```
/*
*  Copyright 2004 by Texas Instruments Incorporated.
*  All rights reserved. Property of Texas Instruments Incorporated.
*  Restricted rights to use, duplicate or disclose this code are
*  granted through contract.
*
*/
/*****************************************************************************
******
 *
 * Filename : chk_mmc.c
 *
 * Version  : 1.0
 *
 * Author   : Raja Vydhynathan.
 *
 * Date     : Friday the 13th, December,2002
 *
 * Description :
 *
 * Version 1.0 : This source file provides an API called chk_mmc() . It checks the MMC
 *               for proper operation, with the data server. It does a integrity check
of
 *               the Master Boot Record (MBR) and  Boot Record ( BR), using the values
 *               from the MMC CSD register as reference. The format of the MMC
 *               ( Floppy /Hard disk) is based on the MMC CSD register.
 *
 *               The following fields are not checked by this API, since it does not
 *               affect data server operation.
 *
 *               Master Boot Record:
 *               1.MBR code
 *               2.MMC ID
 *               3.partition 1 -> start Cylinder,Head,Sector ( Cylinder Head Sector)
 *               4.partition 1 -> End   Cylinder,Sector.
 *               5.partition 1 -> Boot descriptor.
 *
 *               Boot Record:
 *               1. OEM name.
 *               2. sectors per track.
 *               3. number of heads.
 *               4. number of hidden sectors.
 *               5. drive number
 *               6. reserved
 *               7. volume serial number.
 *               8. volume label.
 *               9. optional partition boot code.
 *
 * Version 1.1 : Susmit, 04-Mar-2003
 *      (a) Added a new unsigned int *disk_type parameter to chk_mmc().
 *      (b) Modifying MBR & BR checks to validate MBR & BR as much as possible.
 *          Trying to conform to FAT specs as much as possible.
 *      (c) Removed a few checks as they are not mandated by the FAT spec.
 *
 * Version 1.2 : Pramod R, 24-Apr-2003
 *      (a) Added a check for second possible jump instruction 0xE9 in Boot Record on
 *          first byte position.
 *
 * Version 1.3 : Pratap, 07-Sep-2009
 *      (a) Modified getMMCSize() to calculate disk size from CSD structure ver2.0
 *
 *
*****************************************************************************
***/


 #include "chk_mmc.h"
```

```
 /*****************************************************************************
******\
  *
  * Function : MMC_ERR_int16  chk_mmc(AtaState *pAtaDrive);
  *
  * Parameters : AtaState *pAtaDrive -> Pointer to initialised AtaState variable. The
MMC
  *              card should have been reset successfully before this API can be
called.
  *
  * Return Values :
  *
  * MMC_ERR_NONE                  -> No Problem has been detected in the MMC.
  *
  * MMC_ERR_BAD_CSD_FILE_FORMAT   -> The file format indicated by CSD register
  *                                  is not floppy and not hard disk types. This card
is not
  *                                  supported by data server.
  *
  * MMC_ERR_CARD_NOT_READABLE     -> Unable to read card using ATA_readSector()
  *
  * MMC_ERR_MBR_BAD_SIGNATURE     -> The MBR is not having the signature 0x55AA.
  *
  * MMC_ERR_MBR_BAD_FS_DESCRIPTOR -> The Partition 1->file system descriptor must NOT
be zero.
  *                                  (Probably the card is not formatted.)
  *
  * MMC_ERR_MBR_BAD_PART_END_HEAD -> The Partition 1->Partition end head must NOT be
zero.
  *                                  (Windows cannot read the card if it is zero.)
  *
  * MMC_ERR_MBR_BAD_NUM_PART_SECTORS -> The Partition 1-> no of partition sectors is
larger
  *                                     than the total number of sectors reported by
CSD
  *                                     register
  *
  * MMC_ERR_MBR_BAD_PARTITIONS_234   -> All the partition entries in partitions 2,3,4
must be
  *                                     zero.Atleast one of them is non zero.
