CSR	MASSACHUSETTS INSTITUTE OF TECHNOLOGY CENTER FOR SPACE RESEARCH CAMBRIDGE, MASSACHUSETTS 02139	
REVISION LOG	**TITLE: Software Detailed Design** **FEP Science Timed Exposure Bias Calibration**	**DOC. NO.** **36-53226 Rev. B**

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
01	5/11/95	36-255	all	New document	
A	5/1/96	36-613	all	Incorporate earlier review comments and updated design	RFG 5/23/96
B	6/17/96	36-623	1235, 1241	Added support to ignore "n" initial frames	6/17/96

~~Controlled Copy~~

22.0 FEP Timed Exposure Bias Calibration (36-53226 B)

22.1 Purpose

The ***fepTimedBias*** module, executing in the FEP, calibrates the bias map for subsequent timed-exposure science processing. It is called from ***fepCtl*** with a single argument, `fp`, a pointer to the `fepParm` structure. Before invoking ***fepTimedBias***, the FEP must be commanded to load a timed-exposure `FEPparmBlock` into *`fp->tp`*.

22.2 Uses

The ***fepTimedBias*** function operates in one of the following modes, according to the value of *`fp->tp.type`*:

Use 1:: Calculate bias using a "whole-frame" algorithm.
Use 2:: Calculate bias thresholds using a "strip" algorithm.

FIGURE 183. fepTimedBias Structure in "Whole-Frame" Mode

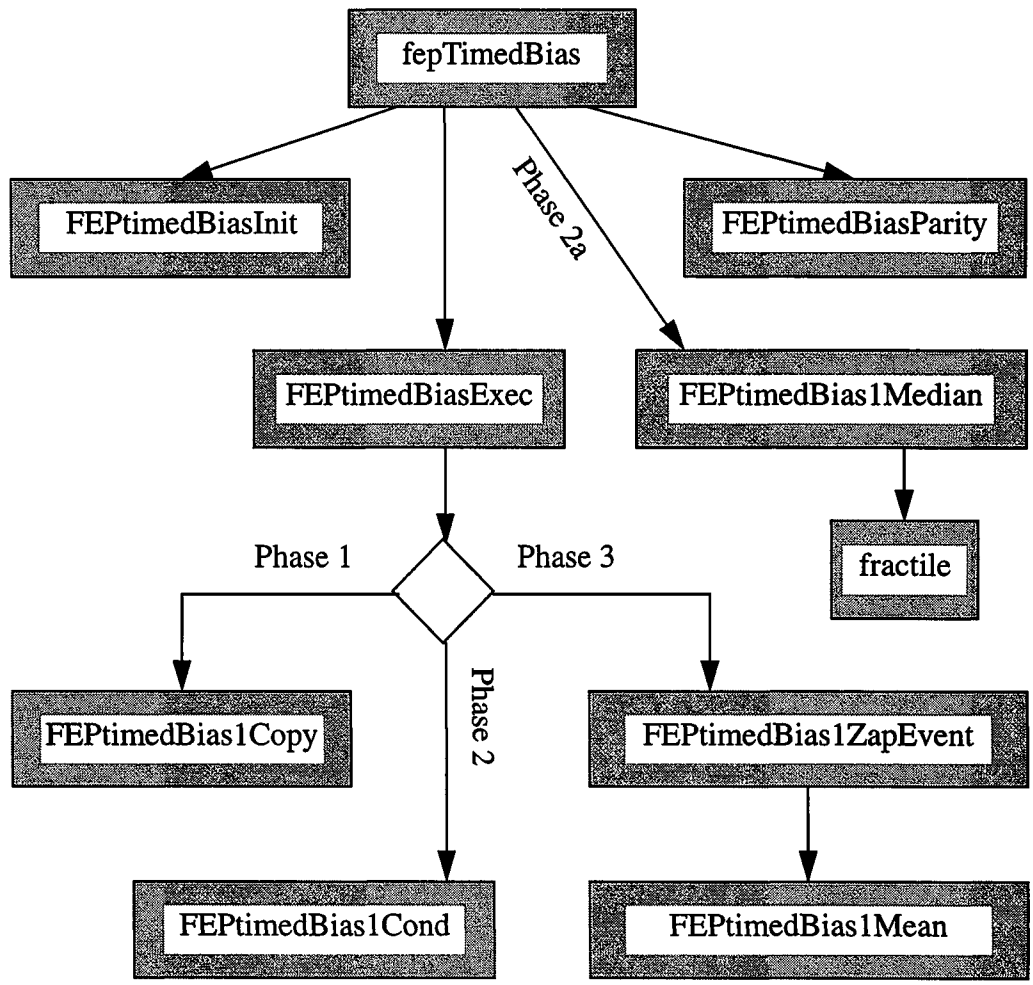
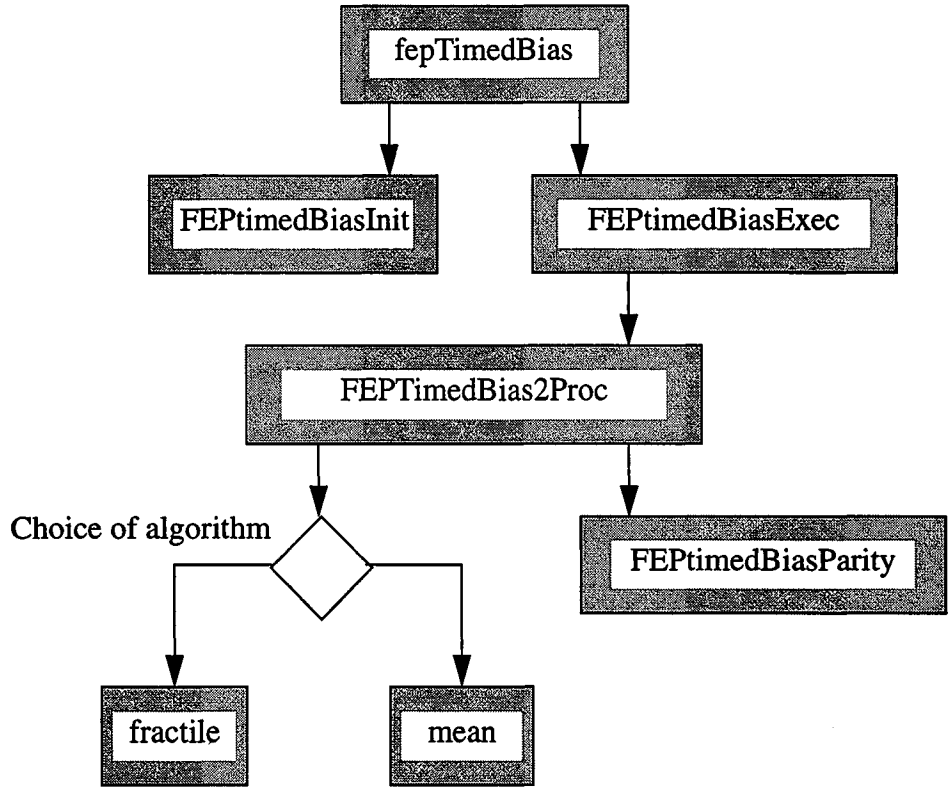


