36-53223.0101
Rev. 01


```
/*****************************************************************************
*
*       Include File Name:       fio.h
*
*       Description:     Include file for FEP IO Library
*
*       Author:          Ann Davis
*
*       Configuration #: 36-53223.0101
*
*
* $Log: fio.h,v $
 * Revision 1.12  1995/06/22  18:37:46  amd
 * Unit tested and updated for release to rev 01
 *
 * Revision 1.11  1995/05/05  21:16:47  amd
 * Changes updated for release to Rev. A review
 *
 * Revision 1.10  1995/04/14  18:12:35  amd
 * Files passed unit testing
 *
 * Revision 1.9  1995/02/17  22:25:33  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 1.8  1995/02/10  18:47:12  amd
 * Log current updates. fio.c compiles and is currently being unit tested.
 * Most changes are a result of errors encountered during testing
 *
 * Revision 1.7  1995/02/01  15:34:38  amd
 * fio.c has been modified with the current function definitions.
 * The file now compiles and is compatible with fio.h and fep.h
 *
 * Revision 1.6  1995/01/27  22:04:36  amd
 * Update with current changes due to code review
 * **Notice** fio.c may still have compile problems
 *
 * Revision 1.5  1995/01/24  13:53:20  amd
 * NOTE: This version of fio.c won't compile.
 * Checkpoint naming changes.
 *
 * Revision 1.4  1995/01/23  19:32:23  amd
 * Log current changes resulting from review process.
 *
 * Revision 1.3  1995/01/06  22:02:02  amd
 * current revision for design/review
 *
 * Revision 1.2  1995/01/04  21:43:04  amd
 * Log the current version before unit testing
 *
 * Revision 1.1.1.1  1994/12/19  19:18:41  amd
 * Initial cut at the Front End Processors IO library.
 *
 * Revision 1.2  1994/12/19  19:07:14  amd
 * save changes with inlines, compiles as is
 *
 * Revision 1.1  1994/12/09  20:38:41  amd
 * Initial revision
 *
*****************************************************************************/
#ifndef fio_h
#define fio_h

/* define TRUE and FALSE if not already defined
 */
#ifndef TRUE

#define TRUE (1)                /* define TRUE */
#define FALSE (0)               /* define FALSE */

#endif

/* Constants
 */
#define BIAS_PPLANE_ADDR (0x0000000) /* address of Bias Parity Plane */
#define TPLANE_ADDR      (0x0000000) /* address of T-Plane */
#define OVERCLOCK_ADDR   (0x0000000) /* address of overclock buffer */
#define CMD_MBOX_START   (&cmbox) /* address of command mailbox */
#define CMD_MBOX_SIZE    (128)    /* size of command mailbox */

#define CMD_TYPE_DIAG     (0)   /* Highest value of diagnostic commands */
#define CMD_EXECUTE_MEM   (-1)  /* call the function indicated */
#define CMD_READ_ICACHE   (-2)  /* read data from I-cache */
#define CMD_READ_MEM      (-3)  /* read data from memory */
#define CMD_WRITE_ICACHE  (-4)  /* write data to I-cache */
#define CMD_WRITE_MEM     (-5)  /* write data to memory */
#define CMD_LAST_CMD      (-5)  /* last valid diagnostic command */

#ifdef HARDWARE
#define CSI_REG_ADDR      (0xbf400000) /* address of mongoose CSI register */
#else
#define CSI_REG_ADDR      (&csireg)
#endif
#define HARDWARE_REGS_ACTL1  (&hregs1) /* hardware registers */
#define HARDWARE_REGS_ACTL2  (&hregs2) /* hardware registers */

#define DCACHE_START     (0x80000000) /* start of D-cache */
#define DCACHE_END       (0x8007ffff) /* end of D-cache */
#define ICACHE_START     (0x80080000) /* start address of I-cache */
#define ICACHE_END       (0x800effff) /* end address for I-cache */

#define DMAMASK         (0x0c000000) /* mask for DMA type in config register */

#define ICACHEADDR_MASK  (0x0ffffc)   /* Mask of I-cache addresses */
#define ICACHEADDR_R     (0x80000000) /* bit set for I-cache read */
#define ICACHEADDR_W     (0x40000000) /* bit set for I-cache write */


#define INTERRUPT_VAL    (0x1)        /* bit set for BEP interrupt */

#define MAX_ARGS         (20)         /* max num args for CMD_EXECUTE_MEM */