  *
  * MMC_ERR_BR_BAD_JMP_OPCODE        -> The first byte of BR was not 0xEB OR
  *                                     the third byte was not 0x90
  *
  * MMC_ERR_BR_BAD_BYTES_PER_SECTOR    -> The BR bytes per sector was not 512.
  *
  * MMC_ERR_BR_BAD_SECTORS_PER_CLUSTER -> The sectors per cluster is not a power of 2.
  *
  * MMC_ERR_BR_BAD_RESERVED_SECTORS    -> Boot sector + reserved sector is not pointing
to FAT table. (0x fff8 ffff)
  *
  * MMC_ERR_BR_BAD_NUM_OF_FATS         -> The number of FATs is not 2.
  *
  * MMC_ERR_BR_BAD_NUM_OF_ROOT_ENTRIES -> The number of root directory entries is not
512.
  *
  * MMC_ERR_BR_BAD_NUM_PART_SECTORS    -> * For boot_record_type = HD_BOOT_RECORD
  *                                       The number of partition sectors does not
  *                                       match with MBR (ref_num_of_sectors)
  *                                     * For boot_record_type = FD_BOOT_RECORD
  *                                       The number of partition sectors is greater
  *                                       than sectors in disk from CSD
(ref_num_of_sectors).
  *                                  Note: If the function parameter
ref_num_of_sectors > 0xFFFF
  *                                       the extended number of sectors in partition
```

```
is used
  *                                          in place of number of partition sectors.
  *
  *
  * MMC_ERR_BR_BAD_MEDIA_DESCRIPTOR    -> The media descriptor was not 0xF8. Win2k sets
this
  *                                          field as 0xF8 for both hard disk type and
  *                                          floppy type formats.
  *
  * MMC_ERR_BR_BAD_SECTORS_PER_FAT     -> The value was not between 1 and 256.
  *
  * MMC_ERR_BR_BAD_EXTENDED_BOOT_SIGNATURE -> The extended boot signature is not 0x29.
  *
  * MMC_ERR_BR_BAD_FILE_SYS_TYPE          -> The filesystem type is not " FAT16"
  *
  * MMC_ERR_BR_BAD_SIGNATURE              -> The boot record signature was not 0x55AA
  *
  * MMC_ERR_MBR_DSKSIZE_MISMATCH          -> Number of sectors in partition + boot
sector
  *                                          is not equal to number of sectors given
by CSD register
  *
  */

 //#pragma DATA_SECTION ( master_boot_record, "master_boot_record")
 MBR_struct master_boot_record; /* 512 byte buffer for checking master boot record */

 //#pragma DATA_SECTION ( boot_record, "boot_record")
 BR_struct  boot_record;        /* 512 byte buffer for checking boot record */

/* NOTE: This function will returns total number of sectors in the disk,
   Not the actual size of the disk  */
AtaUint32 getMMCSize(AtaState *pAtaDrive)
{
    AtaUint16    csd_data[8];
    AtaUint32    c_size = 0;
    AtaUint16    c_size_mult = 0;
    AtaUint16    read_bl_len = 0;
    AtaUint32    tempsize = 0;
    AtaUint32    totalSectors = 0;
    //unsigned int i = 0;
    AtaUint16    tempdiv = 512;
    Uint16       csdVersion = 0;
    //Uint16       sectorSize = 0;

    /* Read the Card Specific Data(CSD) */
    MMC_read_CSD(pAtaDrive->pAtaMediaState, (unsigned int *)csd_data);

    /* Get the CSD card structure version; Size calculataion will be different
      for different versions of CSD structures */
    csdVersion = (csd_data[7] >> 14) & 0x3;

    if(csdVersion == 0)  /* CSD structure version is 1.x */
    {
        read_bl_len = csd_data[5]& 0x0F;

        c_size = (AtaUint32)((csd_data[3] & 0xC000) >> 14);
        c_size |= (AtaUint32)((csd_data[4] & 0x3FF) << 2);
        c_size_mult = (AtaUint16)((csd_data[2] & 0x8000) >> 15);