FIGURE 184. fepTimedBias Structure in "Strip" Mode



22.3 Organization

All interactions with the BEP and with FEP hardware are made through ***fepio*** library functions described in Section 39.0. The commands that are passed between FEP and BEP are defined in that document and in Section 4.10. ***fepTimedBias*** makes use of the following functions which call each other in the manner shown in Figure 183 ("whole-frame" mode) or in Figure 184 ("strip" mode), depending on the value of `fp->tp.btype` (see Table 46):

- ***FEPtimedBiasInit***—validates the `FEPparmBlock`, `fp->tp`, and performs all necessary mode-dependent initialization.
- ***FEPtimedBiasExec***—processes a single image frame. In "whole-frame" mode (Figure 183), it sums the overclocks and calls a subroutine, either ***FEPtimedBias1Copy***, ***FEPtimedBias1Cond***, ***FEPtimedBias1ZapEvent***, or ***FEPtimedBias1Mean***, depending on the value of the mode parameter, to process each line. In "strip mode" (Figure 184), this routine merely adjusts the hardware pointers for each strip until the image map is filled, when it sums the overclocks of the most recent exposure and calls ***FEPtimedBias2Proc*** to process the pixel values.
- ***FEPtimedBiasParity***—inspects one or more rows of bias map pixels and constructs a bias parity table in which each bit represents the parity (`EVEN=0`, `ODD=1`) of the corresponding bias value. In "whole-frame" mode, this routine is called once at the very end of the task. In "strip" mode, it is called after each strip of bias values has been created.

