```
/*****************************************************************************
 *
 *    Service Processor Module
 *
 *    The Service Processor Module (SPM) attaches to the system bus and
 *    provides various service functions for the system. These functions
 *    include system console interfaces, front panel interface, real time
 *    clock, system clock interrupt service, power supply controls, floppy
 *    disk controller, system wide interrupt dispatcher, various system
 *    monitor functions (power, temperature), and system diagnostic capabil-
 *    ities.
 *    The SPM consists of two boards, the main board which plugs into the CSS
 *    bus backplane, and the real world interface board which contains all
 *    connectors for the serial ports, the power supply control connectors,
 *    and the temperature sensor connectors. The real world interface board
 *    is connected to the main board via a third 96pin din connector at the
 *    bottom of the main board. The real world interface board resides behind
 *    the CSS backplane in the interface slot behind the Service Processor
 *    Main board.
 *
 *****************************************************************************/

/*****************************************************************************
*                          Service Processor Memory Map
*****************************************************************************
*
* 31302928 27262524 23222120 19181716 15141312 111098 7654 3210 - address bits
*
*---------------------------------------------------------------------------
* 0 x x x  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Eprom
* 0 x x x  0 0 0 0  0 0 0 0  0 0 0 0  1 1 1 1  1 1 11 1111 1111 - 512k <----
* 0 x x x  0 0 0 0  0 0 0 0  0 0 0 1  1 1 1 1  1 1 11 1111 1111 - 1M
*---------------------------------------------------------------------------
* 0 x x x  0 0 0 0  0 0 0 0  0 0 1 0  0 0 0 0  0 0 00 0000 0000 Spare
* 0 x x x  0 0 0 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*---------------------------------------------------------------------------
*    0x01000000
*    0x0103ffff
* 0 x x x  0 0 0 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Sram (256k)
* 0 x x x  0 0 0 1  0 0 0 0  0 0 1 1  1 1 1 1  1 1 11 1111 1111
*---------------------------------------------------------------------------
* 0 x x x  0 0 0 1  0 0 0 0  0 1 0 0  0 0 0 0  0 0 00 0000 0000 Spare
* 0 x x x  0 0 0 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*---------------------------------------------------------------------------
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 reserved
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 01 0000 0000 local A/D conv
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 10 0000 0000 local CIO
*---------------------------------------------------------------------------
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 11 0000 0000 reserved for
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 1 11 0000 0000 local devices
*---------------------------------------------------------------------------
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 0 00 0000 0000 aux. SCC 0
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 0 01 0000 0000 aux. SCC 1
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 0 10 0000 0000 aux. CIO
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 0 11 0000 0000 aux. A/D conv
*---------------------------------------------------------------------------
```

```
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 1 00 0000 0000 reserved for
* 0 x x x  0 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  1 1 11 0000 0000 aux. devices
*-------------------------------------------------------------------------------
* 0 x x x  0 0 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 real time clock
* 0 x x x  0 0 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111 and NV ram
*-------------------------------------------------------------------------------
* 0 x x x  0 1 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 reserved
* 0 x x x  0 1 0 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  0 1 0 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 clock control
*-------------------------------------------------------------------------------
* 0 x x x  0 1 0 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0001 reserved
* 0 x x x  0 1 0 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  0 1 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 reserved
* 0 x x x  0 1 1 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Wr cntl 0
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Wr cntl 0 (read)
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0010 0000 Wr cntl 1
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0010 0000 Wr cntl 1 (read)
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0100 0000 Wr cntl 2
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0100 0000 Wr cntl 2 (read)
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0110 0000 Reserved
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 1000 0000 CSS command reg
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 1010 0000 Status reg
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 1100 0000 CSS error reg
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 1110 0000 Dispatcher error
*                                                                register
*-------------------------------------------------------------------------------
* 0 x x x  0 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 1110 0001 Spare
* 0 x x x  0 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 0 0 0  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Map RAM loc 0
* 0 0 0 1  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  1
* 0 0 1 0  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  2
* 0 0 1 1  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  3
* 0 1 0 0  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  4
* 0 1 0 1  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  5
* 0 1 1 0  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  6
* 0 1 1 1  1 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000  "   "   "  7
*-------------------------------------------------------------------------------
* 0 x x x  1 0 0 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Floppy Disk
* 0 x x x  1 0 0 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 0 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 reserved
* 0 x x x  1 0 1 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 0 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 I.D. Idle Queue
* 0 x x x  1 0 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 0 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0001 0000 unused
* 0 x x x  1 0 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 1 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 I.D. Queue Ram
* 0 x x x  1 1 0 0  0 0 0 0  0 0 0 0  0 0 1 1  1 1 11 1111 1111
```

```
*-------------------------------------------------------------------------------
* 0 x x x  1 1 0 0  0 0 0 0  0 0 0 0  0 1 0 0  0 0 00 0000 0000 unused
* 0 x x x  1 1 0 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 1 0 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 I.D. Pointer Ram
* 0 x x x  1 1 0 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------
* 0 x x x  1 1 1 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 Int Dispatcher
* 0 x x x  1 1 1 0  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111 Misc. Rams
*-------------------------------------------------------------------------------
* 0 x x x  1 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 00 0000 0000 reserved
* 0 x x x  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 11 1111 1111
*-------------------------------------------------------------------------------

/*****************************************************************************
 *
 *    EPROM
 *
 *    The EPROM is 8 bits wide and can be read by the processor on data bits
 *    24 to 31. There are 2 sockets for the EPROM, one or both can be populat-
 *    ed with either 27256s (32k x 8 each), or 27512s (64k x 8 each) or
 *    271001s (128k x 8 each). Address bits Add.00 to Add.15 or Add.16 or
 *    Add.17 are used to address the EPROM. The decoding of the high bit is
 *    controlled by the Promsize field in control register 2. This allows the
 *    two EPROM chips to be contiguous regardless of their size. (See Control
 *    register 2 description for details).
 *
 *****************************************************************************/
#define PROMSIZ    0x10000          /* cpu monitor prom*/
#define PROMSTART  0x00000000       /* start of prom area */

/*****************************************************************************
 *
 *    Static Ram
 *
 *    The SRAM is 32 bits wide and consists of 8 or 4 32k x 8 static RAM chips
 *    for 64k x 32 or 32k x 32 bits of memory. It is addressed with address
 *    bits Add.02 to Add.17.
 *
 *****************************************************************************/
#define SRAMSTART 0x01000000
#define SRAMSIZE  0x40000

/*****************************************************************************
 *
 *   Real Time Clock
 *
 *   The real time clock is a MK48T12 chip. This chip has a built in crystal
 *   oscillator and a lithium battery. In addition to a real time clock, it
 *   has 2K bytes of nonvolatile RAM. This RAM can be used to keep config-
 *   uration information. The service processor does not have any switches.
 *
 *****************************************************************************/

#define RTC ((struct rtc *)(0x030007f8))
```

```
/*****************************************************************************
 *
 *   Local A/D converter
 *
 *   The local A/D converter is an MC14442 and a TL431A voltage reference.
 *   It is used to measure the temperature on the Service Processor (i.e. the
 *   cardcage) and the 6 voltages present on the SPM(i.e. +5V main, +5V aux.,
 *   +12V main, +12V aux., -12V main, and -12V aux.)