        c_size_mult |= (AtaUint16)((csd_data[3] & 0x0003) << 1);

        tempsize = (AtaUint32)(c_size+1);

// mwei total capacity computation
#if 0
        for(i = 0; i < (c_size_mult + 2); i++)
        {
            tempsize *= 2;
```

```
        }

        for(i = 0; i < read_bl_len; i++)
        {
            tempdiv /= 2;
        }

        totalSectors = (AtaUint32)(tempsize*tempdiv);
#else
        tempsize <<= (c_size_mult + 2);
        tempdiv = 1 << (read_bl_len-9);
        totalSectors = tempsize*tempdiv;
#endif
    }
    else  /* CSD structure version is 2.x */
    {
        read_bl_len = csd_data[5] & 0x000F;

        c_size = (AtaUint32)(csd_data[3] & 0xFFFF);
        c_size |= ((AtaUint32)(csd_data[4] & 0x3F) << 16);

        //sectorSize = 1 << read_bl_len;

        /* Calculate the size of the disk */
        tempsize = (AtaUint32)(c_size+1)*512;

        /* For CSD ver2.0 size will be in KBytes, multiply
          with 1024 to convert to bytes */
// mwei CSD ver2.0 size is in KB not byte
#if 0
        tempsize = tempsize*1024;

        totalSectors = (AtaUint32)(tempsize/sectorSize);
#else
        // 1KB has two sectors (512 Byte)
        tempdiv = 2;
        totalSectors = (AtaUint32)(tempsize*tempdiv);
#endif
    }

    /* Return the valus of Number of sectors */
    return(totalSectors);
}

 /* <susmit : Added a new parameter, unsigned int *disk_type : 04-Mar-2003> */
 MMC_ERR_int16  chk_mmc(AtaState *pAtaDrive, unsigned int *disk_type)
 {
    AtaError      ata_error = ATA_ERROR_NONE;
    AtaUint32     disk_sectors_from_csd = 0xbeefbeef ;
    unsigned long sectors_in_partition_from_MBR = 0xbeefbeef;
    unsigned long boot_record_sector = 0xbeefbeef;
    unsigned int  ii = 0;
    /* <susmit : Variables to hold the error value and boot record type : 04-Mar-2003>
*/
    MMC_ERR_int16 mmcerror = MMC_ERR_NONE;
    unsigned int boot_record_type = HD_BOOT_RECORD;

#if 0 /* For the cards with csd ver2.0 it is not possible to determine the file system
format */
    unsigned int  mmc_file_system_format = 0xbeef;

    /* Get the file format information from MMC CSD register */
    mmc_file_system_format = MMC_file_system_format(pAtaDrive->pAtaMediaState);

    /* <susmit : If the file format is Universal File Format or Unknown, return error :
04-Mar-2003> */
    if((mmc_file_system_format!=0)&&(mmc_file_system_format!=1))
        return MMC_ERR_BAD_CSD_FILE_FORMAT;
```

```
#endif

    /* Get total number of sectors on MMC from CSD register  */
    disk_sectors_from_csd = getMMCSize(pAtaDrive);

    /* Try to read Master Boot Record from sector */
    ata_error = ATA_readSector(0, pAtaDrive, (AtaUint16*)&master_boot_record, 1);

    /* ata_error should be ATA_ERROR_NONE unless the card is damaged and unreadable */
    if(ata_error != ATA_ERROR_NONE)
        return MMC_ERR_CARD_NOT_READABLE;

    /* Check for all partition entries of partitions 2,3,4 to be zero */
    for(ii = 0; ii < 3*sizeof(PARTITION_TABLE); ii++){

        if( *((int*)(&master_boot_record.partition_two) + ii) != 0)
            /*return MMC_ERR_MBR_BAD_PARTITIONS_234;*/
            mmcerror = MMC_ERR_MBR_BAD_PARTITIONS_234;   /* <susmit : Set the error :
04-Mar-2003> */
    }

    /* Verify number of sectors in partition as reported by MBR is less than
       the total number of sectors reported by CSD register */

    sectors_in_partition_from_MBR =
    ((unsigned long)(master_boot_record.partition_one.byte1_no_of_sectors_in_partition)
|
     (unsigned long)
