CSR	**MASSACHUSETTS INSTITUTE OF TECHNOLOGY** **CENTER FOR SPACE RESEARCH** **CAMBRIDGE, MASSACHUSETTS 02139**	
REVISION LOG	**TITLE: Software Detailed Design** **FEP Science Continuously Clocked Bias Calibration**	**DOC. NO.** **36-53227 Rev. A**

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
A	6/18/96	36-623	all	Initial released version. Incorporated comments from initial review. Added support to ignore "n" initial frames.	6/18/96

22.0 FEP Continuously Clocked Bias Calibration (36-53227 A)

22.1 Purpose

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

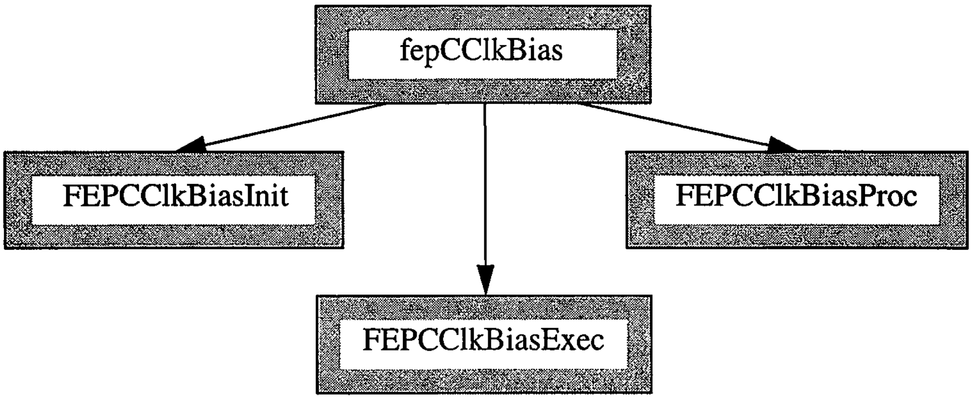
In continuous clocking mode, all pixels within the same column have moved through all rows of CCD image and frame stores, and are therefore characterized by identical values of bias threshold. However, the FEP hardware thresholder requires the bias map to be 2–dimensional as in timed exposure more, although all entries in a given column will be identical. It is therefore particularly simple to compute the continuously clocked bias map —the image map is filled with *N* consecutive rows of pixels (typically 1024), and the bias threshold for each column is determined from the *N* values. The values are replicated in each row of the bias map. If a bias value is subsequently upset by a high-energy event and is detected by its incorrect parity, the correct value can be retrieved from the remaining values in the same column of the bias map.

22.2 Uses

The *fepCClkBias* function has a single mode of operation:

Use 1:: Calculate a continuously-clocked bias map.

FIGURE 190. fepCClkBias Structure



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. *fepCClkBias* makes use of the following functions which call each other in the manner shown in Figure 1:

- ***FEPCClkBiasInit***—is called from ***fepCClkBias*** to validate the `FEPparmBlock`, *`fp->tp`*, and perform all necessary initialization.
- ***FEPCClkBiasExec***—is called from ***fepCClkBias*** to process a pair of 512-row image frames. It sums the overclocks and leaves the image pixels in the image buffer.
- ***FEPCClkBiasProc***—is called from ***fepCClkBias*** to process each column of the image buffer. It copies columns of pixels into a vector, calls ***mean*** or ***fractile*** (defined in the ***fepTimedBias*** module) to compute the bias value, and stores the result into each element of the corresponding column of the bias map. At the same time, it construct a bias parity table in which each bit represents the parity (`EVEN=0, ODD=1`) of the corresponding bias map value.