 *
 *****************************************************************************/

#define ADC_CNTL ((unsigned short*)(0x02000102))
#define ADC_ADATA ((unsigned short*)(0x02000100))
#define ADC_SC    0x0100
#define ADC_CH0   0x0000        /* measures +5v for master system */
#define ADC_CH1   0x0001        /* reference voltage 4.5v must read FF */
#define ADC_CH2   0x0002        /* Measure +5v. aux */
#define ADC_CH3   0x0003        /* Measure +12v */
#define ADC_CH4   0x0004        /* Measure +12v aux */
#define ADC_CH5   0x0005        /* Measure -12v */
#define ADC_CH6   0x000e        /* Measure -12v aux */
#define ADC_CH7   0x000f        /* measure on board temperature. */
#define ADC_EOC   0x8000
#define ADC_MASK  0x00ff        /* mask for valid data. */


/*****************************************************************************
 *
 *    Local CIO
 *
 *    The local CIO is a Zilog 8036, used for various timing functions includ-
 *    ing the CSS bus timeout and the main system clock interrupt.
 *
 *****************************************************************************/

#define LOCCIO ((struct cio *)(0x02000200))

/*****************************************************************************
 *
 *    SCC
 *
 *****************************************************************************/
#define ASCC ((struct ascc *)(0x02000800))
#define AUXASCC0B ((struct ascc *)(0x02000800))
#define AUXASCC0A ((struct ascc *)(0x02000820))
#define AUXASCC1B ((struct ascc *)(0x02000900))
#define AUXASCC1A ((struct ascc *)(0x02000920))

/*****************************************************************************
 *
 *    Clock Control (write only)
 *
 *    The Clock control port is a single bit port which is written with data
 *    bit 0. Data.00 = 0 means the CPU is using its on-board 10Mhz clock,
 *    Data.00 = 1 means the CPU is using the CSS bus clock divided by two as
 *    its clock source. When switching from one clock source to the other,
```

```
 *    approximately 1us after the write to the clock control port, the hard-
 *    ware will issue a reset to the CPU, hold the reset for approximately
 *    500us, and switch clocks during the reset. This is required to meet the
 *    timing specifications of the 68020. In the process, the entire board
 *    will be reset, so any VLSI chips will have to be reinitialized. The
 *    memory, however, will keep its data valid. The state of the clock bit
 *    (i.e. which clock is being used) can be read in the status register. If
 *    the CSS bus clock is not running(e.g. the CSS bus is powered down), the
 *    hardware will not select the CSS clock.
 *
 *      31 ---------- 1 0
 *     +---------------+-+
 *     |       x       |c|
 *     +---------------+-+
 *             |         |
 *             |         +-------------- 0 - on board 10Mhz clock
 *             |                         1 - CSS bus clock
 *             +----------------------- unused
 ************************************************************************/

#define CLKCNTL  ((unsigned *)(0x05000000))

/************************************************************************
 *
 *   CSS Command Register (read only)
 *
 *   The CSS command register is a 32 bit register which allows the local CPU
 *   to read the value written to the SPM via a CSS write to addr 0xXXXXAAYY
 *   by any other CSS module. X's are don't cares and AA will be captured by
 *   the SPM and can be read in the Status register. YY must be of the form
 *   000X XXXX. The SPM only decodes the top three bits of the least signif-
 *   icant byte of CSS address. When another module writes to an address of
 *   the form described above on the SPM, the 68020 recieves an interrupt
 *   informing it that a CSS command has been received. The 68020 will read
 *   the CSS command register and the status register to determine the data
 *   of the command.
 *
 *      31 ---------- 0
 *     +---------------+
 *     |   command     |
 *     +---------------+
 *             |
 *             +----------------------- command to SPM via CSS write
 *
 ************************************************************************/

#define CSSCMD ((unsigned *)(0x07000080))


/************************************************************************
 *
 *    Status Register (read only)
 *
 *    The status register is a 24 bit register, data bits 00 to 07 and 16 to
 *    23 and 24 to 31 can be read by the 68020. Bits 24 to 27 are the bus slot
 *    number given by the position of the SPM in the backplane. For diagnostic
```

```
*    purposes, the SPM does not have its slot id hardwired, so for the board
*    to function properly, the slot id must be read in the status register
*    and written in write control registers 0 and 1. Bit 31 in the Status
*    register is a status bit from the interrupt dispatcher state machine.
*    When this bit is 0, the interrupt dispatcher has suspended its dispatch-
*    ing function and the CPU is allowed to access the various RAMs in the
*    interrupt dispatcher. This is needed for initializing the interrupt
*    dispatcher and for diagnostic purposes. When the bit is a 1, the
*    interrupt dispatcher is in its normal operating mode and the CPU must
*    not access any of the dispatcher RAMs. Bit 30 is an indicator from the
*    floppy disk, floppy ready. Bit 29 is a status bit from the clock select
*    mechanism. When this bit is a 1, the 68020 is using the local clock,
*    when 0, the 68020 is using the CSS clock divided by 2. Bit 28 is reserv-
*    ed. Bits 00 to 07 are part of the CSS error register as follows:
*    Bit 00 has the state of the CSS bus nack signal at the time of the last
*    CSS bus error, bit 01 has the state of the CSS bus ack, bit 02 has the
*    state of the CSS arbiter grant error, bit 03 is the bad command bit.
*    This bit is a 0 if the SPM has detected a CSS bus protocol violation
*    (e.g. data parity is bad), it is a 1 if the SPM has not detected such
*    a violation but it has nevertheless detected a CSS bus error. Bit 04
*    is the state of the CSS arbiter burst signal. Bits 05 to 07 are reserv-
*    ed. Bits 16 to 23 are the CSS data byte 6 at the time of the most recent
*    CSS command write.
*
*      31  30  29 28 27 - 24 23 - 16
*     +--+----+--+--+-------+-------+
*     |ID|FRDY| c| x|slot # | byte 6|
*     +--+----+--+--+-------+-------+
*      |     |   |  |       |       |
*      |     |   |  |       |       +----- CSS data byte 6 at the time of the mos
*      |     |   |  |       |              recent CSS command write.
