CSR	MASSACHUSETTS INSTITUTE OF TECHNOLOGY CENTER FOR SPACE RESEARCH CAMBRIDGE, MASSACHUSETTS 02139	
REVISION LOG	**TITLE: Software Detailed Design** **FEP-BEP Protocols**	**DOC. NO.** **36-53204.03** **Rev. B**

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
-	3/30/95	-	all	New document	
A	5/11/95	36-254	all	Incorporate review comments	RFG 5/22/95
B	6/17/96	36-623	132	Added "*initskip*" field to **FEPparmBlock** structure	6/18/96

4.10 FEP to BEP Science Protocols and Formats (36-53204.03 B)

4.10.1 Purpose

This chapter describes the Science-level interface between the FEPs and the BEP, as defined in the *fepBep.h* include file. The details of the interface itself are further described in Section 39.0 on page 1158. Here we concentrate on those aspects that directly concern the science processes executing within the FEP.

4.10.2 Uses

The ***fepBep*** interface is used in the following situations:

Use 1:: BEP tells a FEP to start a bias calibration or science run.
Use 2:: BEP tells a FEP to stop a bias calibration or science run.
Use 3:: FEP reports an exposure record to the BEP.
Use 4:: FEP reports an event record to the BEP.
Use 5:: FEP reports a raw pixel record to the BEP.
Use 6:: FEP reports a raw pixel histogram to the BEP.
Use 7:: FEP reports the end of exposure record to the BEP.
Use 8:: FEP reports a fiducial pixel record to the BEP.
Use 9:: FEP reports a bias parity error to the BEP.
Use 10:: BEP loads a parameter block into a FEP.
Use 11:: BEP temporarily suspends and resumes processing in a FEP.
Use 12:: BEP asks a FEP to report its current processing status.
Use 13:: BEP loads one or more fiducial pixel addresses into a FEP.

4.10.3 Organization

FEP exposures, events, fiducial pixels, and bias errors are reported to the BEP via the FEP's ring buffers. All other communication uses the BEP-FEP mailbox (see Section 39.5 on page 1162).

4.10.3.1 Ring Buffer Records

All ring buffer records are an integral number of 32-bit words in length, and start on a 32-bit boundary. Their first 32-bit field is a `fepRingType` code indicating their contents,

```
typedef enum {
  FEP_EXPOSURE_REC,       /* output is FEPexpRec */
  FEP_EXPOSURE_END_REC,   /* output is FEPexpEndRec */
  FEP_EVENT_REC_3x3,      /* FEPeventRec3x3 */
  FEP_EVENT_REC_5x5,      /* FEPeventRec5x5 */
  FEP_EVENT_REC_RAW,      /* FEPeventRecRaw */
  FEP_EVENT_REC_HIST,     /* FEPeventRecHist */
```

```
  FEP_EVENT_REC_1x3,      /* FEPeventRec1x3 */
  FEP_FID_PIX_REC,        /* FEPfidPixRec */
  FEP_ERROR_REC           /* output is FEPerrorRec */
} fepRingType;
```

The transfer itself is performed by a call to ***FIOappendBlock()***.

4.10.3.2 Exposure Start Record

When the FEP software detects the arrival of a new image frame from the DEA, it appends a copy of the `FEPexpRec` structure to the ring buffer.

```
typedef struct {
  fepRingType type;           /* = FEP_EXPOSURE_REC */
  unsigned expnum;            /* exposure number */
  unsigned timestamp;         /* time stamp */
  unsigned short bias0[4];    /* initial overclocks */
  short dOclk[4];             /* changes in overclock */
} FEPexpRec;
```

`expnum` — the frame counter value of this new exposure, as reported by the FEP hardware.

`timestamp` — the value of the 1 MHz system clock as latched by the FEP hardware upon arrival of the first pixel of the new frame.

`bias0[4]` — the average DEA output node overclock values that were derived from the initial frame of the most recent bias calibration.

`dOclk[4]` — the 4 overclock correction factors to be applied to each DEA output node of the new frame, where index 0 corresponds to Output Node A, 1 to Output Node B, 2 to Output Node C, and 3 to Output Node D (NOTE: If any of the output nodes are unused for a given mode, the corresponding overclock correction factor is also unused, and is set to 0). Each of these values consists of a raw computed overclock value for the output node, minus the corresponding `bias0`.