22.4 Global Variables

The following `FEPparm` fields, defined in *fepCtl.h* and *fepBep.h*, are invariably addressed by the *`fp`* pointer parameter, are used by the bias calculation:

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	
expnum	current "exposure" number	
timestamp	microsecond timer value at start of *expnum*	
expcount	number of 512-row bias frames processed	
fepStatus	FEP status reported to BEP	
biasflag	=1 if bias has been computed	
flags	flag bits:	
	FP_SUSPEND	BEP_FEP_CMD_SUSPEND sent
	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
nextexpnum	the next exposure index that the FEP is to process	
quadrants	the number of DEA output nodes being sampled	
tp	exposure parameter block (see Table 47)	
bparm[5]	mode-dependent parameters	
initskip	number of initial frames to ignore	
ncols	number of CCD columns per output node	
noclk	number of overclocks per output node	
nrows	number of CCD rows between frame markers	
quadcode	output node clocking mode	

22.5 Scenario

The following paragraphs describe the basic functions performed by ***fepCClkBias*** during continuously clocked bias calibrations, which are determined by the fields in the parameter block, `fp->tp` shown in Table 47.

TABLE 47. Parameters used by fepCClkBias

Field Type	*Field Name*	*Description*
unsigned	*nrows*	Number of pixel rows between frame markers.
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
int	*bparm*[1]	=0 to use the Interated Mean algorithm, ***mean*** =1 to use the Fractile algorithm, ***fractile***
int	*bparm*[2]	For ***mean***, specifies σ rejection criterion. For ***fractile***, index of sorted pixel array.

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 #36–56101–02, MIT CSR, Revision 2.1, June 19, 1995.

22.6 Algorithms

22.6.1 The Iterated Mean Algorithm

This algorithm, described in Section 3.2.4 of 36–56101–02, is implemented by the function ***mean*** (see page 1252 of the current document). It takes the N pixel values p_i, i=0,N–1, (N is `2*fp->tp.nrows`, usually 1024 in continuously clocked mode), and computes their mean value $\bar{p}$ and variance σ^2:

$$\bar{p} = \frac{1}{N}\sum_{i=0}^{N-1} p_i \qquad \text{(EQ 9)}$$

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

$$|p_i| \le (\sigma \cdot BPARM[2]) \qquad \text{(EQ 10)}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1} (p_i - \bar{p})^2 \qquad \text{(EQ 11)}$$

Finally, it recomputes $\bar{p}$ from the remaining p_i using Eq. 9, and returns it as the bias value.

22.6.2 The Fractile Algorithm

This algorithm, which is a generalization of the Median method described in Section 3.2.5 of 36–56101–02, is implemented in the function ***fractile*** (see page 1253 of the current document). 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, 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. In practice, N is `2*fp->tp.nrows`, usually 1024.

22.6.3 Use 1: Calculate a Continuously-Clocked Bias Map

fepCClkBias calls ***FEPCClkBiasInit*** to validate the `fp->tp` parameter block, to initialize the parity table (`fp->parity`), and set the hardware registers to write the first half-frame (512 rows) of image pixels into the first half of the image buffer. It then loops over calls to ***FIOgetExpInfo***, waiting for the image frames to be received from the DEA. The first `fp->tp.initskip` frames are ignored.

Immediately a valid frame arrives, ***fepCClkBias*** resets the hardware registers to cause the second set of 512 rows to be written into the remainder of the image buffer. It then calls ***FEPCClkBiasExec*** to monitor the arrival of the `2*fp->tp.nrows` rows and accumulating overclock values. Once the last row has been processed, ***FEPCClkBiasExec*** saves the average overclock for each CCD quadrant and returns. While waiting for each row to arrive, ***FEPCClkBiasExec*** calls ***FIOgetNextCmd*** to see whether a command has arrived from the BEP. If it has, ***FEPCClkBiasExec*** calls ***fepHandleCmd*** and then tests the `FP_TERMINATE` bit in `fp->flags`. If the latter has been set, as a result of receiving a `BEP_FEP_CMD_STOP` command, ***FEPCClkBiasExec*** terminates immediately and ***fepCClkBias*** marks the bias map as bad.

fepCClkBias then calls ***FEPCClkBiasProc*** to calculate the bias for each CCD column, using either the "iterated mean" algorithm, or the "fractile" algorithm. These are discussed in detail in Section 3.2 of the *CCD Bias Level Determination* report described in Section 22.5 above. Once computed, the bias values are written into all elements of that column of the bias map. Their parity is also calculated and written into the bias parity buffer.

22.7 Specification

This section describes the functions that are local to the *fepCClkBias* module. The only external is *fepCClkBias* 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.