*      |     |   |  |       +------------- bus slot # determined by position of
*      |     |   |  |                      SPM in backplane
*      |     |   |  +--------------------- reserved
*      |     |   +------------------------ 1 = local clock
*      |     |                             0 = CSS clock
*      |     +-------------------------- floppy ready
*      +---------------------------------- from Interrupt Dispatcher state machine
*                                          0 - dispatcher has been suspended and
*                                              the cpu is allowed access to ID rams
*                                          1 - Int Dispatcher normal mode, cpu
*                                            should not access dispatcher RAM's
*
*      15-8 7-6     5       4       3       2    1    0
*     +----+---+--------+------+-------+-----+----+----+
*     |  x |   | active |burst*|bad cmd|gerr*|back|nack|
*     +----+---+--------+------+-------+-----+----+----+
*       |    |      |       |       |       |    |   |
*       |    |      |       |       |       |    |   +-state of CSS bus nack
*       |    |      |       |       |       |    |     signal at the last
*       |    |      |       |       |       |    |     CSS bus error
*       |    |      |       |       |       |   +-- state of CSS bus ack
*       |    |      |       |       |       +------- state of CSS arbiter
*       |    |      |       |       |                grant error
*       |    |      |       |       +--------------- bad command bit
*       |    |      |       |                        0 - SPM has detected a CSS bus
```

```
 *       |     |      |       |                          protocol violation(data
 *       |     |      |       |                          parity is bad)
 *       |     |      |       |                      1 - SPM has not detected a
 *       |     |      |       |                          violation, but detected a
 *       |     |      |       |                          CSS bus error
 *       |     |      |       +------------------------- state of CSS arbiter
 *       |     |      |                                  burst signal
 *       |     |      +--------------------------------- always a 1 except during
 *       |     |                                         force ack,nack,grant errors
 *       |     +------------------------------------------ reserved
 *       +------------------------------------------------ unused
 *
 ************************************************************************/

#define STATUSREG ((unsigned *)(0x070000a0))
#define STAT_SLOTMASK  0x0f000000
#define STAT_SLOTSH    24
#define STAT_REG_IDRAM 0x80000000
#define STAT_IDLE_FULL 0x10000000    /* if set, idle-que is not full. */

/************************************************************************
 *
 *   CSS Error Register (read only)
 *
 *   When the SPM detects an error on the CSS bus the hardware will interrupt
 *   the 68020. The 68020 will read the CSS error information register to
 *   determine the nature of the error. The conditions which cause such an
 *   error are: bad data parity, bad type parity, destination error, source
 *   error, invalid type, bus nack, or grant error. All error conditions
 *   except the last two will latch in valid data for the CSS error informa-
 *   tion register as shown below. Additional error information is captured
 *   in the status register. When a bus error condition has been latched in
 *   the error register no new error conditions can be latched until after
 *   the bus error register has been read by the 68020.
 *
 *   31 - 27 26-25  24  23-20 19-16  15   14   13-8  7-0
 * +--------+-----+----+-----+-----+----+----+----+----+
 * |CSS data|  x  |bpar|dest | src |derr|serr|btype|bpar|
 * +--------+-----+----+-----+-----+----+----+----+----+
 *      |        |    |     |     |    |    |    |
 *      |        |    |     |     |    |    |    +- bus data parity
 *      |        |    |     |     |    |    |       0 - 7
 *      |        |    |     |     |    |    +------ bus type 0 - 5
 *      |        |    |     |     |    +----------- source error
 *      |        |    |     |     +---------------- destination error
 *      |        |    |     +---------------------- bus source 0 to 3
 *      |        |    +-------------------------- bus dest 0 to 3
 *      |        +------------------------------- bus type parity
 *      +---------------------------------------- reserved
 *     +----------------------------------------- CSS data bits 57,40-43
 *
 ************************************************************************/

#define CSSERROR    ((unsigned *)(0x070000c0))
#define DERR        15
#define SERR        14
```

```
#define BTYPE_SH    8
#define SRC_SH      16
#define DEST_SH     20
#define BPAR_SH     24
#define BPAR_MASK   0x01000000
#define DEST_MASK   0x00f00000
#define SRC_MASK    0x000f0000
#define BTYPE_MASK  0x00003f00
#define TYPE_SH     0x0b
#define TYPE_MASK   0x3800
#define SIZE_MASK   0x0700
#define SIZE_SH     0x08

/*****************************************************************************
 *  Dispatcher Error Register
 *
 *  When the Interrupt Dispatcher detects an error it will suspend its
 *  operation and latch error information in this register.
 *
 *   31 - 24  23  22-20 19-18 17-16 15-12 11-8   7  6-0
 * +--------+----+-----+-----+-----+-----+-----+---+---+
 * | Int Vec|dint| err |  x  | pri |dest | src |int|src|
 * +--------+----+-----+-----+-----+-----+-----+---+---+
 *      |       |     |     |     |     |     |   |
 *      |       |     |     |     |     |     |   +-- i/o bus source slot
 *      |       |     |     |     |     |     |       (only valid if bit 7=1)
 *      |       |     |     |     |     |     +------- i/o bus int
 *      |       |     |     |     |     |              1 - int came from an
 *      |       |     |     |     |     |                  i/o module
 *      |       |     |     |     |     |              0 - int came from a CSS
 *      |       |     |     |     |     |                  bus module
 *      |       |     |     |     |     +------------- CSS bus source slot
 *      |       |     |     |     +------------------- CSS bus dest slot
 *      |       |     |     +------------------------- Interrupt priority
 *      |       |     +----------------------------- reserved
 *      |       +------------------------------------ error code
 *      |                                             (see below for codes)
 *      +------------------------------------------- Directed interrupt
 *     +-------------------------------------------- Int vector number
