative to the whole array, in pixels, and negative coordinates supplied for the subapertures to be ignored... for more information see ospReadNulls. The postions of the bottom left hand corners of the subapertures are calculated, given the values of the detector readout geometry stored in the context structure, and the offsets of the given null positions from these coordinates calculated for each subaperture. This information is stored in the wfsSpecific->centres array, and here is copied to the centres argument.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Possibly irrelevant

3.26 ospNullCorrection - To provide a simple interface by which the null positions file can be corrected using a FITS image of nulled spots

INVOCATION :

ospNullCorrection(wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To provide a simple interface by which the null positions file can be corrected using a FITS image of nulled spots

DESCRIPTION : The function issues a series of prompts for data which allows the null positions file to be corrected given a FITS image of nulled spots. Invocation of the function is particularly simple, and the first prompt allows the functionality of this function to be bypassed in the event of of the current nullfile being known to be accurate. If null correction is to take place, the readout geometry of the OSP_CONTEXT structure is changed to full frame (all calibration must take place wrt full frame data to cope with all geometries of the reduced frames) and the output from ospCalculateSubaps which is called as a consequence allow the next prompt (asking if the current nullfile is approximately accurate) to be answered: yes, if the subapertures with no null positions are those we have chosen to ignore, and if no 'Badly centred null position' warnings are present; no, otherwise. If yes, the filename of a full frame float FITS image showing the nulled SH spots is prompted for. As this file may be older and have had the dark and flat field applied already, an option is given to avoid the application of the current subtractive and multiplicative offset frames. The calculation of the corrections is performed by a call to ospSaveNullPositions, which backs up the current nullfile with a .bak suffix, writes the new values into a file with the same name as the original nullfile to save editing the .ini file, and applies the new null positions to the OSP_CONTEXT structure, which is reset to its original readout geometry proir to exit. This function may typically be called prior to ospNewCalibrate. See also the header of ospSaveNullPositions for more details of the nullfile format.

EXTERNAL VARIABLES : None

EXTERNAL REFERENCES : fitsio library

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.27 ospPrintError - To report cfitsio errors, as related to the status codes in fitsio.h

INVOCATION :

ospPrintError(status)

PARAMETERS : (">" input, "!" modified, "<" output)

>	status	int	status value relevant to cfitsio functions

FUNCTION VALUE :

int	A status value equal to ERROR.

PURPOSE : To report cfitsio errors, as related to the status codes in fitsio.h

DESCRIPTION : Receives a status code from the cfitsio function which calls it, prints the appropriate error report, and returns an ERROR value.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : fitsio.h

DEFICIENCIES : None known

3.28 ospPrintHeaders - To print out all the header keywords in all extensions of a FITS file

INVOCATION :

ospPrintHeaders(infile)

PARAMETERS : (">" input, "!" modified, "<" output)

>	infile	char *	name of input file

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To print out all the header keywords in all extensions of a FITS file

DESCRIPTION : Function taken from cfitsio library and modified slightly. Prints out all the keywords records from all HDUs in a FITS file until an EOF is is encountered.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : fitsio.h

DEFICIENCIES : None known

3.29 ospReadFloatImage - To read a float image into a buffer from a FITS file

INVOCATION :

ospReadFloatImage(buffp,infile,buffsize)

PARAMETERS : (">" input, "!" modified, "<" output)

<	buffp	float *	pointer to buffer for image read in
>	infile	char *	name of input FITS file
>	buffsize	int	size of frame buffer

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To read a float image into a buffer from a FITS file

DESCRIPTION : The FITS file is opened and the NAXIS keywords read to get the image size. If the size is greater than buffsize, only enough of the image to fill the buffer is read in. If the image is smaller than or equal to the size of the buffer, the whole image is read in. No padding to fill any unassigned elements of the buffer takes place, as the image dimensions for any subsequent processing should be strictly controlled to match the xframesize and yframesize dimensions specified in the context structure. The FITS file is then closed. This function was written primarily for input of test data, but is also used for input of the offsets frames- it may benefit from also taking the context strtucture as an argument, to enable checking of the image dimensions.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : That the buffer pointed to by buffp has been allocated large enough to accomodate buffsize floats

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : None known

3.30 ospReadHeaderInt - To read the integer values of an array of keywords from a FITS header

INVOCATION :

ospReadHeaderInt (filename,numelem,keynames,values)

PARAMETERS : (">" input, "!" modified, "<" output)