(master_boot_record.partition_one.byte2_no_of_sectors_in_partition)<<8|
     (unsigned long)
(master_boot_record.partition_one.byte3_no_of_sectors_in_partition)<<16 |
     (unsigned long)
(master_boot_record.partition_one.byte4_no_of_sectors_in_partition)<<24 );

    if(sectors_in_partition_from_MBR > disk_sectors_from_csd)
        mmcerror |= MMC_ERR_MBR_BAD_NUM_PART_SECTORS;   /* <susmit : Set the error : 04
-Mar-2003> */


    /* Calculate the sector number of the boot record */
    boot_record_sector =
        ((unsigned long)master_boot_record.partition_one.byte1_first_sector_position) |
        ((unsigned
 long)master_boot_record.partition_one.byte2_first_sector_position<<8) |
        ((unsigned
 long)master_boot_record.partition_one.byte3_first_sector_position<<16) |
        ((unsigned
 long)master_boot_record.partition_one.byte4_first_sector_position<<24);

    /* <susmit : The number of sectors in partition plus the number of reserved sectors
must be equal
     *  to the number of sectors present in the disk as indicated by the CSD register :
04-Mar-2003> */
    if((sectors_in_partition_from_MBR+boot_record_sector)!=disk_sectors_from_csd)
        mmcerror |= MMC_ERR_MBR_DSKSIZE_MISMATCH;

    /* <susmit : If no errors upto this, then this might be a disk with both MBR & BR.
     *  Otherwise set boot sector as zero & check : 04-Mar-2003> */
    if(mmcerror) {
        boot_record_sector = 0;
        boot_record_type = FD_BOOT_RECORD;
        sectors_in_partition_from_MBR = disk_sectors_from_csd;
        *disk_type = 1;   /* floppy-like file format (without partition table) */
    }

    /* Now check Boot Record  */
    return Check_boot_record(boot_record_sector,
                             sectors_in_partition_from_MBR,
```

```
                                boot_record_type,  /* <susmit : Can be HD or FD boot
record : 04-Mar-2003> */
                                pAtaDrive,
                                &boot_record);

 }


/*************************************************************************************
******
 *
 * Function :  MMC_ERR_int16  Check_boot_record(unsigned long boot_record_sector,
 *                                               unsigned long ref_num_of_sectors,
 *                                               unsigned int  boot_record_type,
 *                                               AtaState  *pAtaDrive,
 *                                               BR_struct *pBootRecord
 *
 * Parameters:
 *
 * boot_record_sector -> The sector number to fetch the BR from.
 * ref_num_of_sectors -> The reference number of sectors on disk to validate BR
against.
 * boot_record_type   -> Indicates whether this is floppy type or hard disk type boot
record.
 * pAtaDrive          -> Pointer to initialised AtaState structure.
 * pBootRecord        -> 512 byte buffer to read boot record.
 *
 *
 * Return Values:
 *
 * MMC_ERR_CARD_NOT_READABLE     -> Unable to read card using ATA_readSector()
 * MMC_ERR_BR_BAD_JMP_OPCODE     -> The first byte of BR was not 0xEB OR
 *                                  the third byte was not 0x90
 * MMC_ERR_BR_BAD_BYTES_PER_SECTOR    -> The BR bytes per sector was not 512.
 * MMC_ERR_BR_BAD_SECTORS_PER_CLUSTER -> The sectors per cluster is not a power of 2.
 * MMC_ERR_BR_BAD_RESERVED_SECTORS    -> Boot sector + reserved sector is not pointing
to FAT table. (0x fff8 ffff)
 * MMC_ERR_BR_BAD_NUM_OF_FATS         -> The number of FATs is not 2.
 * MMC_ERR_BR_BAD_NUM_OF_ROOT_ENTRIES -> The number of root directory entries is not
512.