 *
 *
 *        error information bits 22 21 20
 *                               -- -- --
 *                                x  x  0 - request error
 *                                1  0  x - ack error, bus error on ack response
 *                                0  0  x - ack error, no previous requests
 *                                x  1  1 - recieve error, fix pointers or reset
 *                                          recieve
 *
 *****************************************************************************/

#define DISPERROR     ((unsigned *)(0x070000e0))
#define DISP_ACK_ERR        1       /* insane ack, an ack for no good reason */
#define DISP_REQ_ERR        2       /* request with no int. pending */
/*      unused              3       /* recieve err  */
/*      unused              4       /* */
```

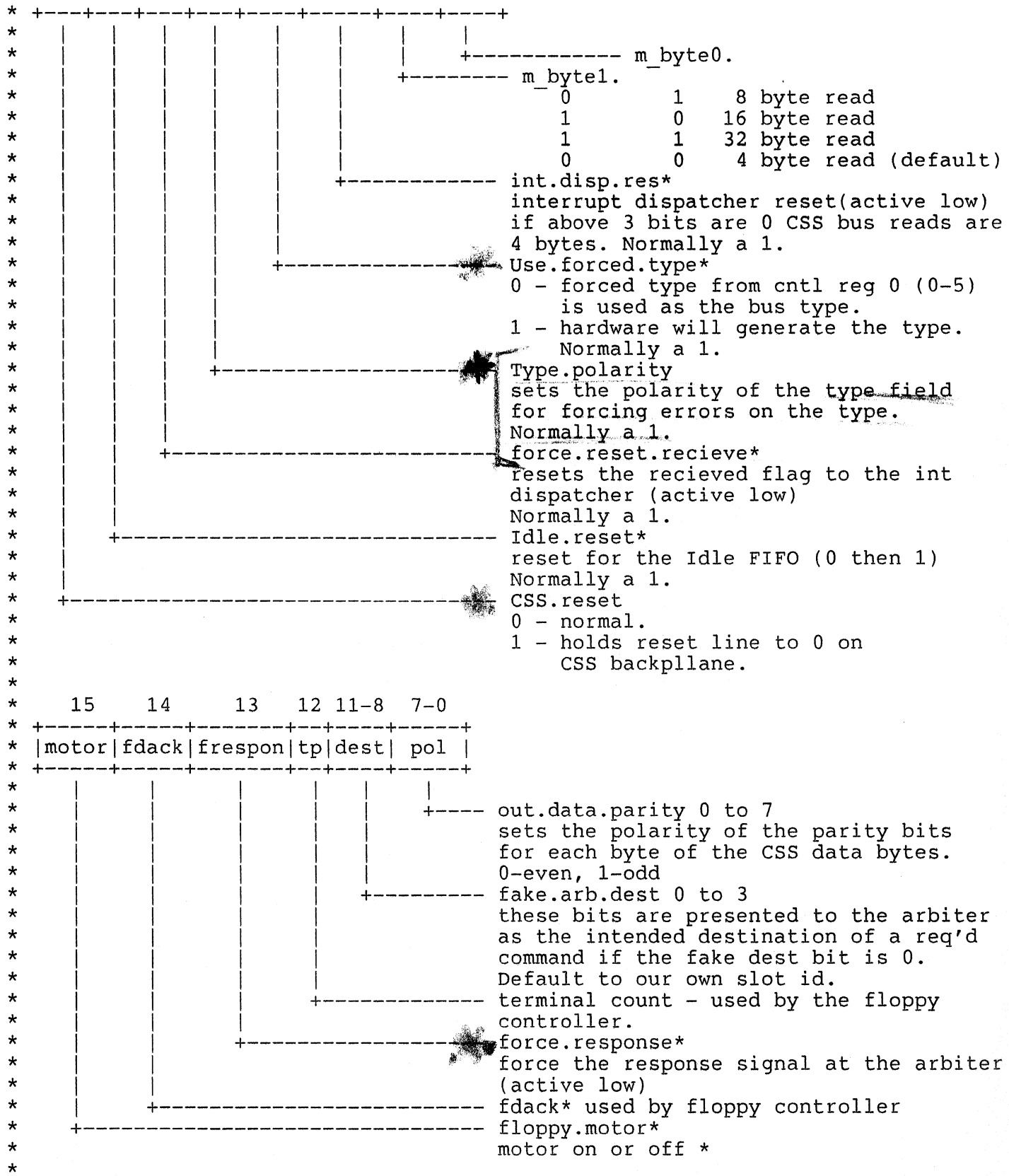
```
#define DISP_ACK_TOUT       5       /* timeout on ack response */
#define DISP_REQ_TOUT       6       /* timeout on request */
#define DISP_RCV_ERR        7
#define DISP_ERR_SH         20

/*****************************************************************************
 *
 *        Write Control Register 0 (write only)
 *
 *   31-29  28    27    26    25    24   23-20  19-16
 *  +-----+-----+-----+-----+-----+-----+------+-----+
 *  |     |inh7 |freez| dest |grant| ack |srcID*|srcID|
 *  +-----+-----+-----+-----+-----+-----+------+-----+
 *     |     |     |     |     |     |       |     |
 *     |     |     |     |     |     |       |     +---- out.src 0-3 source id
 *     |     |     |     |     |     |       |           used by SPM when it is
 *     |     |     |     |     |     |       |           issuing a command
 *     |     |     |     |     |     |       +---------- out.src 0 to 3*
 *     |     |     |     |     |     |                   complement of source id
 *     |     |     |     |     |     +------------------ force ack/nak error.
 *     |     |     |     |     |
 *     |     |     |     |     +------------------------ force grant error.
 *     |     |     |     |
 *     |     |     |     +---------------------------- force bad dest.
 *     |     |     |
 *     |     |     +-------------------------------- Freeze CPU bit:
 *     |     |                                       active 1 (Must have
 *     |     |                                       CSS Reset rdy 1)
 *     |     +---------------------------------------- Inhibit level 7
 *     |                                             CSS int's.
 *     |
 *     +-----------------------------------------
 *
 *
 *
 *     15-12  11-8  7-6  5-0
 *    +------+------+---+-----+
 *    |slotID|slotID|f10|Ftype|
 *    +------+------+---+-----+
 *       |      |     |     |
 *       |      |     |     +---- forced type
 *       |      |     |           (used if "use.forced.type" bit is active)
 *       |      |     +---------- forced address bits 00 & 01
 *       |      +---------------- slot id used by int dispatcher
 *       |                        (tells SPM where to place pointers for
 *       |                        non-dir ints 0-3)
 *       +------------------------ slot id used with CSS backplane
 *                                (to pretend to be a CPU)
 *
 *****************************************************************************/

#define WRCNTL0 ((unsigned *)(0x07000000))
#define WR0_FTYPE_MASK    0x0000003f
#define WR0_FADD01_MASK   0x000000c0
#define WR0_FADD01_SH     0x06
#define WR0_PRETEND_MASK  0x0000f000
```

```
#define WR0_PRETEND_SH     12
#define WR0_FRC_ACK        0x01000000
#define WR0_FRC_GRANT      0x02000000
#define WR0_FRC_DEST       0x04000000
#define WR0_FRC_FREE       0x08000000
#define WR0_FRC_INH7       0x10000000

/*****************************************************************************
 *
 *         Write Control Register 1 (write only)
 *
 *    31   30   29    28     27    26    25    24
 *  +---+---+----+----+-----+----+----+----+
 *  |ack|rdy|rrdy|fmod|fdest|diag|diag|dreq|
 *  +---+---+----+----+-----+----+----+----+
 *    |    |    |     |     |     |     |     |
 *    |    |    |     |     |     |     |     |---- cpu.disp.req*
 *    |    |    |     |     |     |     |           0 - indicates to int dispatcher that
 *    |    |    |     |     |     |     |           68020 wants to access its rams.
 *    |    |    |     |     |     |     |           1 - int dispatcher resumes normally.
 *    |    |    |     |     |     |     +---------- diag.frc.int.rec*
 *    |    |    |     |     |     |                 0 - decode a CSS cpu int request
 *    |    |    |     |     |     |                     level as a CSS command.
 *    |    |    |     |     |     |                 1 - normal mode.
 *    |    |    |     |     |     +---------------- diag.any.type*
 *    |    |    |     |     |                       0 - accept any CSS bus action as a
 *    |    |    |     |     |                           valid response to a SPM
 *    |    |    |     |     |                           initiated read request.
 *    |    |    |     |     |                       1 - normal mode.
 *    |    |    |     |     +---------------------- use.fake.dest*
 *    |    |    |     |                             0 - destination at arbiter is forced
 *    |    |    |     |                                 to be bits 8 to 11 of cntl reg 1
 *    |    |    |     |                             1 - normal - dest at arbiter is the
 *    |    |    |     |                                 same as the dest which will be
 *    |    |    |     |                                 used on the CSS bus with the
 *    |    |    |     |                                 requested CSS bus transaction.
 *    |    |    |     +---------------------------- force.modify*
 *    |    |    |                                   0 - forces a modify cycle on the CSS
 *    |    |    |                                       bus as long as the bit is 0
 *    |    |    |                                   1 - normal mode.
 *    |    |    +------------------------------------ cpu.res.ready*
 *    |    |                                         1 to 0 resets the ready count in the
 *    |    |                                           arbiter for the SPM.
 *    |    |                                           Normally a 1.
 *    |    +------------------------------------------ cpu.inc.ready*
 *    |                                              1 to 0 increments the ready count
 *    |                                                in the arbiter for the SPM.
 *    |                                                Normally a 1.