>	filename	char *	name of FITS file whose header is being read
>	numelem	int	maximaum number of keywords to be read
>	keynames	char **	name of array of keywords
<	values	int *	array of integer values of keywords

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To read the integer values of an array of keywords from a FITS header

DESCRIPTION : Uses functions from the cfitsio library to open a FITS file for reading, and read the values of specified keywords with integer values. The keywords elements of an array, and their values are read into the corresponding elements of an array of integers. The number of elements of the keyword array may exceed the number of keywords actually present, but obviously not the number of array elements allocated. The keywords should be consecutive elements of their array, as reading of keywords will end when a NULL value is encountered as a keyword. The file is closed when reading is finished.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : Restricted to keywords with integer values.

3.31 ospReadNulls - To read the coordinates of the null positions from a file

INVOCATION :

ospReadNulls(nullfile, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	nullfile	char *	name of file containing the null positions of the SH spots in the subapertures
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To read the coordinates of the null positions from a file

DESCRIPTION : The file named nullfile is opened, and the null positions for the subapertures stored in it read into the wfsSpecific->nulls[] structure elements. The null positions are in units of pixels, and are relative to the bottom left hand corner of the CCD array. The centre of the bottom left hand pixel in the array has coordinate (1,1). The data is stored in the file with two records per line, separated by a space, corresponding to the x and y coordinate, and each line corresponding to a subaperture. The data are read into wfsSpecific->nulls in the order wfsSpecific->nulls[0] = first x coordinate, wfsSpecific->nulls[1] = first y coordinate, wfsSpecific->nulls[2] = second x coordinate etc. The lines of data are ordered to match the order in which the subaperture coordinates are subsequently calculated: the null position coordinates for the bottom left subaperture are first, then the subaperture along the bottom row from left to right, moving up a row at a time. By way of example, for a 3 x 3 array of subapertures, they would be ordered as follows:

7	8	9
4	5	6
1	2	3

Subapertures for which a centroid is not to be calculated, and/or for which a null position may not be calculated (e.g. they may lie outwith the illuminated field) still require a line of data corresponding to coordinates to be input- these coordinates however should be negative to indicate that the subaperture is to be subsequently ignored. The pairs of subapertures are read until an EOF is encountered, or until the limit set by OSP_SUBAPSMAX in osp.h is reached. The number of pairs of coordinates is checked to be consistent with the number of subapertures specified by the ccd readout geometry stored in wfsSpecific, and an ERROR returned instead of OK, after closing the file.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.32 ospReadUShortImage - To read an unsigned short int image into a buffer from a FITS file

INVOCATION :

ospReadUshortImage(buffp,infile,buffsize)

PARAMETERS : (">" input, "!" modified, "<" output)

<	buffp	unsigned short int *	pointer to buffer for image read in
>	infile	char *	name of input FITS file
>	buffsize	int	size of frame buffer

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To read an unsigned short int image into a buffer from a FITS file

DESCRIPTION : The FITS file is opened and the NAXIS keywords read to get the image size. If the size is greater than buffsize, only enough of the image to fill the buffer is read in. If the image is smaller than or equal to the size of the buffer, the whole image is read in. No padding to fill any unassigned elements of the buffer takes place, as the image dimensions for any subsequent processing should be strictly controlled to match the xframesize and yframesize dimensions specified in the context structure. The FITS file is then closed. This function was written primarily for input of test data, but may benefit from also taking the context strtucture as an argument, to enable checking of the image dimensions.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : That the buffer pointed to by buffp has been allocated large enough to accomodate buffsize unsigned short ints

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : None known

3.33 ospReduceFits - To generate a reduced frame FITS file from a full frame FITS file

INVOCATION :

ospReduceFits(infile, outfile, wfsSpecific, buffsize)

PARAMETERS : (">" input, "!" modified, "<" output)

>	infile	char *	name of input full frame FITS file
<	outfile	char *	name of output reduced frame FITS file
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure
>	buffsize	int	buffer size for reduced frame

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To generate a reduced frame FITS file from a full frame FITS file

DESCRIPTION : Buffers for the full and reduced frame are dynamically allocated. The full frame FITS file is read into the full frame buffer, and ospMeasure is called to write the reduced version to the reduced frame buffer, which is then written to a FITS file named outfile. The structure element framesizeflag is set to zero (corresponding to a reduced frame) and the geometric parameters of the CCD readout used to reduce the frame are written to the FITS header by calling ospAddContextToHeader. framesizeflag is restored to its entry value and the buffers are freed.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None.