- ***FEPtimedBias1Copy***—copies a line of pixels from the image map to the bias map. This is called to process the first bias exposure in "whole-frame" mode.
- ***FEPtimedBias1Cond***—compares a line of pixels against a line of bias values, updating the bias with the corresponding pixel value when the latter is lower in value. This is called a total of `fp->tp.bparm[0]-1` times in "whole-frame" mode.
- ***FEPtimedBias1ZapEvent***—compares each image pixel against its corresponding bias value. When the former exceed the latter by at least `fp->tp.bparm[3]`, the image pixel and its 8 neighbors are set equal to PIXEL_BAD. This routine must be followed immediately by a call to ***FEPtimedBias1Mean***. The pair of routines are called a total of `fp->tp.bparm[1]` – `fp->tp.bparm[0]` – 1 times in "whole-frame" mode.
- ***FEPtimedBias1Mean***—examines each image pixel value, *p*, and, unless their value is PIXEL_BAD, replaces the corresponding bias value, *b*, by $(n*b+p)/(n+1)$, where *n* is the post-conditioning exposure index, `fp->expnum - fp->tp.bparm[0] + 1`.
- ***FEPtimedBias1Median***—is only called in "whole-frame" mode (Figure 183). It will be called when the exposure number is `fp->tp.bparm[0]` and when `fp->tp.bparm[2]` is non-zero. It examines all 3×3 blocks of bias values. When the central value is less than all but one of its neighbors by at least `fp->tp.bparm[2]`, it replaces the central value by the median of its neighbors. It should only be invoked when it is suspected that the image pixels contain anomalously low values that would otherwise corrupt the "whole-frame" bias map.
- ***FEPtimedBias2Proc***—is called from ***fepTimedBias*** once the image map contains a set of strips of pixels from multiple exposures of the same CCD rows. It extracts each set of pixels into a vector, calls ***mean*** or ***fractile*** to compute the bias value, and stores the result into the bias map. When the strips have been processed, it calls ***FEPtimedBiasParity*** to update the bias parity plane.
- ***mean*** is a utility function that returns the truncated mean of an array of values. It first computes the mean and RMS variance, then re-computes the mean, rejecting those values that differ from the first-order mean by more than a constant times the RMS variance. It is only called in "strip" mode.
- ***fractile*** is a utility function that sorts an array of values, and returns the value that is indexed by a constant, where the index is 0 for the smallest value, 1 for the next smallest, etc. Note that this function is used in both modes—in "whole-frame" mode, it is called by ***FEPtimedBias1Median*** to compute the median value of neighboring pixels; in "strip" mode, it is called from ***FEPtimedBias2Proc*** to compute the bias itself.

22.4 Global Variables

The following FEPparm fields, defined in *fepCtl.h* and *fepBep.h*, and is invariably addressed by the *fp* pointer parameter, are used by all timed exposure bias modes:

bepCmd	latest command received from BEP	
br	pointer to bias calibration parameters	
bias0[4]	average overclocks for first bias frame	
biassum	sum of the 4 *bias0* values	
ex	current FEPexpRec record	
d0clk[4]	change in average overclock since last exposure	
expnum	current exposure number	
timestamp	microsecond timer value at start of expnum	
expcount	number of bias frames processed	
fepStatus	FEP status reported to BEP	
biasflag	=1 if bias has been computed	
flags	flag bits:	
	FP_SUSPEND	BEP has sent BEP_FEP_CMD_SUSPEND
	FP_PAST_EOR	FEP hardware has finished with the current frame
	FP_TERMINATE	BEP has sent BEP_FEP_CMD_STOP
	FP_DONE	BEP is terminating normally
image	pointer to start of current image row	
nextexpnum	the next exposure index that the FEP is to process	
parity	pointer to 4096-element bias parity table	
quadrants	the number of DEA output nodes being sampled	
tp	exposure parameter block (see Table 46)	
bparm[5]	mode-dependent parameters	
btype	type of bias calibration desired	
initskip	number of initial frames to ignore	
ncols	number of CCD columns clocked	
noclk	number of overclocks per node per row	
nrows	number of CCD rows clocked	
nskip	2-exposure alternation factor	
quadcode	output node clocking mode	

22.5 Scenarios

The following paragraphs describe the basic functions performed by ***fepTimedBias*** during timed exposure bias calibrations, which are determined by the fields in the parameter block, `fp->tp` shown in Table 46.