 *    +---------------------------------------------- frc.reset.ack*
 *                                                   use to reset ACK request during ACK
 *                                                   error interrupt service.
 *                                                   Normally a 1.
 *
 *    23   22   21   20   19    18    17   16
 *  +---+---+---+---+----+-----+----+----+
 *  |rst|rst|rst|pol|type|idrst| m1 | m0 |
```


m_byte0.
m_byte1.
0 1 8 byte read
1 0 16 byte read
1 1 32 byte read
0 0 4 byte read (default)
int.disp.res*
interrupt dispatcher reset(active low)
if above 3 bits are 0 CSS bus reads are
4 bytes. Normally a 1.
Use.forced.type*
0 - forced type from cntl reg 0 (0-5)
is used as the bus type.
1 - hardware will generate the type.
Normally a 1.
Type.polarity
sets the polarity of the type field
for forcing errors on the type.
Normally a 1.
force.reset.recieve*
resets the recieved flag to the int
dispatcher (active low)
Normally a 1.
Idle.reset*
reset for the Idle FIFO (0 then 1)
Normally a 1.
CSS.reset
0 - normal.
1 - holds reset line to 0 on
CSS backpllane.
15 14 13 12 11-8 7-0
motor fdack frespon tp dest pol
out.data.parity 0 to 7
sets the polarity of the parity bits
for each byte of the CSS data bytes.
0-even, 1-odd
fake.arb.dest 0 to 3
these bits are presented to the arbiter
as the intended destination of a req'd
command if the fake dest bit is 0.
Default to our own slot id.
terminal count - used by the floppy
controller.
force.response*
force the response signal at the arbiter
(active low)
fdack* used by floppy controller
floppy.motor*
motor on or off

```
 ***************************************************************************/

#define WRCNTL1 ((unsigned *)(0x07000020))
#define WR1_DIAG_ANY_TYPE    0x04000000

#define WR1_FRC_RST_ACK      0x80000000
#define WR1_IRDY             0x40000000
#define WR1_RRDY             0x20000000
#define WR1_FMOD             0x10000000
#define WR1_FDEST            0x08000000
#define WR1_DATYPE           0x04000000
#define WR1_FINTREC          0x02000000
#define WR1_DREQ             0x01000000
#define WR1_CSSRST           0x00800000
#define WR1_IDLERST          0x00400000
#define WR1_FRC_RST_RCV      0x00200000
#define WR1_RSTREC           0x00200000
#define WR1_TPOL             0x00100000
#define WR1_UFTYPE           0x00080000
#define WR1_ID_RST           0x00040000
#define WR1_CSSRD_MASK       0x00030000
#define WR1_32BYTE_RD        0x00030000
#define WR1_RD32             0x00030000
#define WR1_16BYTE_RD        0x00020000
#define WR1_RD16             0x00020000
#define WR1_8BYTE_RD         0x00010000
#define WR1_RD8              0x00010000
#define WR1_4BYTE_RD         0x00000000
#define WR1_MOTOR            0x00008000
#define WR1_FDACK            0x00004000
#define WR1_FRES             0x00002000
#define WR1_TP               0x00001000
#define WR1_DEST_SH          8
#define WR1_POL              0x000000ff

/***************************************************************************
 *
 *        Write Control Register 2 (write only)
 *
 *   31  30   29   28   27  26-25 24
 * +---+----+----+----+----+----+----+
 * |rst|led0|led1| grn| red|size|prom|
 * +---+----+----+----+----+----+----+
 *   |    |    |    |    |    |    |
 *   |    |    |    |    |    |    +------ prom.at.1
 *   |    |    |    |    |    |             0 - eprom at 0x00000000
 *   |    |    |    |    |    |             1 - eprom at 0x01000000
 *   |    |    |    |    |    +------------ promsiz.0 and promsiz.1
 *   |    |    |    |    |                  prom size 00 - 27256 (32k x 8)
 *   |    |    |    |    |                            01 - 27512 (64k x 8)
 *   |    |    |    |    |                            10 - 27010 (128k x 8)
 *   |    |    |    |    |                            11 - reserved
 *   |    |    |    |    |
 *   |    |    |    |    +----------------- red fault* led (top of board)
 *   |    |    |    +---------------------- green led* (second from top)
 *   |    |    +--------------------------- red led 1* (third from top)
```

```
 *    |    +------------------------------- red led 0* (fourth from top)
 *    +------------------------------------ bd.reset*
 *                                          board reset when 0, cpu must set to
 *                                          1 on power up
 *
 ****************************************************************************/

#define WRCNTL2 ((unsigned char *)(0x07000040))

#define WR2_BDRST 0x80
#define WR2_LED0 0x40
#define WR2_LED1 0x20
#define WR2_GRNLED 0x10
#define WR2_REDLED 0x08
#define WR2_LED01_SH 0x05

/****************************************************************************
 *
 *     Map Ram
 *
 *     When the SPM is performing a CSS access, add.31 = 1, the map ram is
 *     read. CPU address bits 28 to 30 select one of eight locations in the
 *     map ram. The top 4 bits of the map ram output give the destination
 *     (i.e. the slot id of the board being accessed) and the bottom four bits
 *     are mapped to address bits bus_data_44 to bus_data_47 for the CSS bus.
 *     The lower 28 bits of the 68020s address are directly mapped to address
 *     bits bus_data_40 to 43 and bus_data_50 to 77. Before accessing the CSS
 *     bus, the 68020 must initialize the map ram.
 *
 *    31  30   29   28   27   26   25   24
 *  +---+----+----+----+----+----+----+----+
 *  | destination slot |  CSS add 44 - 47  |
 *  +---+----+----+----+----+----+----+----+
 *
 ****************************************************************************/

#define MAPBASE (unsigned char *)0x08000000
#define MAP0    (unsigned char *)0x08000000
#define MAP1    (unsigned char *)0x18000000
#define MAP2    (unsigned char *)0x28000000
#define MAP3    (unsigned char *)0x38000000
#define MAP4    (unsigned char *)0x48000000
#define MAP5    (unsigned char *)0x58000000
#define MAP6    (unsigned char *)0x68000000
#define MAP7    (unsigned char *)0x78000000

/****************************************************************************
 *
 *    Interrupt Idle Queue
 *
 *    The queue is a 16 character fifo, where the programmed character is an
 *    "idle" cpu board.  This should cuase the next non-directed interrupt to
 *    be passed off to this cpu, rather than the next cpu in the main queue.
 *
 *    31  30   29   28   27   26   25   24    23 - 0
 *  +---+----+----+----+----+----+----+----+----------+
```

```
 * |  Not used        |     idle cpu      |  Not used |
 * +---+----+----+----+----+----+----+----+-----------+
 *
 ************************************************************************/

#define IDLEREG     (unsigned int *)0x0b000000
#define IDLESHFT    24

/************************************************************************
 *
 *      Interrupt Dispatcher Queue
 *
 *      The main queue is a 4k x 32 bit RAM which contains all the information
 *      attached to an interrupt as it is defined on the CSS bus. It is
 *      recommended that it is initialized with a known value for diagnostic
 *      reasons. Below is a map describing where interrupt data is placed in
 *      the Queue Ram.