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h, fitsio.h

DEFICIENCIES : FITS header from original file not copied to new reduced file, thus losing the user specified keyword information from the original. Reads and writes float images specifically- a future modification should read the BITPIX keyword from the input FITS file and react accordingly. No provision for binning of pixels into superpixels.

3.34 ospReduceFrame - To reduce the frame size to the minimum containing all subaperture pixels

INVOCATION :

ospReduceFrame (buffp, newbuffp, wfsSpecific, buffsize)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	float *	pointer to full frame as input
<	newbuffp	float *	pointer to reduced frame
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure
>	buffsize	int	buffer size for reduced frame

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To reduce the frame size to the minimum containing all subaperture pixels To reduce the size of the data frame by removing the pixels which lie outwith the square which contains the subapertures, and also the pixels which lie on the lines between the subapertures. Thus a full frame of 80x80 pixels with a 6x6 array of subapertures each of 6x6 pixels will be reduced to a an 36x36 subframe. This function has been written to enable reduced frame versions of the offset frames to be generated from their full frame versions in response to any change of detector readout geometry, and also to enable testing of realistic readout geometries from full frame data.

DESCRIPTION : The function uses the geometric parameters specified by the sectors, xstart, ystart, xbin, ybin, xraster, yraster, xspace, yspace, xsubap, ysubap, xarraysize and yarraysize elements of the OSP_CONTEXT structure to calculate the (full frame) coordinates of the pixels which lie within the grid of subapertures, and then calculates the new coordinates for the reduced frame, such that (xstart+1, ystart+1) becomes (1,1), and the subapertures are immediately adjacent to one another. The symmetry requirements imposed by either the two or four sector readouts are observed in the calculation of the subaperture coordinates, in the same manner as ospCalculateSubaps. xtail and ytail are calculated but no testing is done to see if they are realisable physically- this testing is done in ospInit or ospChangeGeometry.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : That the OSP_CONTEXT structure has been updated to correspond with the current geometric parameters for the CCD readout. It is not necessary for structure element framesizeflag to be set to 0, corresponding to a reduced frame.

INCLUDE FILES : osp.h

DEFICIENCIES : No provision for binning of pixels into superpixels.

3.35 ospSaveNullPositions - To create or update null positions data from a calibration file

INVOCATION :

ospSaveNullPositions(nullname,outfile, wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	nullname	char *	name of full frame FITS image from which null positions are to be read
>	oflag	int	determines whether offsets are to be applied OFFSET_CORRECTION = 1, NO_OFFSET_CORRECTION=0
>	outfile	char *	name of modified nullfile
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To create or update null positions data from a calibration file

DESCRIPTION : The calibration frame is read from a FITS file, with the NAXIS1 and NAXIS2 keyword values read to ensure that we are dealing with full frame data. Centroiding is carried out to determine the centroid positions, which here are corrections to the current null positions. These corrections are added to the current null values and written out to a file. We use a 'null' file of x and y coordinates relative to the bottom left hand corner of the whole array, which is read in and the bottom left corner positions of the subapertures generated from knowledge of the detector geomemtry, and the corresponding xoff, yoff values calculated. As the subaperture positions are calculated in order from left to right and from top to bottom with no prior knowledge of which subapertures are unused, unused subapertures are indicated with a null position of -1,-1.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Should save previous nullfile before overwriting, just in case.

3.36 ospShow - To display the current values of the wfs context

INVOCATION :

ospShow(wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To display the current values of the wfs context

DESCRIPTION : Generates a neatly formatted summary of most of the context structure's current values on standard output. Parameters reported are:

osp structure:	wfsSpecific
SECTORS :	wfsSpecific->sectors
XSTART :	wfsSpecific->ospxstart
YSTART :	wfsSpecific->ospystart
XBIN :	wfsSpecific->ospxbin
YBIN :	wfsSpecific->ospybin
XRASTER :	wfsSpecific->ospxraster
YRASTER :	wfsSpecific->ospyraster
XSPACE :	wfsSpecific->ospxspace
YSPACE :	wfsSpecific->ospyspace
XSUBAP :	wfsSpecific->ospxsubap
YSUBAP :	wfsSpecific->ospysubap
OSP_FSZ (0:reduced,1:full frame) :	wfsSpecific->framesizeflag
Number of Zernikes (np) :	wfsSpecific->np
2*TOTAL number of subaps (mp) used :	wfsSpecific->mp
xarraysize (in pixels) :	wfsSpecific->xarraysize
yarraysize (in pixels) :	wfsSpecific->yarraysize
xframesize (in pixels) :	wfsSpecific->xframesize
yframesize (in pixels) :	wfsSpecific->yframesize
side length of subap (in pixels) :	wfsSpecific->side
size of buffer for frame :	wfsSpecific->buffsize
nsigma (used in ospThreshold) :	wfsSpecific->nsigma
Read noise per subaperture,squared :	wfsSpecific->readsq
Weighting applied? (0:no, 1:yes) :	wfsSpecific->weight
Threshold parameter :	wfsSpecific->thresh
Coaddcounter (frames coadded -1) :	wfsSpecific->coaddcounter
WfsSource (wfs id) :	wfsSpecific->wfsSource
WfsMode (AO = 0, FG = 1) :	wfsSpecific->wfsMode

Centres data : number of centres*4 :	wfsSpecific->centres[0] :
x0, xoff, y0, yoff :	wfsSpecific->centres[]...
Control matrix used (first 10 cols):	wfsSpecific->c[][]...
Displacements s-x, ds-x, s-y, ds-y :	wfsSpecific->s[]...
Fitting variances :	wfsSpecific->fvars[]...
Measurement variances :	wfsSpecific->mvars[]...
Zernikes calculated :	wfsSpecific->z[]...

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Should test if the structure is NULL

3.37 ospSimulateCentroids - To generate centroid data for a specified combination of Zernike functions

INVOCATION :

ospSimulateCentroids(matr,vect)

PARAMETERS : (">" input, "!" modified, "<" output)

>	matr	float **	reconstructor matrix relative to basis functions of Zernike polynomials in ascending order starting with tip, tilt, focus...
<	v	float *	vector of displacements (in pixels) corresponding to specified combination of Zernike polynomials

FUNCTION VALUE :

void

PURPOSE : To generate centroid data for a specified combination of Zernike functions

DESCRIPTION : Using displacements for normalised model Zernike functions, the specified combination is built up by addition of the scaled displacements from each individual Zernike function, stored in the form of a reconstructor matrix, The scale factor is prompted for on the standard output and suppplied on the standard input.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Should implement error checking to ascertain that none of the columns of the matrix matr are zero vectors

3.38 ospSubtractFrameFromFrame - To subtract a frame from an existing frame of the same size, pixel by pixel

INVOCATION :

ospSubtractFrameFromFrame(buffp1,buffp2,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp1	float *	pointer to buffer from which frame is to be subtracted
>	buffp2	float *	pointer to buffer containing frame to be subtracted
>	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To subtract a frame from an existing frame of the same size, pixel by pixel

DESCRIPTION : Subtracts a frame pointed to by buffp2 from an existing frame buffer pointed to by buffp1. The frames are both assumed to be wfsSpecific->xframesize by wfsSpecific->yframesize. The subtraction is done pixel by pixel. No checking of the frame size is done, due to the function's intended use of making corrections to fast guide data frames, which requires low latency.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the pointers point to buffers which are already allocated and large enough.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.39 ospTestData - To generate a known test frame of data

INVOCATION :

ospTestData (buffp,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to output buffer containing test frame
>	wfsSpecific	struct OSP_CONTEXT *	wfs context structure

FUNCTION VALUE :

void

PURPOSE : To generate a known test frame of data

DESCRIPTION : Generates a linear ramp of increasing intensity in the x direction.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS : That the buffer pointed to by buffp has been allocated large enough to accomodate wfsSpecific->xarraysize x wfsSpecific->yarraysize floats.

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.40 ospThreshold - To estimate and/or impose a threshold for each subaperture to reduce the effects of noise

INVOCATION :

ospThreshold(buffp, meanval,wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	buffp	float *	pointer to frame buffer
<	meanval	float *	pointer to array of intensity values subtracted from above threshold pixels in each subap
!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To estimate and/or impose a threshold for each subaperture to reduce the effects of noise