TABLE 46. Parameters used by fepTimedBias

Field Type	*Field Name*	*"Whole-Frame" Mode*	*"Strip" Mode*
unsigned	`nrows`	Number of bias rows to be calibrated.	
unsigned	`ncols`	Number of pixels per output node per row	
fepQuadCode	`quadcode`	Output node configuration, i.e., ABCD, AC, or BD.	
unsigned	`noclk`	Number of overclocks per row per output node	
fepBiasType	`btype`	`FEP_BIAS_1`	`FEP_BIAS_2`
int	`bparm[0]`	Number of conditioning exposures (PHASE2)	Number of exposures per pixel
int	`bparm[1]`	Number of approximation-to-mean exposures (PHASE3)	=0 to use ***mean*** =1 to use ***fractile***
int	`bparm[2]`	Rejection threshold for low-pixel elimination (immediately prior to PHASE3)	For ***mean***, specifies σ rejection criterion. For ***fractile***, index of sorted pixel array.
int	`bparm[3]`	Threshold for event rejection (PHASE3)	Ignored
int	`bparm[4]`	Rejection threshold for approximation-to-mean	Ignored
unsigned	`nskip`	Exposure skip factor; if non-zero, don't use those with "non-standard" exposure times for bias calibration.	

The bias calibration algorithms themselves are presented in some detail in the ACIS report entitled "*CCD Bias Level Determination*" by Rita Somigliana and Peter Ford, ACIS part #36–56012–02, MIT CSR, Revision 2, May 30, 1995.

22.5.1 Use 1: Calculate Bias using a Whole-Frame Algorithm

fepTimedBias calls ***FEPtimedBiasInit*** to check the `fp->tp` parameter block, to initialize the parity table (`fp->parity`), and set the hardware registers. It then loops, waiting for the next exposure to be received from the DEA. Subsequent processing occurs in three distinct phases, with an optional 2a phase invoked in special circumstances.

- *Phase 1: initialization.*—the image pixels p_i of the first exposure frame are copied directly to the bias buffer, forming the zeroth order bias values, b_i^0.

$$b_i^0 = p_i \qquad \text{(EQ 1)}$$

- *Phase 2: conditioning*—a series of exposures are examined, n=1,N. In each, an image pixel will replace the corresponding bias pixel if the image pixel is lower in value.

$$\begin{aligned} b_i^n &= p_i && \text{if } p_i < b_i^{n-1} \\ &= b_i^{n-1} && \text{otherwise} \end{aligned} \qquad \text{(EQ 2)}$$

  After a number of exposures, typically 5–10 for the anticipated radiation levels, the chance of any bias map value being influenced by radiation is vanishingly small.[1] On the contrary, the values will typically be lower than the "true" bias values. If any of the pixel values in Phase 1 or 2 is anomalously low, e.g. more than 4σ lower than the mean, (σ is the standard deviation in measured values of that pixel), that value will become the bias value at the end of Phase 2, and must be filtered out by the optional Phase 2a before proceeding to Phase 3.

- *Phase 2a: fix-up by median filtering*—no new exposures are examined during this optional processing phase. Bias values that are much lower than their neighboring values are identified and corrected by median filtering, i.e.,

$$b_i^N = M\left[\{b_{i,i\pm1}^N\}\right] \qquad \text{if } b_i^N < \{b_{i,i\pm1}^N\} + BPARM[2] \qquad \text{(EQ 3)}$$

  where M[] represents the median of the 8 surrounding pixels.

- *Phase 3: approximation to the mean*—a further series of exposures, m=N+1,M, is examined. Each exposure is first examined for events, i.e. pixels that exceed their corresponding "conditioned" bias values by more than a threshold supplied in the parameter block. Once found, that pixel, and its immediate neighbors are set to a special "illegal" value.

$$p_{i\pm0,1}^m = 4095 \qquad \text{if } p_i^m > b_i^{m-1} + BPARM[3] \qquad \text{(EQ 4)}$$

  The same exposure is examined again. This time, only non-illegal pixels are considered. Those that do not exceed their bias values by more than a certain threshold are used to refine the bias level by a "running average" algorithm.

---

1. Care must be taken when observing a bright target during bias calibration that the pile-up in any single pixel doesn't violate this condition. At XRCF, bias calibration should probably be performed with the source turned off or with ACIS translated away from the focal axis.

$$b_i^m = \frac{m-N}{(m-N+1)} b_i^{m-1} + \frac{1}{(m-N+1)} p_i^m \qquad \text{(EQ 5)}$$

if $p_i^m \neq 4095$ and $p_i^m < p_i^{m-1} + BPARM[4]$, and $b_i^m = b_i^{m-1}$ otherwise.

Before processing each line of image pixels, ***FIOgetNextCmd*** is called. If it returns TRUE, ***fepHandleCmd*** is called to process a single BEP command.

After the last exposure has been processed, ***FEPtimedBiasParity*** is called to construct a bias parity buffer that contains a single parity bit for each pixel in the bias map. This will be used by the FEP hardware to detect single-bit flips in the bias map during subsequent timed exposure science runs.

