36-53223.0103
Rev. 01


```
/*****************************************************************************
 *
 *     Include File Name:      fep.h
 *
 *     Description:        Inline functions fro FEP
 *
 *     Author:             A. Davis
 *
 *     Configuration #:    36-53223.0103
 *
 *     $Log: fep.h,v $
 * Revision 1.9  1995/06/22  18:37:50  amd
 * Unit tested and updated for release to rev 01
 *
 * Revision 1.8  1995/05/05  21:16:24  amd
 * Changes updated for release to Rev. A review
 *
 * Revision 1.7  1995/02/17  22:25:22  amd
 * Updates to fio.c and fio.h resulting from unit testing.
 * Currently files are compatible with running on the decstation
 * the will have to be modified to run on the target.
 * fep.h redifined constants representing hardware addresses.
 *
 * Revision 1.6  1995/02/10  18:47:01  amd
 * Log current updates. fio.c compiles and is currently being unit tested.
 * Most changes are a result of errors encountered during testing
 *
 * Revision 1.5  1995/01/27  22:19:57  amd
 * Change the function prototype for FIOsetThresholdRegister
 *
 * Revision 1.4  1995/01/27  22:04:19  amd
 * Update with current changes due to code review
 * **Notice** fio.c may still have compile problems
 *
 * Revision 1.3  1995/01/26  16:00:57  amd
 * Include static definitions for functions and prototypes
 *
 * Revision 1.2  1995/01/24  13:53:07  amd
 * NOTE: This version of fio.c won't compile.
 * Checkpoint naming changes.
 *
 * Revision 1.1  1995/01/23  19:27:11  amd
 * Include file for inline functions accessing hardware interface
 *
 *
 *****************************************************************************/

#ifndef fep_h
#define fep_h

/*
 * constants
 */
#define BIAS_MAP_ADDR    ((unsigned *) 0x000001)
#define IMAGE_MAP_ADDR   ((unsigned *) 0x000001)
#define BIAS_CONFIG_SAVE ((unsigned *) 0x800ef000)

/*
 * function prototypes
 */
static unsigned* FIOgetBiasMapPtr();
static void      FIOgetBiasConfig(unsigned *biasrec, unsigned cnt);
static void      FIOsetBiasConfig(unsigned *biasrec, unsigned cnt);
static unsigned* FIOgetBiasParityPlanePtr();
static void      FIOsetBiasParityPlanePtr(unsigned *ptrval);
static unsigned  FIOgetCcdRowStart();
static void      FIOsetCcdRowStart(unsigned row_index);
static unsigned* FIOgetImageMapPtr();
static unsigned  FIOgetImageMapRowIndex();
static unsigned  FIOgetImageMapRowLength();
static void      FIOsetImageMapRowLength(unsigned);
static unsigned  FIOgetImageMapRowStart();
static void      FIOsetImageMapRowStart(unsigned);
static unsigned  FIOgetOffsetReg(unsigned regnum);
static void      FIOsetOffsetReg(unsigned regnum,short value);
static unsigned* FIOgetOverclockBufPtr();
static void      FIOsetOverclockBufPtr(unsigned *ptr);
static short     FIOgetThresholdRegister(unsigned regnum);
static void      FIOsetThresholdRegister(unsigned regnum,short value);
static unsigned  FIOgetThresholdXings();
static unsigned* FIOgetTPlanePtr();
static void      FIOsetTPlanePtr(unsigned* val);

/*
 * inline functions
 */
/* pointer to Bias map */
inline static unsigned* FIOgetBiasMapPtr()
{
    return(BIAS_MAP_ADDR);
}

/* retrieve bias configuration data from Icache */
inline static void FIOgetBiasConfig(unsigned *biasrec, unsigned cnt)
{
     fioReadIcache(BIAS_CONFIG_SAVE,biasrec,cnt);
}

/* write bias configuration data to Icache */
inline static void FIOsetBiasConfig(unsigned *biasrec, unsigned cnt)
{
     fioWriteIcache(biasrec,BIAS_CONFIG_SAVE,cnt);
}

/* pointer to bias parity plane */
inline static  unsigned* FIOgetBiasParityPlanePtr()
{
    return(HwRegs2->biasParityPlane);
}

/* set bias parity plane pointer */
inline static void FIOsetBiasParityPlanePtr(unsigned* val)
{
    HwRegs2->biasParityPlane = val;

}

/* index of start row of CCD */
inline static unsigned FIOgetCcdRowStart()
{
    return(HwRegs1->ccdStartRow);
}

/* set index of start row of CCD */
inline static void FIOsetCcdRowStart(unsigned row_index)
{
    HwRegs1->ccdStartRow = row_index;

```

```
119 }

120 /* pointer to Image map */
121 inline static unsigned* FIOgetImageMapPtr()
122 {
123     return(IMAGE_MAP_ADDR);
124 }

125 /* index to current image map row */
126 inline static unsigned FIOgetImageMapRowIndex()
127 {
128     return(HwRegs1->imRowIndex);
129 }
130 /* get count of CCD rows */
131 inline static  unsigned FIOgetImageMapRowLength()
132 {
133     return(HwRegs1->ccdRowCnt);
134 }

135 /* set CCD row count */
136 inline static void FIOsetImageMapRowLength(unsigned length)
137 {
138     HwRegs1->ccdRowCnt = length;
139
140 }


141 /* index to start row of image map */
142 inline static  unsigned FIOgetImageMapRowStart()
143 {
144     return(HwRegs1->imStartRow);
145 }

146 /* pointer to start row of image map */
147 inline static void FIOsetImageMapRowStart(unsigned index)
148 {
149     HwRegs1->imStartRow = index;
150
151 }

152 /* get direction offset resgisters */
153 inline static unsigned FIOgetOffsetReg(unsigned reg)
154 {
155     return(HwRegs1->offsetReg[reg]);
156
157 }
158 /* Set direction offset resgisters */
159 inline static void FIOsetOffsetReg(unsigned reg,short value)
160 {
161     HwRegs1->offsetReg[reg] = value;
162
163 }

164 /* pointer to Overclock buffer */
165 inline static unsigned* FIOgetOverclockBufPtr()
166 {
167     return(HwRegs2->overclock);
168
169 }

170  /* set pointer to Overclock buffer */
171 inline static void FIOsetOverclockBufPtr(unsigned* val)
172 {
173     HwRegs2->overclock = val;
174 }

175 /* Get threshold resgisters */
176 inline static short FIOgetThresholdRegister(unsigned regnum)
177 {
178     int value;
179
180     /* retrieve value */
181     value = (int)HwRegs1->threshold[regnum];

182     /* mask of high 19 bits and ensure proper sign extension */
183     value <<= 19;
184     value >>= 19;
185
186     /* return 13 bit value */
187     return((short)value);
188 }

189
190 /* Set threshold resgisters */
191 inline static void FIOsetThresholdRegister(unsigned regnum,short value)
192 {
193     HwRegs1->threshold[regnum] = (unsigned)((int)value);
194
195 }
196 /* value of threshold crossings */
197 inline static unsigned FIOgetThresholdXings()
198 {
199     return(HwRegs1->tXings);
200 }

201 /* pointer to T-plane */
202 inline static unsigned* FIOgetTPlanePtr()
203 {
204     return(HwRegs2->tPlane);
205 }

206  /* set T-plane pointer */
207 inline static void FIOsetTPlanePtr(unsigned* val)
208 {
209     HwRegs2->tPlane = val;
210 }
211 #endif
```