 * MMC_ERR_BR_BAD_NUM_PART_SECTORS    -> * For boot_record_type = HD_BOOT_RECORD
 *                                       The number of partition sectors does not
 *                                       match with MBR (ref_num_of_sectors)
 *                                    * For boot_record_type = FD_BOOT_RECORD
 *                                       The number of partition sectors is greater
 *                                       than sectors in disk from CSD
(ref_num_of_sectors).
 *                                  Note: If the function parameter ref_num_of_sectors
> 0xFFFF
 *                                       the extended number of sectors in partition
is used
 *                                       in place of number of partition sectors.
 *
 *
 * MMC_ERR_BR_BAD_MEDIA_DESCRIPTOR    -> The media descriptor was not 0xF8. Win2k sets
this
 *                                     field as 0xF8 for both hard disk type and
 *                                     floppy type formats.
 * MMC_ERR_BR_BAD_SECTORS_PER_FAT     -> The value was not between 1 and 256.
 *
 * MMC_ERR_BR_BAD_EXTENDED_BOOT_SIGNATURE -> The extended boot signature is not 0x29.
 *
 * MMC_ERR_BR_BAD_FILE_SYS_TYPE           -> The filesystem type is not " FAT16"
 *
 * MMC_ERR_BR_BAD_SIGNATURE               -> The boot record signature was not 0x55AA
 */
```

```
 MMC_ERR_int16  Check_boot_record(unsigned long boot_record_sector,
                                                 unsigned long ref_num_of_sectors,
                                                 unsigned int  boot_record_type,
                                                 AtaState  *pAtaDrive,
                                                 BR_struct *pBootRecord
                                                 )
 {

    AtaError ata_error = ATA_ERROR_NONE;
    unsigned long l_reserved_sectors = 0xbeefbeef;
    unsigned long  l_fat_signature = 0xbeef;
    unsigned long l_num_partition_sectors = 0xbeefbeef;
    unsigned int  l_sectors_per_fat = 0xbeef;

    /* Read Boot Record from sector number = boot_record_sector  */
    ata_error = ATA_readSector(boot_record_sector, pAtaDrive, (AtaUint16*)pBootRecord,
1);

    /* ata_error should be ATA_ERROR_NONE unless the card is damaged and unreadable */
    if(ata_error != ATA_ERROR_NONE)
        return MMC_ERR_CARD_NOT_READABLE;

    /* Check first and third byte of boot record for valid opcodes */
    /* <pramod : Added check for second possible jump instruction 0xE9 : 24-Apr-2003>
*/
    if(((pBootRecord->short_jump_instr_byte_1 != 0xEB) &&
        (pBootRecord->short_jump_instr_byte_1 != 0xE9)) ||
       (pBootRecord->short_jump_instr_byte_3 != 0x90))
        return MMC_ERR_BR_BAD_JMP_OPCODE;

    /* Verify that bytes per sector to be 512 */
    if(((pBootRecord->UB_bytes_per_sector<<8) | (pBootRecord->LB_bytes_per_sector)) !=
512)
        return MMC_ERR_BR_BAD_BYTES_PER_SECTOR;

    /* Verify that sectors per cluster is a power of 2 */
    if( ((pBootRecord->sectors_per_cluster%2) != 0) &&
        (pBootRecord->sectors_per_cluster != 1) &&
        (pBootRecord->sectors_per_cluster>64) )   /* <susmit : Cluster size should not
exceed 32Kbytes : 04-Mar-2003> */
        return MMC_ERR_BR_BAD_SECTORS_PER_CLUSTER;

    /* Verify that data at boot sector + reserved sector is 0xFFF8 FFFF */
    l_reserved_sectors = (pBootRecord->UB_reserved_sectors<<8)|
(pBootRecord->LB_reserved_sectors);

    /* <susmit : Commented off this code for now but should be enabled : 04-Mar-2003>
*/