Finally, ***fepTimedBias*** exits with writes to the image map disabled. It returns to its command mode and waits for more BEP commands.

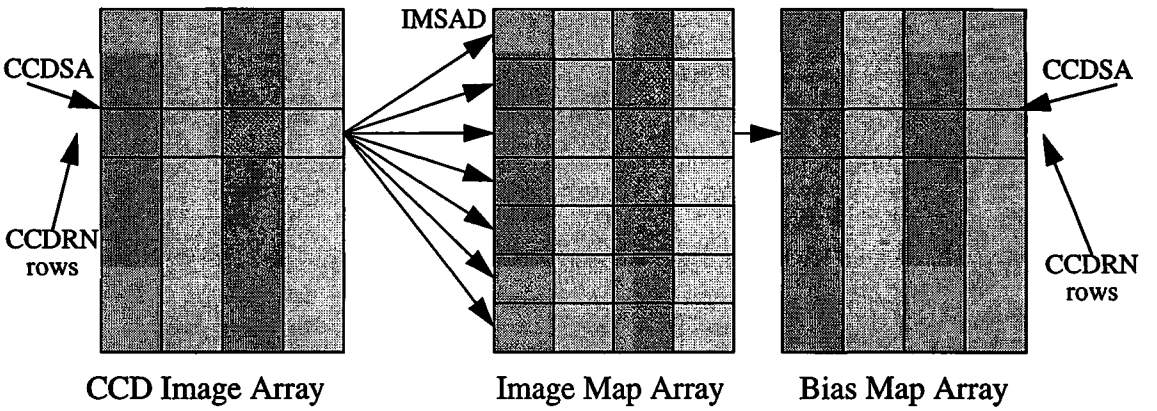
22.5.2 Use 2: Calculate Bias using a Strip Algorithm

fepTimedBias calls ***FEPtimedBiasInit*** to check the `fp->tp` parameter block, to initialize the parity table (`fp->parity`), and set the hardware registers. It then loops over calls to ***FIOgetExpInfo***, waiting for the next exposure to be received from the DEA.

The algorithms require each image pixel to be exposed several times, and its values after each exposure must be available simultaneously. Since there is insufficient memory available to store more than a single copy of the image map, the pixels must be processed a few at a time. On each exposure, the FEP hardware is therefore commanded to write only a strip of CCD image pixels to the image map. On subsequent exposures, the registers are adjusted so that each strip is written into a different part of the image map, as shown in Figure 185.

FIGURE 185. The Relation between CCD strips, Image strips, and exposures

A series of CCD exposures (left) is made and a single strip of CCDRN rows is copied to the image map (center), starting at row IMSAD. After each exposure, IMSAD is advanced by CCDRN. When the image map is full, it is processed and the resulting bias values are stored into the bias map (right), starting at row CCDSA. CCDSA is then advanced by CCDRN, IMSAD is reset to zero, and the process repeated for the next set of CCD strips. The register name mnemonics are taken from the ACIS SI Digital Processor Assembly, Hardware Specification and System Design, ACIS part #36–02104, Rev. A, MIT-CSR, October 4, 1995.



The choice of strip size is determined by the number of exposures desired (the value of `fp->tp.bparm[0]`), i.e. its size in rows is the smallest integer that does not exceed 1024 divided by the number of exposures. Once the image map is filled with strips, the program copies the corresponding pixel value in each strip to an array in D-cache for faster access. It then operates on the array values and uses either their truncated mean or their fractile as the bias value, which it stores in the bias map.

22.5.2.1 The Iterated Mean Algorithm

This algorithm, described in Section 3.2.4 of 36–56012–02, is implemented by the function ***mean*** (see Section 22.6.11). It takes the N pixel values p_i, i=0,N–1, and computes their mean value $\bar{p}$ and variance σ^2:

$$\bar{p} = \frac{1}{N}\sum_{i=0}^{N-1} p_i \qquad \text{(EQ 6)}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1} (p_i - \bar{p})^2 \qquad \text{(EQ 7)}$$

If the value of `BPARM[2]` is zero, ***mean*** returns $\bar{p}$ as the bias value. Otherwise, it inspects the p_i and removes any that do not satisfy the condition

$$|p_i| \le (\sigma \cdot \text{BPARM[2]}) \qquad \text{(EQ 8)}$$

Finally, it recomputes $\bar{p}$ from the remaining p_i using Eq. 6, and returns that as the bias value.