4.10.3.3 Exposure End Record

When the FEP ends its processing of an image frame, it appends a copy of the `FEPexpEndRec` structure to the ring buffer.

```
typedef struct {
  fepRingType type;         /* = FEP_EXPOSURE_END_REC */
  unsigned expnum;          /* exposure number */
```

```
  unsigned thresholds;     /* # of threshold crossings*/
  unsigned parityerrs;     /* # of bias parity errors */
} FEPexpEndRec;
```

`expnum` the frame counter value of this new exposure, as reported by the FEP hardware.

`thresholds` the number of threshold crossings detected in the frame.

`parityerrs` the number of bias parity errors detected in the frame.

4.10.3.4 3x3 Event Record

When the FEP software detects a threshold event, it appends a copy of the `FEPeventRec3x3` structure to the ring buffer.

```
typedef struct {
  fepRingType type;          /* = FEP_EVENT_REC_3x3 */
  unsigned short row, col; /* center pixel address */
  unsigned short p[3][3];  /* pixel values */
  unsigned short b[3][3];  /* bias values */
} FEPeventRec3x3;
```

`row` the row index of the center pixel. The first row in FEP image memory has row index 0, and the index increases by 1 for each subsequent row.

`col` the column index of the center pixel. The first pixel from the first DEA output node has column index 0, and the index increases by one for each subsequent pixel.

`p[3][3]` the 9 pixel values, `p[rows][cols]`, including the central value, `p[1][1]`. Each 12 low-order bits contain the pixel value, with the remaining 4 high-order bits set to zero.

`b[3][3]` the 9 bias values, `b[rows][cols]`, including the central value, `b[1][1]`. Each 12 low-order bits contain the bias value, with the remaining 4 high-order bits set to zero. A bias value of 4094 indicates that the original bias value has suffered a parity error since the most recent bias calibration. A bias value of 4095 indicates that the corresponding image pixel is a member of the ***Bad Pixel List***.

4.10.3.5 5x5 Event Record

When the FEP software detects a threshold event, it appends a copy of the `FEPeventRec5x5` structure to the ring buffer.

```
typedef struct {
  fepRingType type;          /* = FEP_EVENT_REC_5x5 */
  unsigned short row, col; /* center pixel address */
  unsigned short p[3][3];  /* pixel values */
  unsigned short b[3][3];  /* bias values */
  unsigned short pe[16];   /* edge pixel values */
  unsigned short be[16];   /* edge bias values */
} FEPeventRec5x5;
```

`row` — the row index of the center pixel. The first row in FEP image memory has row index 0, and the index increases by 1 for each subsequent row.

`col` — the column index of the center pixel. The first pixel from the first DEA output node has column index 0, and the index increases by one for each subsequent pixel.

`p[3][3]` — the center 9 pixel values, `p[rows][cols]`, including the central pixel, `p[1][1]`. Each 12 low-order bits contain the pixel value, with the remaining 4 high-order bits set to zero.

`b[3][3]` — the center 9 bias values, `b[rows][cols]`, including the central bias, `b[1][1]`. Each 12 low-order bits contain the bias value, with the remaining 4 high-order bits set to zero. A bias value of 4094 indicates that the original bias value has suffered a parity error since the most recent bias calibration. A bias value of 4095 indicates that the corresponding image pixel is a member of the ***Bad Pixel List***.

`pe[16]` — the 16 outer pixel values—`pe[0]` is 2 rows and 2 columns before the center, `pe[1]` is two rows and 1 column before it, etc. Each 12 low-order bits contain the pixel value, with the remaining 4 high-order bits set to zero.

`be[16]` — the 16 outer bias values—`be[0]` is 2 rows and 2 columns before the center, `be[1]` is two rows and 1 column before it, etc. Each 12 low-order bits contain the bias value, with the remaining 4 high-order bits set to zero. A bias value of 4094 indicates that the original bias value has suffered a parity error since the most recent bias calibration. A bias value of 4095 indicates that the corresponding image pixel is a member of the ***Bad Pixel List***.

4.10.3.6 1x3 Event Record

When the FEP software detects a threshold event, it appends a copy of the `FEPeventRec1x3` structure to the ring buffer.

```
typedef struct {
  fepRingType type;          /* = FEP_EVENT_REC_1x3 */
  unsigned short row, col;   /* center pixel address */
  unsigned short p[3];       /* pixel values */
  unsigned short b[3];       /* bias values */
} FEPeventRec1x3;
```

`row` — the row index of the center pixel. The first row in FEP image memory has row index 0, and the index increases by 1 for each subsequent row.