#define MEM_SIZE         (500)
#define MEM_START        ((unsigned *)gmem)   /* start of general memory */
#define MEM_END          ((unsigned *)(gmem + MEM_SIZE)) /* end of general memory
*/
#define REQUEST_MBOX_START (&rmbox)      /* address of request mailbox */
#define REQUEST_MBOX_SIZE  (128)      /* size of request mailbox */

#define WATCHDOG_TIMER_RESET 0xffffffff /* reset value for watchdog timer */
#define XCAUSE_DMADONE   (0x00002000)   /* DMA done mask for cause register */
#define XCAUSE_DMAFAILED (0x00000080)   /* DMA failed mask for cause register */
#define XCAUSE_REG_ADDR  ((unsigned*) (0xbfc00004)) /* (unsigned *)&causereg) */

#define RINGBUFSTART     ((RINGBUF *)&ringbuf)   /* start of ring buffer */
#define RINGBUFEND       ((unsigned *)&ringbuf->data[1000])  /* end of ring buffe
r */
```

```
113  #define RINGBUFSIZE      (1000) /* size of ring buffer */


114  /* Enumerated types and typdefs
115   */

116  typedef int bool;

117  typedef struct s_hardware_reg1
118  {
119    unsigned ctrlReg;             /* control register */
120    union {
121      unsigned pulseReg;          /* pulse register (W/O) */
122      unsigned statusReg;         /* status register (R/O) */
123    } ps;
124    unsigned tXings;              /* T-plane crossings (R/O) */
125    unsigned imRowIndex;          /* image row index (R/O) */
126    unsigned threshold[4];        /* thresholds for nodes A-D */
127    unsigned offsetReg[4];        /* offsets for nodes A-D */
128    unsigned imStartRow;          /* image Starting Row */
129    unsigned ccdStartRow;         /* CCD Starting Row */
130    unsigned ccdRowCnt;           /* number of CCD rows */
131    unsigned spare;               /* spare */
132  } HW_REGS1;

133  typedef struct s_hardware_reg2
134  {
135      unsigned *biasParityPlane;  /* address of parity plane */
136      unsigned *tPlane;           /* address of T-plane */
137      unsigned *overclock;        /* address of overclock buffer  */
138      unsigned watchdog;          /* watchdog timer */
139      unsigned expCtr;            /* exposure counter */
140      unsigned expTimestamp;      /* exposure time stamp */
141      unsigned bepFepcmd;         /* bep to fep command register */
142      unsigned fepBepMbox;        /* fep to bep control register */
143  } HW_REGS2;


144  typedef struct s_csi_reg
145  {
146      unsigned intr_mask;         /* interrupt mask */
147      unsigned xcause;            /* xcause register */
148      unsigned config;            /* config register */
149      unsigned dmaOrigin;         /* dma source address */
150      unsigned dmaDest;           /* dma destination address */
151      unsigned dmaBlockSize;      /* dma transfer size */
152      unsigned t1Count;
153      unsigned t2Count;
154      unsigned spare;
155      unsigned uartCmd;
156      unsigned uartData;
157      unsigned icacheAddr;        /* icache address register */
158      unsigned icacheData;        /* icache data register */
159  } HW_CSI_REG;
160
161
162  /* structure to define the ringbuffer */
163  typedef struct s_ringbuf
164  {
165      unsigned readIndex;         /* read location */
166      unsigned writeIndex;        /* write location */
167      unsigned data[RINGBUFSIZE]; /* ringbuffer */
168  } RINGBUF;

169  /* enum to define the state of the ring buffer */
170  typedef enum
171  {
172      RB_FULL,                    /* Ring buffer full */
173      RB_EMPTY,                   /* Ring buffer empty */
174      RB_NOT_FULL,                /* Ring buffer not full */
175      RB_ERROR                    /* Ring buffer error */
176  } RINGBUFstatus;


177  /* mailbox flags */
178  typedef enum
179  {
180      MBOX_NO_MESSAGE,            /* no new messages pending */
181      MBOX_NEW_MESSAGE,           /* new message pending */
182      MBOX_REPLY_RDY              /* reply to message ready */