DESCRIPTION : Using the readout geometry data from wfsSpecific (particularly that in wfsSpecific->centres[] which records the coordinates of the bottom left pixel of each subaperture) a threshold is applied to each subaperture which is in use, modifying the frame buffer pointed to by buffp correspondingly. The manner in which the threshold is determined for each subaperture depends on the value of wfsSpecific->thresh: if it is -1, threshold value for each subaperture is determined by averaging the intensities of the lowest (intensity) three of the four corner pixels, calculating their standard deviation and setting the threshold equal to mean + n * standard deviation, where n is the value specified by wfsSpecific->nsigma; if it is -3, the threshold value for each subaperture is determined by averaging three pixels at each corner, calculating their standard deviation and setting the threshold equal to mean + standard deviation, where etc.; if it is >=0, that value is applied as the threshold; for any other value, an error is reported and returned. For the first two cases, the threshold is applied by setting pixel values below it to zero, and subtracting the mean calculated from the corner pixels from the pixel values above zero. The actual values subtracted are returned in the array pointed to by meanval. For the third case, the pixel values below threshold are set to zero, the ones above threshold have threshold subtracted, and the threshold value is returned in the elements of meanval.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : Would be better integrated into the centroid function.

3.41 ospTidyUp - To free all the memory allocated from the corresponding ospInit()

INVOCATION :

ospTidyUp(wfsSpecific)

PARAMETERS : (">" input, "!" modified, "<" output)

!	wfsSpecific	struct OSP_CONTEXT *	pointer to wfs context structure

FUNCTION VALUE :

int	A status value equal to OK or ERROR

PURPOSE : To free all the memory allocated from the corresponding ospInit()

DESCRIPTION : The memory allocated when wfsSpecific was created using ospInit is freed, including the memory allocated to contain the structure itself.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : None

PRIOR REQUIREMENTS :

INCLUDE FILES : osp.h

DEFICIENCIES : None known

3.42 ospWriteFloatImage - To write a float image to a FITS file

INVOCATION :

ospWriteFloatImage(buffp, outfile, xarraysize, yarraysize)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	float *	pointer to float image to be written to FITS file
>	outfile	char *	name of output FITS file
>	xarraysize	int	number of pixels in x of image
>	yarraysize	int	number of pixels in y of image

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To write a float image to a FITS file

DESCRIPTION : A 2-d float image is written as a FITS primary array, with a minimal header. If the detector geometry is required to be added to the header use ospAddContextTo-Header.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : None

INCLUDE FILES : fitsio.h

DEFICIENCIES : None known

3.43 ospWriteUShortImage - To write an unsigned short int image to a FITS file

INVOCATION :

ospWriteUShortImage(buffp, outfile, xarraysize, yarraysize)

PARAMETERS : (">" input, "!" modified, "<" output)

>	buffp	unsigned short int *	pointer to unsigned short int image to be written to FITS file
>	outfile	char *	name of output FITS file
>	xarray-size	int	number of pixels in x of image
>	yarray-size	int	number of pixels in y of image

FUNCTION VALUE :

int A status value equal to OK or ERROR

PURPOSE : To write an unsigned short int image to a FITS file

DESCRIPTION : A 2-d unsigned short int image is written as a FITS primary array, with a minimal header. If the detector geometry is required to be added to the header use ospAddContextToHeader.