22.5.2.2 The Fractile Algorithm

This algorithm, which is a generalization of the Median method described in Section 3.2.5 of 36–56012–02, is implemented in the function ***fractile*** (see Section 22.6.12). It takes the N pixel values p_i, i=0,N–1, sorts them into ascending order, and returns the value indexed by the value of `bparm[2]`. For instance, if N were 11 (it is usually 1024), and `bparm[2]` were 5, and the pixel values p_i were,

```
212 216 205 1041 208 217 211 214 215 206 210
```

their bias value would be 212, since, when the values are sorted into ascending order,

```
205 206 208 210 211 212 214 215 216 217 1041
```

that is the value of the element p_5.

22.6 Specification

This section describes the functions that are local to the ***fepTimedBias*** unit. The only external is ***fepTimedBias*** itself which is called from ***fepCtl***. The `FEPparm` structure is defined in *fepCtl.h* and *fepBep.h*, along with several pixel access macros. The interface to the FEP I/O library is described in Section 39.0, and data and messages exchanged between FEP and BEP are described in Section 4.10.

22.6.1 fepTimedBias()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

```
FEPparm *fp
```

Description:

fepTimedBias is called from ***fepCtl*** with a single argument: the pointer `fp` to the `fepParm` structure. It is responsible for all FEP actions associated with a timed exposure bias calibration. It performs the following actions:

- Copies the address of the `par` array in its D-cache stack, to `fp->parity`, where it will be initialized within ***FEPtimedBiasInit*** and used within ***FEPtimedBiasParity***.
- Calls ***FEPtimedBiasInit*** to initialize various `fepParm` fields and hardware registers. If ***FEPtimedBiasInit*** returns `FEP_CMD_NOERR`, ***fepTimedBias*** calls ***fepAckCmd*** and continues. Otherwise, it calls ***fepNackCmd*** to pass the error code to the BEP, indicating that the command has failed, and ***fepTimed-Bias*** then returns to ***fepCtl***
- Loops over exposures until either the `FP_DONE` or the `FP_TERMINATE` flag is set in `fp->flags`.
  - Entirely ignores the first `fp->tp.initskip` exposure frames.
  - When the bias type (`fp->tp.btype`) is "strip" mode (`FEP_BIAS_2`), sets the `mode` parameter to BIAS2. Otherwise, this is "whole-frame" mode and the `mode` parameter is set according to the number of fully-processed exposures (`fp->expcount`) and the `fp->tp.bparm` values. The first exposure will be processed as `BIAS1_PHASE1`, the next `fp->tp.bparm`[0] exposures will be processed as `BIAS1_PHASE2`, and the final `fp->tp.bparm`[1] exposures will be processed as `BIAS1_PHASE3`.

  - Calls ***fioWriteImPulseReg***, to set the `IPULSE_ARMNXTACQ` bit in the FEP's image pulse register.
  - Loops over calls to ***FIOgetExpInfo*** until the exposure number changes, i.e. until the hardware begins to write pixels from the next exposure into the image map. During this loop, calls are made to ***FIOgetNextCmd*** to intercept and process commands from the BEP, and ***FIOtouchWatchdog*** to keep the watchdog timer alive.
  - Calls ***FEPtimedBiasExec*** to process the exposure. NOTE: when the CCDs are clocked with two exposure times (i.e. non-zero *`fp->tp.nskip`*), ***FEPtimedBiasExec*** will not be called for those exposures with the initial (i.e. less frequent) exposure time.
- In "whole-frame" mode, calls ***FEPtimedBiasParity*** to initialize the bias parity buffer.
- Marks the bias map as "good" by setting *`fp->biasmode`* to `TRUE` and *`fp->br.biassum`* to the sum of the 4 elements in the *`fp->br.bias0`* array.

22.6.2 FEPtimedBiasExec()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

BiasMode mode

Description:

This function is called from ***fepTimedBias*** to process a single frame. *mode* selects either a phase of the "whole-frame" algorithm: BIAS1_PHASE1, BIAS1_PHASE2, or BIAS1_PHASE3, or the "strip" algorithm, BIAS2.

In the "whole-frame" algorithm, the routine loops over image frame rows, calling ***FIOgetNextCmd*** once per row to catch incoming commands from the BEP and, when detected, ***fepHandleCmd*** is called to process them. ***FEPtimedBiasExec*** then sums the overclocks and calls a subroutine to process the image pixels, depending on the *mode* value:

BIAS1_PHASE1—***FEPtimedBias1Copy*** copies a row of image pixels to the bias map without any change.

BIAS1_PHASE2—***FEPtimedBias1Cond*** reads a row of image pixels and uses them to replace the corresponding bias map values when the latter are larger than the former. The result of performing this operation over several consecutive exposures is to "condition" the bias map values, removing any contamination from CCD events.