183  } MBOXstate;

184  /* definitions of command */
185  typedef struct s_command
186  {
187      int   type;                 /* type of command */
188      unsigned len;               /* number of args + type */
189      unsigned args[CMD_MBOX_SIZE]; /* data in command */
190  } COMMAND;

191  /* definition of request */
192  typedef struct
193  {
194      int   type;                 /* type of request */
195      unsigned len;               /* number of args + type */
196      unsigned args[REQUEST_MBOX_SIZE - 5]; /* data in request */
197  } REQUEST;

198  /* command mailbox */
199  typedef struct s_cmd_mbox
200  {
201      MBOXstate state;            /* state of mailbox */
202      COMMAND   cmd;              /* contents of mailbox */
203  } CMD_MBOX;

204  /* request mailbox */
205  typedef struct
206  {
207      MBOXstate state;            /* state of mailbox */
208      REQUEST   req;              /* contents of mailbox */
209  } REQ_MBOX;

210
211  /* the type of DMA transfers */
212  typedef enum
213  {
214      NO_DMA  =       (0 << 26),   /* DMA inactive */
215      DCACHE_TO_MEM = (0x1 << 26), /* transfer from Dcache to memory */
216      MEM_TO_DCACHE = (0x2 << 26), /* transfer from memory to Dcache */
217      MEM_TO_MEM =    (0x3 << 26)  /* transfer from memory to memory */
218  } TransferType;



219  /*
220   * external variable
```

```
 */
extern volatile bool  DmaDoneStatus;    /* indicates state of last DMA transfer
*/
extern CMD_MBOX cmbox;
extern REQ_MBOX rmbox;
extern HW_REGS1  *HwRegs1;
extern HW_REGS2  *HwRegs2;
extern HW_REGS1  hregs1;
extern HW_REGS2  hregs2;
extern HW_CSI_REG  csireg;
extern RINGBUF   ringbuf;
extern unsigned gmem[];
```

```
/* function prototypes
 */
void      FIOappendData(unsigned* ptr, unsigned wordcnt);
static bool      FIOdmaDone();
void      FIOgetExpInfo(unsigned *expnum, unsigned *timestamp);
void      FIOinit();
void      FIOreadReply(REQUEST *reply);
void      FIOsendRequest(REQUEST request);
static void      FIOtouchWatchdog();
void      FIOwriteCmdReply(COMMAND *cmdreply);
void      fioClearBitCtrlReg(unsigned mask);
void      fioDisableInt();
void      fioEnableInt();
void      fioGetCmd(COMMAND* cptr);
unsigned  fioGetStatusReg();
static bool      fioPollMBox(MBOXstate *mbox,MBOXstate msg);
```

```
/* function prototypes
 */
void      FIOappendData(unsigned* ptr, unsigned wordcnt);
static bool      FIOdmaDone();
void      FIOgetExpInfo(unsigned *expnum, unsigned *timestamp);
void      FIOinit();
void      FIOreadReply(REQUEST *reply);
void      FIOsendRequest(REQUEST request);
static void      FIOtouchWatchdog();
void      FIOwriteCmdReply(COMMAND *cmdreply);
void      fioClearBitCtrlReg(unsigned mask);
void      fioDisableInt();
void      fioEnableInt();
void      fioGetCmd(COMMAND* cptr);
unsigned  fioGetStatusReg();
static bool      fioPollMBox(MBOXstate *mbox,MBOXstate msg);
RINGBUFstatus fioRbStatus(unsigned *available);
void      fioReadMem(const unsigned* src,unsigned* dest, unsigned wordcnt);
void      fioSetBitCtrlReg(unsigned mask);
void      fioWriteMem(const unsigned* src,unsigned* dest, unsigned wordcnt);
void      fioWritePulseReg(unsigned mask);

#ifdef HARDWARE_TEST
void      FIOdmaTransfer(const unsigned* src, volatile unsigned* dest,
                         unsigned wordcnt);
void      fioReadIcache(const unsigned *src, unsigned* dest,unsigned wordcnt);
void      fioWriteIcache(const unsigned* src, unsigned *dest,unsigned wordcnt);
#endif
bool      FIOgetNextCmd(COMMAND *cmd);
```

```
/*
 * inline functions
 */
inline static bool FIOdmaDone()
{
    return(DmaDoneStatus);

}

inline static void FIOtouchWatchdog()
{

    HwRegs2->watchdog = WATCHDOG_TIMER_RESET;

}
inline static bool fioPollMBox(MBOXstate *mbox,MBOXstate msg)
{

    return ((*mbox == msg) ? TRUE : FALSE);

}
#endif
```