 *
 *        loc                                interrupt type
 *        ---                                --------------
 *
 *      +------------+-------------------------------------------------+ ----
 *      | 0x0c000000 |                  non-directed level 0           | 128
 *      | 0x0c0001fc |                                                 | ints
 *      +------------+-------------------------------------------------+ ----
 *      | 0x0c000200 |                  reserved                       |
 *      | 0x0c0007fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c000800 |                  non-directed level 1           |
 *      | 0x0c0009fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c000a00 |                  reserved                       |
 *      | 0x0c000ffc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c001000 |                  non-directed level 2           |
 *      | 0x0c0011fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c001200 |                  reserved                       |
 *      | 0x0c0017fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c001800 |                  non-directed level 3           |
 *      | 0x0c0019fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c001a00 |                  reserved                       |
 *      | 0x0c001ffc |                                                 |
 *      +------------+-------------------------------------------------+ ----
 *      | 0x0c002000 |                  directed level 4, CPU 0        | 64
 *      | 0x0c0020fc |                                                 | ints
 *      +------------+-------------------------------------------------+ ----
 *      | 0x0c002100 |                  directed level 4, CPU 1        |
 *      | 0x0c0021fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c002200 |                  directed level 4, CPU 2        |
 *      | 0x0c0022fc |                                                 |
 *      +------------+-------------------------------------------------+
 *      | 0x0c002300 |                  directed level 4, CPU 3        |
```

```
*       | 0x0c0023fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002400 |                          directed level 4, CPU 4 |
*       | 0x0c0024fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002500 |                          directed level 4, CPU 5 |
*       | 0x0c0025fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002600 |                          directed level 4, CPU 6 |
*       | 0x0c0026fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002700 |                          directed level 4, CPU 7 |
*       | 0x0c0027fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002800 |                          directed level 4, CPU 8 |
*       | 0x0c0028fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002900 |                          directed level 4, CPU 9 |
*       | 0x0c0029fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002a00 |                          directed level 4, CPU A |
*       | 0x0c002afc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002b00 |                          directed level 4, CPU B |
*       | 0x0c002bfc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002c00 |                          directed level 4, CPU C |
*       | 0x0c002cfc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002d00 |                          directed level 4, CPU D |
*       | 0x0c002dfc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002e00 |                          directed level 4, CPU E |
*       | 0x0c002efc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c002f00 |                          directed level 4, CPU f |
*       | 0x0c002ffc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003000 |                          directed level 5, CPU 0 |
*       | 0x0c0030fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003100 |                          directed level 5, CPU 1 |
*       | 0x0c0031fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003200 |                          directed level 5, CPU 2 |
*       | 0x0c0032fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003300 |                          directed level 5, CPU 3 |
*       | 0x0c0033fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003400 |                          directed level 5, CPU 4 |
*       | 0x0c0034fc |                                                  |
*       +------------+--------------------------------------------------+
*       | 0x0c003500 |                          directed level 5, CPU 5 |
*       | 0x0c0035fc |                                                  |
*       +------------+--------------------------------------------------+
```

```
 *       | 0x0c003600 |                          directed level 5, CPU 6   |
 *       | 0x0c0036fc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003700 |                          directed level 5, CPU 7   |
 *       | 0x0c0037fc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003800 |                          directed level 5, CPU 8   |
 *       | 0x0c0038fc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003900 |                          directed level 5, CPU 9   |
 *       | 0x0c0039fc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003a00 |                          directed level 5, CPU A   |
 *       | 0x0c003afc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003b00 |                          directed level 5, CPU B   |
 *       | 0x0c003bfc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003c00 |                          directed level 5, CPU C   |
 *       | 0x0c003cfc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003d00 |                          directed level 5, CPU D   |
 *       | 0x0c003dfc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003e00 |                          directed level 5, CPU E   |
 *       | 0x0c003efc |                                                    |
 *       +------------+----------------------------------------------------+
 *       | 0x0c003f00 |                          directed level 5, CPU f   |
 *       | 0x0c003ffc |                                                    |
 *       +------------+----------------------------------------------------+
 *
 *
 ******************************************************************************/

#define QUEUE_RAM (unsigned *)(0x0c000000)
#define QUEUE_LEN      0x4000
#define QUEUE_LWORDS   0x1000
#define VECTOR_MASK    0xff000000
#define VECTOR_SH      24
#define Q_DEST_MASK    0x0000f000
#define Q_DEST_SH      12
#define Q_SRC_MASK     0x00000f00
#define Q_SRC_SH       8

/******************************************************************************
 *
 *       Interrupt Dispatcher Pointer
 *
 *       There are two pointer RAM's, each 256 x 7 bits, called the incoming
 *       pointer RAM and the outgoing pointer RAM. These pointer RAM's allow the
 *       main queue to function as multiple FIFOs. The incoming and outgoing
 *       pointer associated with a particular long word in the Pointer RAM will
 *       either point to a queue for directed level 4 or 5 interrupts for a CPU
 *       in slot x, or if a SPM is in slot x, the pointers are used to maintain
 *       interrupt info for non-directed level 0 to 3 interrupts.