BIAS1_PHASE3—first, ***FEPtimedBias1ZapEvent*** is called to mark all image map pixels that may contain events, followed by ***FEPtimedBias1Mean*** to use the unmarked pixels to update the bias values.

In the "strip" algorithm, when *mode* has the value FEP_BIAS_2, ***FEPtimedBiasExec*** does very little until the last of a set of strips has been written to the image map. It merely adjusts the hardware pointers via calls to ***FIOsetImageMapRowStart*** and ***FIOsetImageMapRowLength*** to march the strips down the image map, and to ***FIOsetCcdRowStart*** to select different rows of the CCD. After the image map is full of strips, ***FEPtimedBiasExec*** sums the overclocks of the last frame and then calls ***FEPtimedBias2Proc*** to update the bias map.

22.6.3 FEPtimedBiasInit()

#include fepCtl.h

Scope: Science

Return type: fepCmdRetCode

Arguments

*FEPparm *fp*

Description:

This function is called from ***fepTimedBias*** to verify the contents of the FEPparmBlock, *fp->tp*, and to perform the following initializations:

- *fp->quadrants* are set to 2 or 4, depending on the value of *fp->tp.quadcode*.
- *fp->parity*[0] *through fp->parity*[4095] are either set to PARITY_EVEN or to PARITY_ODD (defined in *fepCtl.h*) according to the bit parity of the binary integers 0–4095.
- FEP hardware registers are set by calls to ***FIOsetCcdRowStart***, ***FIOsetImageMapRowStart***, ***FIOsetImageMapRowLength***, and ***fepSetAddrMode***. Thresholding and bias parity error detection are disabled. Overclock processing is enabled.

If an error is detected, ***FEPtimedBiasInit*** returns a *fepCmdRetCode* value as defined in *fepBep.h*; otherwise it returns FEP_CMD_NOERR.

22.6.4 FEPtimedBiasParity()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

unsigned rowstart

unsigned nrows

Description:

In "strip" mode, this function is called from ***FEPtimedBias2Proc*** to update *nrows* in the bias parity buffer, starting at row *rowstart*, from the corresponding values in the bias map. In "full-frame mode", it is called from ***FEPtimedBias*** to compute the entire parity buffer at the end of the bias calibration run. Each 12-bit bias map value is used as an index into the *fp->parity* array, whose elements are either PARITY_EVEN or PARITY_ODD, according to the parity of the index. For instance, the number fifteen is represented by the bit pattern 01111, which contains an even number of '1's. Its parity is therefore even, so *fp->parity*[15] = PARITY_EVEN.

Precondition:

fp->parity must point to an array of 4096 32-bit values, which have been initialized to PARITY_EVEN if the index number possesses even parity, or to PARITY_ODD if odd.

22.6.5 FEPtimedBias1Copy()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***FEPtimedBiasExec*** in BIAS1_PHASE1 to copy a single row of pixels from the image map (located at *fp->image*) to the bias map (located at *fp->image* + BIAS_OFFSET).

Since this routine is only called for the first exposure of the bias calibration sequence, no overclock correction factor need be applied.

22.6.6 FEPtimedBias1Cond()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***FEPtimedBiasExec*** in BIAS1_PHASE2 to inspect a single row of pixels from the image map (located at *fp->image*) and update the corresponding values in the bias map (located at *fp->image* + BIAS_OFFSET) when the former are smaller than the latter.

Before making the comparison with the bias pixel, each image pixel is corrected for any change in average overclock by subtracting the *fp->ex.dOclk* element appropriate to the pixel's output node. Since the correction factor is based on the average overclocks from the previous exposure, this can only compensate for slow changes in the analog system.

22.6.7 FEPtimedBias1Mean()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***FEPtimedBiasExec*** in BIAS1_PHASE3 to read a single row of image pixels (located in *fp->image* - PIXEL_STRIDE) and update the corresponding bias values (located in *fp->image* + BIAS_OFFSET - PIXEL_STRIDE). Image pixels with the PIXEL_BAD value are skipped since they have been identified in a prior call to ***FEPtimedBias1ZapEvents*** as possibly containing events.

Before making the comparison with the bias pixel, each image pixel is corrected for any change in average overclock by subtracting the *fp->ex.dOclk* element appropriate to the pixel's output node. Since the correction factor is based on the average overclocks from the previously processed exposure, this can only compensate for slow changes in the analog system.

When the pixel value p exceeds the corresponding bias value b by not more than *fp->bparm*[4], b is replaced by $(n*b+p)/(n+1)$, where n is the exposure count, i.e. n=1 for the first exposure of BIAS1_PHASE3, 2 for the second, etc.