22.7.1 fepCClkBias()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

fepCClkBias 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:

- Calls ***FEPCClkBiasInit*** to initialize various `fepParm` fields and hardware registers. It tells the hardware to begin writing the first frame at the top of the image map. If ***FEPCClkBiasInit*** returns `FEP_CMD_NOERR`, ***fepCClkBias*** calls ***fepAckCmd*** and continues. Otherwise, it calls ***fepNackCmd*** to pass the error code to the BEP, indicating that the command has failed, and ***fepCClkBias*** then returns to ***fepCtl***.
- Waits for the first image frame to arrive from the DEA. While waiting, ***fepCClkBias*** executes a tight loop, alternatively calling ***FIOtouchWatchdog*** to keep the watchdog timer alive, and ***FIOgetNextCmd*** to intercept and process commands from the BEP. Once ***FIOgetExpInfo*** indicates that the frame is being copied into the image map, ***fepCClkBias*** calls ***FIOsetImageMapRowStart*** to tell the FEP hardware to write the second frame into the lower half of the image map, starting at row `fp->tp.nrows`, i.e. directly after the first frame.
- Calls ***FEPCClkBiasExec*** to monitor the incoming image rows and sum the overclocks.
- Calls ***FEPCClkBiasProc*** to compute the bias values, and to set the bias parity buffer.

- Unless the FP_TERMINATE bit is set in *fp->flags*, indicating that the BEP has issued a BEP_FEP_CMD_STOP command, it 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, and it saves these values in I-cache via a call to ***FIOsetBiasConfig***.

22.7.2 FEPCClkBiasExec()

#include fepCtl.h

Scope:	Science
Return type:	void

Arguments

*FEPparm *fp*

Description:

This function is called from ***fepCCLKBias*** to process a single frame. It loops over incoming rows of image pixels, calling ***FIOgetNextCmd*** once per row to catch incoming commands from the BEP and, when detected, ***fepHandleCmd*** is called to process them. ***FEPCClkBiasExec*** then sums the overclocks. The process is terminated prematurely if a BEP_FEP_CMD_STOP command is received from the BEP, in which case the FP_TERMINATE flag is set in *fp->flags*.

Once a pair of frames (2 * *fp->tp.nrows rows*) has been processed, the overclock values are summed and saved in *fp->br.bias0*.

22.7.3 FEPCClkBiasInit()

#include fepCtl.h

Scope: Science

Return type: fepCmdRetCode

Arguments

*FEPparm *fp*

Description:

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

- *fp->quadrants* is set to to 2 or 4, depending on the value of *fp->tp.quadcode*.
- 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.

The bias parameter structure, *fp->br* is initialized with "bad" values, i.e. *fp->br.biassum* is set to the unphysical value ~0, and ***FIOsetBiasConfig*** is called to save this in I-cache.

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

22.7.4 FEPCClkBiasProc()

#include fepCtl.h

Scope: Science

Return type: void

Arguments

*FEPparm *fp*

Description:

This function is called from ***fepCClkBias*** once the image map has been filled with two frames of pixel rows. It begins by computing a table of 4096 elements, each either PARITY_EVEN or PARITY_ODD, according to the parity of the integers 0 through 4095. 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.

The image pixels are then examined column by column, a bias value is computed for each column and stored in the corresponding column of the bias map. The bias values are calculated by copying a pair of pixel columns to a pair of value arrays, *Data0* and *Data1*, located in D-cache (for access speed). The algorithm used to compute the bias values is determined by the value of *fp->tp.bparm*[1].

bparm[1]=0: ***mean*** is called to compute the mean of the pixel values,as described in Section 22.6. 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, as described in Section 22.6.2.

Once a pair of bias values has been computed, they are stored in the corresponding columns of the bias map. Then their parities are computed and, every 32 columns, a column of 32-bit parity flags is stored in the bias parity map. It is important to store the bias values and their parity flags as soon as they are calculated since their maps are located in bulk memory which is subject to single-event upsets (SEUs). By saving the values in all rows simultaneously, elements that have suffered SEUs can be replaced at a later time by undamaged duplicate values.