 *
```

```
*          31  30 29 28 27 26 25 24 23  22 21 20 19 18 17 16  15 - 0
*         +---+--------------------+---+--------------------+------+
*         | x |  incoming pointer  | x |   outgoing pointer |   x  |
*         +---+--------------------+---+--------------------+------+
*
*
*              loc                          pointer type
*              ---                          ------------
*         +------------+-------------------------------------------+
*         | 0x0d000000 |              directed level 4, CPU 0      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000004 |              directed level 4, CPU 1      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000008 |              directed level 4, CPU 2      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d00000c |              directed level 4, CPU 3      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000010 |              directed level 4, CPU 4      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000014 |              directed level 4, CPU 5      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000018 |              directed level 4, CPU 6      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d00001c |              directed level 4, CPU 7      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000020 |              directed level 4, CPU 8      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000024 |              directed level 4, CPU 9      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000028 |              directed level 4, CPU A      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d00002c |              directed level 4, CPU B      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000030 |              directed level 4, CPU C      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000034 |              directed level 4, CPU D      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d000038 |              directed level 4, CPU E      |
*         |            |              or non-directed level 0      |
*         +------------+-------------------------------------------+
*         | 0x0d00003c |              directed level 4, CPU F      |
*         |            |              or non-directed level 0      |
```

0x0d000040	directed level 5, CPU 0 or non-directed level 1
0x0d000044	directed level 5, CPU 1 or non-directed level 1
0x0d000048	directed level 5, CPU 2 or non-directed level 1
0x0d00004c	directed level 5, CPU 3 or non-directed level 1
0x0d000050	directed level 5, CPU 4 or non-directed level 1
0x0d000054	directed level 5, CPU 5 or non-directed level 1
0x0d000058	directed level 5, CPU 6 or non-directed level 1
0x0d00005c	directed level 5, CPU 7 or non-directed level 1
0x0d000060	directed level 5, CPU 8 or non-directed level 1
0x0d000064	directed level 5, CPU 9 or non-directed level 1
0x0d000068	directed level 5, CPU A or non-directed level 1
0x0d00006c	directed level 5, CPU B or non-directed level 1
0x0d000070	directed level 5, CPU C or non-directed level 1
0x0d000074	directed level 5, CPU D or non-directed level 1
0x0d000078	directed level 5, CPU E or non-directed level 1
0x0d00007c	directed level 5, CPU F or non-directed level 1
0x0d000080	non-directed level 2 (if SPM in slot 0)
0x0d000084	non-directed level 2 (if SPM in slot 1)
0x0d000088	non-directed level 2 (if SPM in slot 2)
0x0d00008c	non-directed level 2 (if SPM in slot 3)

```
*       +------------+-------------------------------------------------+
*       | 0x0d000090 |          non-directed level 2 (if SPM in slot 4) |
*       +------------+-------------------------------------------------+
*       | 0x0d000094 |          non-directed level 2 (if SPM in slot 5) |
*       +------------+-------------------------------------------------+
*       | 0x0d000098 |          non-directed level 2 (if SPM in slot 6) |
*       +------------+-------------------------------------------------+
*       | 0x0d00009c |          non-directed level 2 (if SPM in slot 7) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000a0 |          non-directed level 2 (if SPM in slot 8) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000a4 |          non-directed level 2 (if SPM in slot 9) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000a8 |          non-directed level 2 (if SPM in slot A) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000ac |          non-directed level 2 (if SPM in slot B) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000b0 |          non-directed level 2 (if SPM in slot C) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000b4 |          non-directed level 2 (if SPM in slot D) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000b8 |          non-directed level 2 (if SPM in slot E) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000bc |          non-directed level 2 (if SPM in slot F) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000c0 |          non-directed level 3 (if SPM in slot 0) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000c4 |          non-directed level 3 (if SPM in slot 1) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000c8 |          non-directed level 3 (if SPM in slot 2) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000cc |          non-directed level 3 (if SPM in slot 3) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000d0 |          non-directed level 3 (if SPM in slot 4) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000d4 |          non-directed level 3 (if SPM in slot 5) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000d8 |          non-directed level 3 (if SPM in slot 6) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000dc |          non-directed level 3 (if SPM in slot 7) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000e0 |          non-directed level 3 (if SPM in slot 8) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000e4 |          non-directed level 3 (if SPM in slot 9) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000e8 |          non-directed level 3 (if SPM in slot A) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000ec |          non-directed level 3 (if SPM in slot B) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000f0 |          non-directed level 3 (if SPM in slot C) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000f4 |          non-directed level 3 (if SPM in slot D) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000f8 |          non-directed level 3 (if SPM in slot E) |
*       +------------+-------------------------------------------------+
*       | 0x0d0000fc |          non-directed level 3 (if SPM in slot F) |
```

```
 *        +-----------+-------------------------------------------------+
 *
 *
 ***************************************************************************/

#define POINTER_RAM (unsigned *)(0x0d000000)
#define POINTER_LEN             0x100
#define POINTER_LWORDS          0x40
#define POINTER_MASK            0x7f7f0000
#define OUT_PNTR_SH             16
#define IN_PNTR_SH              24

/***************************************************************************
 *
 *      Interrupt Dispatcher Miscellaneous RAMs
 *
 *      The miscellaneous RAMs consist of the counter RAM, the service pending
 *      RAM, and the CPU configuration RAM. They are all simultaneously address-
 *      ed as a 32 bit port, with different data bits going to the different
 *      RAMs as follows:
 *
 *         31 - 28 27 - 24  23 - 16       15       14 - 10    9 - 0
 *        +-------+-------+---------+------------+---------+---------+
 *        |   x   | config|    x    | serv. pend |     x   | counter |
 *        +-------+-------+---------+------------+---------+---------+
 *                  16                 128                   8
 *                 deep               deep                 deep
 *
 *      The counter must be initialized to all 1s, the service pending must be
 *      initialized to all 0s, and the configuration RAM must be initialized
 *      with all slot ids of CPUs present in the CSS backplane which are avail-
 *      able for recieving interrupts as follows:
 *
 *                address                       data
 *
 *                CPU slot id n               CPU slot id 1
 *                CPU slot id 1               CPU slot id 2
 *                      .                           .
 *                      .                           .
 *                      .                           .
 *                CPU slot id n-1               CPU slot id n
 *                    n+1                       CPU slot id 1
 *                      .                           .
 *                      .                           .
 *                      .                           .
 *                     16                       CPU slot id 1
 *
 *        Counter
 *        -------
 *        note:       negative true logic is used to store the count
 *                  (0x3ff corresponds to 0)
 *
 *            Address           Bits 9 - 0
 *        +---------------+----------------------+
 *        | 0x0e000000    |     count level 0    |
 *        +---------------+----------------------+
```

```
*         | 0x0e000004    |     count level 1     |
*         +---------------+-----------------------+
*         | 0x0e000008    |     count level 2     |
*         +---------------+-----------------------+
*         | 0x0e00000c    |     count level 3     |
*         +---------------+-----------------------+
*         | 0x0e000010    |     count level 4     |
*         +---------------+-----------------------+
*         | 0x0e000014    |     count level 5     |
*         +---------------+-----------------------+
*         | 0x0e000018    |       reserved        |
*         +---------------+-----------------------+
*         | 0x0e00001c    |       reserved        |
*         +---------------+-----------------------+
*
*         Service Pending Ram
*         -------------------
*
*            Address                 Bit 15
*         +---------------+-----------------------+
*         | 0x0e000000    |    CPU 0, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000004    |    CPU 1, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000008    |    CPU 2, level 0     |
*         +---------------+-----------------------+
*         | 0x0e00000c    |    CPU 3, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000010    |    CPU 4, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000014    |    CPU 5, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000018    |    CPU 6, level 0     |
*         +---------------+-----------------------+
*         | 0x0e00001c    |    CPU 7, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000020    |    CPU 8, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000024    |    CPU 9, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000028    |    CPU A, level 0     |
*         +---------------+-----------------------+