`col` — the column index of the center pixel. The first pixel from the first DEA output node has column index 0, and the index increases by one for each subsequent pixel.

`p[3]` — the 3 pixel values, `p[rows]`, including the central value, `p[1]`. Each 12 low-order bits contain the pixel value, with the remaining 4 high-order bits set to zero.

`b[3]` — the 3 bias values, `b[rows]`, including the central value, `b[1]`. Each 12 low-order bits contain the bias value, with the remaining 4 high-order bits set to zero. A bias value of 4094 indicates that the original bias value has suffered a parity error since the most recent bias calibration. A bias value of 4095 indicates that the corresponding image pixel is a member of the ***Bad Pixel List***.

4.10.3.7 Raw Mode Pixel Record

As images are acquired by the FEP, the FEP software appends a copy of the `FEPeventRecRaw` structure to the ring buffer.

```
typedef struct {
  fepRingType type;          /* = FEP_EVENT_REC_RAW */
  unsigned short row;        /* pixel row address */
  unsigned short p[1024];    /* pixel values */
  unsigned short oc[MAX_NOCLK*4];
                             /* overclock pixel values */
} FEPeventRecRaw;
```

`row` — the row index of the center pixel. The first row in FEP image

memory has row index 0, and the index increases by 1 for each subsequent row.

`p[1024]` up to 1024 pixel values. Each 12 low-order bits contain the pixel value, with the remaining 4 high-order bits set to zero.

`oc[]` up to MAX_OCLK overclock pixels per DEA output node. Each 12 low-order bits contain the overclock value, with the remaining 4 high-order bits set to zero.

4.10.3.8 Histogram Mode Record

The FEP reads exposures, accumulating histograms of raw pixel values (`FEP_TIMED_PARM_HIST`). Once the specified number of exposures (`nhist` in `fepParmBlock`) have been processed, the FEP software appends a copy of the `FEPeventRecHist` structure to the ring buffer.

```
typedef struct {
  fepRingType type;          /* = FEP_EVENT_REC_HIST */
  unsigned expfirst;         /* first exposure number */
  unsigned explast;          /* last exposure number */
  unsigned short omin[4];    /* minimum node overclock */
  unsigned short omax[4];    /* maximum node overclock */
  unsigned short omean[4];   /* mean node overclock */
  unsigned ovar[4];          /* node overclock variance */
  unsigned hist[4][4096];    /* pixel histogram */
} FEPeventRecHist;
```

`expfirst` the frame counter value of the first exposure in which the histogram was accumulated, as reported by the FEP hardware.

`explast` the frame counter value of the last exposure in which the histogram was accumulated, as reported by the FEP hardware.

`omin[4]` the minimum value of the overclocks from the 4 CCD nodes while the histogram was being accumulated.

`omax[4]` the maximum value of the overclocks from the 4 CCD nodes while the histogram was being accumulated.

`omean[4]` the mean of the average overclock values from the 4 CCD nodes while the histogram was being accumulated. The average is computed for each node of each frame and these are then averaged over `nhist` frames.

`ovar[4]` the mean of the variances of the overclock values from the 4

CCD nodes while the histogram was being accumulated. The variance is computed for each node of each frame and these are then averaged over `nhist` frames.

`hist[4][4096]` the pixel histogram. The first index denotes the CCD nodes, A-D, and the second the number of 12-bit values found in all rows of the CCD during that particular set of exposures.

4.10.3.9 Fiducial Pixel Record

When fiducial pixels have been defined by a prior `BEP_FEP_CMD_FIDPIX` command, the FEP software appends a copy of the `FEPfidPixRec` structure to the ring buffer whenever that particular fiducial pixel is encountered (in timed exposure event-finding mode only).

```
typedef struct {
  fepRingType type;          /* = FEP_FID_PIX_REC */
  unsigned index;            /* fiducial pixel index */
  unsigned val;              /* fiducial pixel pair */
} FEPfidPixRec;
```

`index` the index of this pair of fiducial pixels in the *`args`* array of row and column addresses passed to the FEP in the most recent `BEP_FEP_CMD_FIDPIX` command (see Section 4.10.4.9 on page 135).

`val` the value of the pair of fiducial pixels—the 12 low-order bits (0-11) contain the value of the even-column-number pixel and bits 16-27 contain the value of the pixel at the same row and one higher column number. The remaining bits are set to zero.