Since ***FEPtimedBias1Mean*** is called immediately after ***FEPtimedBias1Cond***, and the latter is capable of nullifying pixels in the preceding row, ***FEPtimedBias1Mean*** must also work on that row, whose first pixel is located at *fp->image* - PIXEL_STRIDE, rather than on the current row, pointed to by *fp->image*.

22.6.8 FEPtimedBias1Median()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***fepTimedBias*** at the end of BIAS1_PHASE2 to identify anomalously low valued pixels in the bias map. It does this by comparing each bias value with those of its 8 neighbors. If the central value is smaller than all but one of its neighbors by more than *fp->tp.bparm*[2], ***FEPtimedBias1Median*** replaces the central value by the median of the neighbors.

This function is only invoked if *fp->tp.bparm*[2] is non-zero. It should only be used if it has been found from a study of previous bias maps that some anomalously low image pixel values will be encountered, since these would otherwise dominate the bias map that is constructed during BIAS1_PHASE2.

22.6.9 FEPtimedBias1ZapEvent()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

unsigned irow

Description:

This function is called from ***FEPtimedBiasExec*** in BIAS1_PHASE3 to identify image pixel values (at location *fp->image*) in row irow that are larger than their corresponding bias values (at location *fp->image* + BIAS_OFFSET) by more than a constant *fp->tp.bparm*[3].

Before making the comparison, each image pixel is corrected for any change in average overclock by subtracting the *fp->ex.dOclk* element appropriate to the pixel's output node. Since the correction factor is based on the average overclocks from the previous exposure, this can only compensate for slow changes in the analog system.

Once identified, the image pixels and their immediate neighbors (8 pixels, or less if the identified pixel is on the edge of the CCD) are reset to PIXEL_BAD so that they can be identified in the ***FEPtimedBias1Mean*** routine.

22.6.10 FEPtimedBias2Proc()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***FEPtimedBiasExec*** in the BIAS2 mode ("strip" processing) when the image map is filled with a series of strips of pixels from the same group of CCD rows from *fp->tp.bparm*[0] consecutive exposures.

Each image pixel is corrected for any change in average overclock by subtracting the *fp->ex.dOclk* element appropriate to the pixel's output node. Since the correction factor is based on the average overclocks from the last exposure that contributed to the current strip, this can only compensate for slow changes in the analog system.

Each set of pixel values, one from each exposure, is copied to a buffer in D-cache, *Data0* for pixels from even-indexed columns and *Data1* for pixels from odd-indexed columns. The value to be stored in the bias map is determined by the value of *fp->tp.bparm*[1].

bparm[1] = 0: ***mean*** is called to compute the mean of the pixel values. When fp->tp.bparm[2] is zero, this becomes the bias map value. When fp->tp.bparm[2] is non-zero, pixels with values that differ from the zeroth-order mean by more than fp->tp.bparm[2] times the RMS variance of the values are eliminated, and the mean of the remainder becomes the bias map value.

bparm[1] = 1: ***fractile*** is called to sort the pixel values into ascending order. The element in this sorted list indexed by fp->tp.bparm[2] becomes the bias map value.

After processing the entire set of strips in the image map, ***FEPtimedBias2Proc*** calls ***FEPtimedBiasParity*** to compute parity flags for each of the new bias map pixel values and store the result in the appropriate section of the bias parity buffer.

22.6.11 mean()

#include fepCtl.h

Scope: Science

Return type: unsigned

Arguments

*unsigned *vec*

unsigned nvec

unsigned nsigma

Description:

This function returns the integer closest to the mean value of the *nvec*-element array *vec*[]. Half-integer values are rounded up. If *nsigma* is non-zero, the standard deviation (σ) of the elements is also calculated, and the mean is re-calculated from those elements of *vec*[] that differ from the original mean value by no more than *nsigma* $\times \sigma$. The algorithm is described in Section 22.5.2.1.

22.6.12 fractile()

#include fepCtl.h

Scope: Science

Return type: unsigned

Arguments

*unsigned *vec*

unsigned nvec

unsigned nrep

Description:

This function sorts the elements of the *nvec*-element array `vec[]` into ascending value, and returns the value of the *nrep*'th element, i.e. *nrep*=0 returns the minimum, *nrep*=*nvec*/2 returns the median, etc. The routine uses Shell's method, which was chosen for its computational efficiency—*nvec* × ln(*nvec*)—and low start-up overhead. The algorithm is described in Section 22.5.2.2.