*         | 0x0e00002c    |    CPU B, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000030    |    CPU C, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000034    |    CPU D, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000038    |    CPU E, level 0     |
*         +---------------+-----------------------+
*         | 0x0e00003c    |    CPU F, level 0     |
*         +---------------+-----------------------+
*         | 0x0e000040    |    CPU 0, level 1     |
*         +---------------+-----------------------+
*         | 0x0e000044    |    CPU 1, level 1     |
*         +---------------+-----------------------+
```

```
*       | 0x0e000048    |    CPU 2, level 1     |
*       +---------------+-----------------------+
*       | 0x0e00004c    |    CPU 3, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000050    |    CPU 4, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000054    |    CPU 5, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000058    |    CPU 6, level 1     |
*       +---------------+-----------------------+
*       | 0x0e00005c    |    CPU 7, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000060    |    CPU 8, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000064    |    CPU 9, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000068    |    CPU A, level 1     |
*       +---------------+-----------------------+
*       | 0x0e00006c    |    CPU B, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000070    |    CPU C, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000074    |    CPU D, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000078    |    CPU E, level 1     |
*       +---------------+-----------------------+
*       | 0x0e00007c    |    CPU F, level 1     |
*       +---------------+-----------------------+
*       | 0x0e000080    |    CPU 0, level 2     |
*       +---------------+-----------------------+
*       | 0x0e000084    |    CPU 1, level 2     |
*       +---------------+-----------------------+
*       | 0x0e000088    |    CPU 2, level 2     |
*       +---------------+-----------------------+
*       | 0x0e00008c    |    CPU 3, level 2     |
*       +---------------+-----------------------+
*       | 0x0e000090    |    CPU 4, level 2     |
*       +---------------+-----------------------+
*       | 0x0e000094    |    CPU 5, level 2     |
*       +---------------+-----------------------+
*       | 0x0e000098    |    CPU 6, level 2     |
*       +---------------+-----------------------+
*       | 0x0e00009c    |    CPU 7, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000a0    |    CPU 8, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000a4    |    CPU 9, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000a8    |    CPU A, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000ac    |    CPU B, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000b0    |    CPU C, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000b4    |    CPU D, level 2     |
*       +---------------+-----------------------+
```

```
*       | 0x0e0000b8    |    CPU E, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000bc    |    CPU F, level 2     |
*       +---------------+-----------------------+
*       | 0x0e0000c0    |    CPU 0, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000c4    |    CPU 1, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000c8    |    CPU 2, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000cc    |    CPU 3, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000d0    |    CPU 4, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000d4    |    CPU 5, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000d8    |    CPU 6, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000dc    |    CPU 7, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000e0    |    CPU 8, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000e4    |    CPU 9, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000e8    |    CPU A, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000ec    |    CPU B, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000f0    |    CPU C, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000f4    |    CPU D, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000f8    |    CPU E, level 3     |
*       +---------------+-----------------------+
*       | 0x0e0000fc    |    CPU F, level 3     |
*       +---------------+-----------------------+
*       | 0x0e000100    |    CPU 0, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000104    |    CPU 1, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000108    |    CPU 2, level 4     |
*       +---------------+-----------------------+
*       | 0x0e00010c    |    CPU 3, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000110    |    CPU 4, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000114    |    CPU 5, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000118    |    CPU 6, level 4     |
*       +---------------+-----------------------+
*       | 0x0e00011c    |    CPU 7, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000120    |    CPU 8, level 4     |
*       +---------------+-----------------------+
*       | 0x0e000124    |    CPU 9, level 4     |
*       +---------------+-----------------------+
```

```
 *         | 0x0e000128    |    CPU A, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e00012c    |    CPU B, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e000130    |    CPU C, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e000134    |    CPU D, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e000138    |    CPU E, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e00013c    |    CPU F, level 4     |
 *         +---------------+-----------------------+
 *         | 0x0e000140    |    CPU 0, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000144    |    CPU 1, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000148    |    CPU 2, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e00014c    |    CPU 3, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000150    |    CPU 4, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000154    |    CPU 5, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000158    |    CPU 6, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e00015c    |    CPU 7, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000160    |    CPU 8, level 5     |
 *         +---------------+-----------------------+
 *         | 0x0e000164        |        CPU 9, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e000168        |        CPU A, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e00016c        |        CPU B, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e000170        |        CPU C, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e000174        |        CPU D, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e000178        |        CPU E, level 5         |
 *         +---------------+-----------------------+
 *         | 0x0e00017c        |        CPU F, level 5         |
 *         +---------------+-----------------------+
 *
 *
 ************************************************************************/

#define MISC_RAM (unsigned *)(0x0e000000)
#define MISC_LEN              0x200
#define MISC_LWORDS           0x80
#define MISC_MASK             0x0f0083ff
#define CNTR_MASK             0x000003ff
#define CNTR_LWORDS           8
#define CNTR_ZERO_BIT         0x00000400
#define SERV_MASK             0x00008000
```

```
#define SERV_LWORDS          0x60
#define CONFIG_LWORDS        16
#define CONFIG_MASK          0x0f000000
#define CONFIG_SH            24

#define PIPEDATA (unsigned *)(0x80000018)
#define PIPEADDR (unsigned *)(0x8000001c)

#define DATARESP 4


/*****************************************************************************
 *
 *        Translation table for CSS data bits to local address and data bits.
 *
 *          Byte 0   Byte 1   Byte 2  CSS BUS DATA  Byte 3   Byte 4   Byte 5   Byte 6   Byte 7
 *        +------+------+------+------+------+------+------+------+
 *        |00  07|10  17|20  27|30  37|40  47|50  57|60  67|70  77|
 *        +------+------+------+------+------+------+------+------+
 *        |24  31|16  23|8   15|0    7|24  31|16  23|8   15|0    7|
 *        +------+------+------+------+------+------+------+------+
 *        |<-------- IN DATA -------->|<-------- IN ADDR -------->|
 *
 *                        IN DATA
 *          31 - 24   23 22-18 17 16   15 - 12    11 - 8   7     6 - 0
 *        +----------+-+------+-----+----------+----------+-+-------------+
 *        | Vector # |D|   x  |level|Dest. Slot| Src. Slot|I|I/O Src. Slot|
 *        +----------+-+------+-----+----------+----------+-+-------------+
 *
 *        Vector #
 *        --------
 *        read by the 68020 during the interrupt acknowledge cycle.
 *
 *        Directed Bit
 *        ------------
 *        directed interrupt if a 1, the interrupt must be sent to the
 *        specified destination slot. If a 0, the interrupt may be sent to
 *        any computational module accepting interrupts.
 *
 *        Int Priority
 *        ------------
 *        determines the priority of the interrupt, which increases with value.
 *
 *        Destination Slot
 *        ----------------
 *        If the directed bit is on, the interrupt is sent to the slot specified
 *        by this value. If a non-directed interrupt, this field is ignored.
 *
 *        Source Slot
 *        -----------
 *        The interrupt request came from the slot specified by this value.
 *
 *        I/O Bus Interrupt
 *        -----------------
 *        If a 1, the interrupt came from an i/o bus attached to the S-bus
 *        module. The I/O bus slot requesting the interrupt is specified by
```

```
 *        "I/O Bus Src Slot". If this bit is 0, the interrupt request came from
 *        the S-bus module.
 *
 *        I/O Bus Source Slot
 *        -------------------
 *        If the I/O Bus Interrupt bit is set, the interrupt came from the I/O
 *        bus slot specified by this value. If the I/O bus Interrupt bit is 0,
 *        this field is ignored(set to zero).
 *
 *
 *        D bit   Priority Level              Use
 *        -----   --------------              ---
 *          1      0x07 - 0x04        High priority directed interrupts;
 *                                    mapped into 68020 interrupt level 6.
 *          1      0x03 - 0x00        Low priority directed interrupts;
 *                                    mapped into 68020 interrupt level 5.
 *          0      0x0f - 0x0c        Highest priority i/o interrupts;
 *                                    mapped into 68020 interrupt level 4.
 *          0      0x0b - 0x08        High priority i/o interrupts;
 *                                    mapped into 68020 interrupt level 3.
 *          0      0x07 - 0x04        Low priority i/o interrupts;
 *                                    mapped into 68020 interrupt level 2.
 *          0      0x03 - 0x00        Lowest priority i/o interrupts;
 *                                    mapped into 68020 interrupt level 1.
 *
 *
 ******************************************************************************/
```