4.10.3.10 Error Record

Immediately the FEP software detects a parity error in the bias map, it appends a copy of the `FEPerrorRec` structure to the ring buffer and immediately changes the stored bias value to 4094. NOTE: if *both* of a pair of bias values in neighboring even/odd columns have experienced a parity error, only a single `FEPerrorRec` record will be generated.

```
typedef struct {
  fepRingType type;          /* = FEP_ERROR_REC */
  unsigned short row, col;   /* pixel address */
  unsigned expnum;           /* exposure number */
  unsigned biasval;          /* the erroneous value */
} FEPerrorRec;
```

`row` the row index of the bad bias pixel. The first row in FEP image

memory has row index 0, and the index increases by 1 for each subsequent row.

`col` — the column index of the bad bias pixel. The first pixel from the first DEA output node has column index 0, and the index increases by one for each subsequent pixel.

`expnum` — the frame counter value of the exposure in which the bias parity error was detected, as reported by the FEP hardware.

`biasval` — the value of the 32-bit register containing one or two bad bias values. Bits 0-11 and 16-27 contain the values, bits 12 and 28 contain the corresponding parity bits, and bits 15 and 31 contains the parity error flags.

4.10.4 BEP-FEP Mailbox Messages

The BEP sends commands to the FEPs using a COMMAND mailbox (see Section 39.5 on page 1162). However, this is not accompanied by a hardware interrupt, so the FEP must arrange to make frequent calls to ***FIOgetNextCmd()*** to poll the mailbox for new business. This arrives in the form of a COMMAND structure,

```
typedef struct {
  unsigned len;              /* number of args + type */
  int type;                  /* type of request */
  unsigned args[];           /* data */
} COMMAND;
```

Several BEP-to-FEP commands, signified by negative `type` values, are handled automatically within ***FIOgetNextCmd()***. The remainder must be handled by the science-level code. When the command has been processed, the FEP fills a second COMMAND structure and passes it back to the BEP with a call to ***FIOwriteCmdReply()***.

4.10.4.1 Start Bias Calibration

The BEP signals to the FEP that it is to start a bias calibration by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_BIAS
command.args[0] = (unused)
```

This command should be preceded by a `FEP_CMD_LOAD_PARAM` command to load a parameter block. Otherwise, the FEP won't know how to perform the calibration. The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_BIAS
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the bias calibration started successfully, or a particular `fepCmdRetCode` if it didn't. The codes are defined in *fepBep.h.*

4.10.4.2 Start a Timed Exposure

The BEP signals to the FEP that it is to start a timed exposure run by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_TIMED
command.args[0] = (unused)
```

This command should be preceded by a `FEP_CMD_LOAD_PARAM` command to load a parameter block. Otherwise, the FEP won't know how to start the run. The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_TIMED
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the timed-exposure science run started successfully, or a particular `fepCmdRetCode` if it didn't. The codes are defined in *fepBep.h.*

4.10.4.3 Start a Continuously Clocked Exposure

The BEP signals to the FEP that it is to start a continuously clocked exposure run by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_CCLK
command.args[0] = (unused)
```

This command should be preceded by a `FEP_CMD_LOAD_PARAM` command to load a continuous clocking parameter block. Otherwise, the FEP won't know how to start the run. The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_CCLK
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the continuously clocked science run started successfully, or a particular `fepCmdRetCode` if it didn't. The codes are defined in *fepBep.h.*

4.10.4.4 Terminate the Current Mode

The BEP signals to the FEP that it is to stop whatever it is doing (please) by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_STOP
command.args[0] = (unused)
```

The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_STOP
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the FEP stopped its processing, or a particular `fepCmdRetCode` if it didn't, e.g. because it was idling at the time. The codes are defined in *fepBep.h*.

4.10.4.5 Load a Parameter Block

The BEP transfers a parameter block to the FEP by sending it the following COMMAND structure:

```
command.len = 18
command.type = BEP_FEP_CMD_PARAM
command.args[0] = type
command.args[1] = nrows
...
```