Author : Steven Heddle, UKATC, Edinburgh 18/1/1999

EXTERNAL VARIABLES : none

PRIOR REQUIREMENTS : None

INCLUDE FILES : fitsio.h

DEFICIENCIES : None known

4.0 Structures used in ospLib

4.1 The OSP_CONTEXT structure

```
struct OSP_CONTEXT{
  char nullfile[OSP_MAXSTR];   /* name of ASCII file of null positions, shows unused subaps */
  char subfile[OSP_MAXSTR];    /* name of FITS file of subtractive frame offsets */
  char multfile[OSP_MAXSTR];   /* name of FITS file of multiplicative frame offsets*/
  char controlfile[OSP_MAXSTR];/* name of ASCII file containing control matrix */
```

```
char fvarsfile[OSP_MAXSTR];  /* name of ASCII file containing fitting variances */
int np;                      /* number of Zernike coefficients to be calculated */
int mp;                      /* number of subapertures to be used */
int xarraysize;              /* x dimension in pixels read out from CCD */
int yarraysize;              /* y dimension in pixels read out from CCD */
int buffsize;          /* size of buffer to store full frame=xarraysize * yarraysize */
int side;             /* number of pixels along side of subaperture (assumed square) */
int ospxstart;         /* current number of pixels skipped before first subap in x*/
int ospystart;        /* current number of pixels skipped before first subap in y*/
int ospxbin;                 /* current binning factor for pixels in x */
int ospybin;                 /* current binning factor for pixels in y */
int ospxraster;              /* current x size of subaperture in pixels */
int ospyraster;              /* current y size of subaperture in pixels */
int ospxspace;               /* current separation of subapertures in x, in pixels */
int ospyspace;               /* current separation of subapertures in y, in pixels */
int ospxsubap;               /* current number of subapertures per sector in x */
int ospysubap;               /* current number of subapertures per sector in y */
int defxstart;          /* default number of pixels skipped before first subap in x*/
int defystart;         /* default number of pixels skipped before first subap in y*/
int defxbin;                 /* default binning factor for pixels in x */
int defybin;                 /* default binning factor for pixels in y */
int defxraster;              /* default x size of subaperture in pixels */
int defyraster;              /* default y size of subaperture in pixels */
int defxspace;               /* default separation of subapertures in x, in pixels */
int defyspace;               /* default separation of subapertures in y, in pixels */
int defxsubap;               /* default number of subapertures per sector in x */
int defysubap;               /* default number of subapertures per sector in y */
int sectors;                 /* number of sectors readout from CCD array */
int framesizeflag;           /* flag = 0 for reduced frame, 1 for full frame */
int xframesize;              /* x size of frame currently in use */
int yframesize;              /* y size of frame currently in use */
int framebuffsize;   /* size of buffer to store current frame=xframesize*yframesize*/
float nsigma;     /* no. of std. devs above corner pixel average that threshold is applied*/
float readsq;                /* read noise in (e-)(squared) */
```

```
   int weight;         /* flag = 0 for no weighting, = 1 for weighting**UNUSED AT
PRESENT*/
   float thresh;     /* threshold value applied to corrected data frame- NB -1 and -3 values */
   float nulls[2*OSP_SUBAPSMAX];              /* null positions read in from nullfile */
   float centres[4*OSP_SUBAPSMAX +1];       /* positions of bottom left corner, and dis-
placement of null position from there, for each subaperture used, in coordinates of reduced
frame */
   double time;                        /* variable for timestamp */
   float * ffsubbuff;                  /* pointer to full frame buffer for subtractive offsets */
   float * redsubbuff;          /* pointer to reduced frame buffer for subtractive offsets */
   float * ffmultbuff;         /* pointer to full frame buffer for multiplicative offsets */
   float * redmultbuff;       /* pointer to reduced frame buffer for multiplicative offsets */
   float * err;                        /* pointer to vector of standard errors */
   float **c;                          /* pointer to pointer to control matrix */
   float *s;                 /* pointer to vector of centroid displacements from null positions */
   float *dssq;            /* pointer to vector of vraiances of centroid displacements */
   float *fvars;                       /* pointer to vector of fitting variances */
   float *mvars;                       /* pointer to vector of measurement variances */
   float *z;                           /* pointer to vector of Zernike coefficients */
   float * sumbuff;                    /* pointer to buffer for coadded frames */
   int coaddcounter;                   /* counter of coadded frames */
   clock_t coaddstart;                 /* start time for coadding- used to check timeout */
   int wfsSource;                      /* label of wfs: see enum below */
   int wfsMode;                        /* label of wfs mode: see enum below */
   float ospdiag[DIAG_ARRAY_SIZE];  /* array of diagnostic values for development */
   float tipscale;                     /* scaling factor for tip, used in development */
   float tiltscale;                    /* scaling factor for tilt, used in development */
   float focusscale;                   /* scaling factor for focus, used in development */
   float tipCor;                       /* ?, used in development */
   float tiltCor;                      /* ?, used in development */
};
```

4.2 The OSP_GEOM structure

```
struct OSP_GEOMETRY{
   int sectors;                        /* number of sectors readout from CCD array */
```

```
    int xstart;                 /* number of pixels skipped before first subap in x*/
    int ystart;                 /* number of pixels skipped before first subap in y*/
    int xbin;                   /* current binning factor for pixels in x */
    int ybin;                   /* current binning factor for pixels in y */
    int xraster;                /* x size of subaperture in pixels */
    int yraster;                /* y size of subaperture in pixels */
    int xspace;                 /* separation of subapertures in x, in pixels */
    int yspace;                 /* separation of subapertures in y, in pixels */
    int xsubap;                 /* number of subapertures per sector in x */
    int ysubap;                 /* number of subapertures per sector in y */
    int xarraysize;             /* x dimension in pixels read out from CCD */
    int yarraysize;             /*  dimension in pixels read out from CCD */
    int framesizeflag;          /* flag = 0 for reduced frame, 1 for full frame */
};
```