    /* MS FAT Specs say that this field should never be anything other than 1 for
FAT12/16. But
     * a particular customer's images puts some other values in here. Need to discuss
these */
    /*  if(l_reserved_sectors!=1) return MMC_ERR_BR_BAD_RESERVED_SECTORS;*/

    /* <susmit : _AtaReadDoubleWord() should not be called directly, hence replaced the
call with ATA_readSector() : 04-Mar-2003> */
    /* l_fat_signature = _AtaReadDoubleWord(boot_record_sector + l_reserved_sectors ,
pAtaDrive, 0); */
    ATA_readSector(boot_record_sector + l_reserved_sectors , pAtaDrive,
pAtaDrive->_AtaWriteBuffer, 0);
    l_fat_signature = pAtaDrive->_AtaWriteBuffer[0]|((unsigned long)
(pAtaDrive->_AtaWriteBuffer[1])<<16);

    /* The FAT signature must be 0xFFFFFFF8 . But if the volume is not dismounted
       correctly , windows sets the dirty bit to zero which makes the
       value appear as 0x7FFFFFF8 */
    if((l_fat_signature != 0xFFFFFFF8)&&(l_fat_signature != 0x7FFFFFF8))
        return MMC_ERR_BR_BAD_RESERVED_SECTORS;
```

```
    /* Verify that the number of partition sectors or extended number of partition
sectors
       is valid */

    if(ref_num_of_sectors <= 0xFFFF) {
        l_num_partition_sectors = ((unsigned long)
(pBootRecord->UB_no_of_sectors_on_partition)<<8) |
                                  (unsigned long)
(pBootRecord->LB_no_of_sectors_on_partition);
    }
    else {

        l_num_partition_sectors =
         ((unsigned long)(pBootRecord->byte1_extended_no_of_sectors_on_partition)     |
         ((unsigned long)(pBootRecord->byte2_extended_no_of_sectors_on_partition)<<8) |
         ((unsigned long)(pBootRecord->byte3_extended_no_of_sectors_on_partition)<<16)|
         ((unsigned long)
(pBootRecord->byte4_extended_no_of_sectors_on_partition)<<24));
    }

    /* <susmit : Check whether the format type is FAT16 : 04-Mar-2003> */
    if( ((l_num_partition_sectors/pBootRecord->sectors_per_cluster)<4085)||
        ((l_num_partition_sectors/pBootRecord->sectors_per_cluster)>=65525) )
        return MMC_ERR_BR_BAD_FILE_SYS_TYPE;

    /* Verify that number of root directory entries is 512 */
    /* <susmit : This might be different for some disks but for a FAT16 MMC this check
should be okay : 04-Mar-2003> */
    if(((pBootRecord->UB_no_of_root_dir_entries<<8) |
(pBootRecord->LB_no_of_root_dir_entries)) != 512)
        return MMC_ERR_BR_BAD_NUM_OF_ROOT_ENTRIES;

    /* <susmit : If l_num_partition_sectors is not equal to
sectors_in_partition_from_MBR for HD format
     *  and l_num_partition_sectors is not equal to disk_sectors_from_csd for FD
format, return error : 04-Mar-2003> */
    if(l_num_partition_sectors != ref_num_of_sectors)
        return  MMC_ERR_BR_BAD_NUM_PART_SECTORS;

    /* Verify that sectors per fat is between 1 to 256 */
    l_sectors_per_fat = (pBootRecord->UB_sectors_per_fat <<8) |
(pBootRecord->LB_sectors_per_fat);
    if((l_sectors_per_fat > 256) || (l_sectors_per_fat <1))
        return MMC_ERR_BR_BAD_SECTORS_PER_FAT;

    /* <susmit : We should verify that the number of clusters match the no. of sectors
per fat calculations : 04-Mar-2003> */
    if( (((l_num_partition_sectors - ((unsigned
 long)l_sectors_per_fat*pBootRecord->no_of_fats) -
l_reserved_sectors)/pBootRecord->sectors_per_cluster)*2) \
          > ((unsigned long)l_sectors_per_fat*512) )
        return MMC_ERR_BR_BAD_SECTORS_PER_FAT;

     /* Verify that boot record signature is 0x55AA */
    if( pBootRecord->signature != 0x55AA)
        return MMC_ERR_BR_BAD_SIGNATURE;


    return MMC_ERR_NONE;

 }
```