The `args` array actually contains a copy of the following `FEPparmBlock` structure.

```
typedef struct {
  fepParmType type;     /* parameter block type */
  unsigned nrows;       /* ending pixel row */
  unsigned ncols;       /* # of pixels/row/node */
  fepQuadCode quadcode; /* Quadrant (node) selection */
  unsigned noclk;       /* # overclocks/row/node */
  unsigned nhist;       /* # exposures/histogram */
  fepBiasType btype;    /* bias algorithm type */
  int thresh[4];        /* user specified thresholds */
  int bparm[5];         /* bias calibration params */
  unsigned nskip;       /* exposure skip factor */
  unsigned initskip;    /* # initial frames to ignore */
} FEPparmBlock;
```

```
typedef enum {
  FEP_NO_PARM,          /* no param block specified */
  FEP_TIMED_PARM_RAW,   /* timed raw mode */
  FEP_TIMED_PARM_HIST,  /* timed raw histogram */
  FEP_TIMED_PARM_3x3,   /* timed 3x3 events */
  FEP_TIMED_PARM_5x5,   /* timed 5x5 events */
  FEP_CCLK_PARM_RAW,    /* continuous raw mode */
  FEP_CCLK_PARM_1x3,    /* continuous 1x3 events */
} fepParmType;

typedef enum {
  FEP_QUAD_ABCD,        /* All four nodes in use */
  FEP_QUAD_AC,          /* Only A and C nodes in use */
  FEP_QUAD_BD           /* Only B and D nodes in use */
} fepQuadCode;

typedef enum {
  FEP_NO_BIAS,          /* none */
  FEP_BIAS_1,           /* algorithm #1 */
  FEP_BIAS_2            /* algorithm #2 */
} fepBiasType;
```

Note that the same structure is used to define parameters in timed mode and in continuous clocking mode. Once the FEP has loaded the block, it replies to the FEP as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_PARAM
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the parameter block was successfully loaded, or a particular `fepCmdRetCode` if it wasn't. The codes are defined in *fepBep.h*.

4.10.4.6 Suspend FEP Operations

The BEP signals to the FEP that it is to temporarily suspend its operations by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_SUSPEND
command.args[0] = (unused)
```

The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_SUSPEND
reply.args[0] = status
```

where *status* is FEP_CMD_NOERR if the FEP suspended its current task, or a particular fepCmdRetCode if it didn't, e.g. because it was idling at the time. The codes are defined in *fepBep.h.*

4.10.4.7 Resume FEP Operations

The BEP signals to the FEP that it is to resume its operations by sending it the following COMMAND structure:

```
command.len = 2
command.type = BEP_FEP_CMD_RESUME
command.args[0] = mode
```

where mode is the FEP operating mode that is to resume. The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_RESUME
reply.args[0] = status
```

where *status* is FEP_CMD_NOERR if the FEP resumed its suspended task, or a particular fepCmdRetCode if it didn't, e.g. because it hadn't been suspended. The codes are defined in *fepBep.h.*

4.10.4.8 Command a FEP to return its status

The BEP signals to the FEP that it is to return its current processing status by sending it the following COMMAND structure:

```
command.len = 1
command.type = BEP_FEP_CMD_STATUS
command.args[0] = (unused)
```

The reply sent to the FEP is as follows:

```
reply.len = 4
reply.type = BEP_FEP_REPLY_STATUS
reply.args[0] = mode
reply.args[1] = biasflag
reply.args[2] = biasparityaddr
reply.args[3] = bias0[0] and bias[01]
reply.args[4] = bias0[2] and bias0[3]
```

where *mode* is the command.type of the command that is currently executing in the FEP (or zero if it is idle), *biasflag* is TRUE if the FEP contains a valid bias map, or FALSE if it doesn't, and *biasparityaddr* is the address, in the FEP's address space, of the start of

its bias parity plane. *bias0*[4] is an array of unsigned 16-bit average overclock values from the first frame of the bias calculation, packed into two 32-bit ring buffer words.

4.10.4.9 Load one or more fiducial pixel addresses into a FEP

Fiducial pixels are particular CCD pixels whose values are always to be reported by the FEP in timed exposure event modes. The BEP passes one or more fiducial pixel addresses to the FEP by sending it the following COMMAND structure:

```
command.len = varying
command.type = BEP_FEP_CMD_FIDPIX
command.args[0] = row_and_column_1
command.args[1] = row_and_column_2
command.args[2] = row_and_column_3
...
```

where the column addresses[1] occupy bits 0-11 of each element of the args array, and the corresponding row addresses occupy bits 16-27. The number of fiducial pixels is one less than `command.len`. The reply sent to the FEP is as follows:

```
reply.len = 2
reply.type = BEP_FEP_CMD_FIDPIX
reply.args[0] = status
```

where *status* is `FEP_CMD_NOERR` if the fiducial pixels were successfully loaded, or a particular `fepCmdRetCode` if they weren't. The codes are defined in *fepBep.h*.

1. Since fiducial pixels are always reported in contiguous even/odd pairs, an odd column address will be decremented upon receipt.