AN2392
Rev. 1, 8/2005

# Using the DSI Header File With the MSC8122/26ADS

By Iantha Scheiwe

The Freescale MSC8102, MSC8122, and MSC8126 DSP devices include a direct slave interface (DSI) by which a host can gain direct access to internal MSC8122 resources. The information in this document applies to all three devices, but the MSC8122 is the referenced device in this discussion. On the MSC8122 application development system board (MSC8122/26ADS), an MSC8103 host interfaces with the MSC8122 DSI, which connects to the 60x-compatible system bus of the MSC8103 host. The MSC8103 controls access using its user-programmable machine (UPM) memory controller, so the MSC8122 internal resources appear to the MSC8103 as an additional memory-mapped area. Therefore, the MSC8103 uses a standard C header file to reference internal MSC8122 resources easily. This header file, `MSC812xDSIHost_RegMemMap.h`, is provided with the MSC8122 development tools. The MSC8103 host is used for the example in this application note. However, note that the `MSC812xDSIHost_RegMemMap.h` file can be used with any host and is not restricted for use by an MSC8103 host. If you use a different host, its `typdefs.h` file may need to be modified to match the host's compiler types.

The DSI header file includes references to MSC8122 internal resources such as M1 and M2 memory regions, IPBus 1 and 2 registers, and system bus registers. Each resource is placed at the correct offset in the MSC8122 memory map for easy access by a host. When programming code for the host, you must set the base for the MSC8122 structure. The host can then reference any MSC8122 resource using the syntax contained in the DSI

**CONTENTS**

1 Example DSI Header File ..... 2
2 Example Host Code ..... 2
3 Header File Code ..... 3

host map file. The MSC812xDSIHost_RegMemMap.h file looks similar to the Msc812x_RegMemMap.h header file. Because some internal resources of the MSC8103 host system bus have names and structures that are similar to those of the MSC8122 system bus, structure names have been modified in the MSC812xDSIHost_RegMemMap.h file so that the linker does not encounter conflicts when both the msc8103.h and MSC812xDSIHost_RegMemMap.h files are used in the same project. The structures with modified names in the host map file are the DMA channel parameter RAM and the memory controller registers.

# 1 Example DSI Header File

A short example is included here to show you how to use the structures in the MSC812xDSIHost_RegMemMap.h file. The CodeWarrior™ development tools initialize the MSC8103 UPM memory controller, so initialization is assumed and not included in the following example. However, you must ensure that one of the DSI boot mode options is selected in the development tools so that the board is properly initialized for DSI mode. The MSC8122ADS_Stationery_Readme.txt file explains which switches on the board must be set for DSI mode. To test this example, use “DSI32 DSI Boot Mode.” The example code operates as follows:

1. Initializes pointers to both the msc8103 internal memory map structure (pstIMM) and the msc8122 memory map structure accessible through the DSI port (pstDSI).
2. Writes to an internal MSC8103 DMA register to show how MSC8103 resources are referenced using msc8103.h.
3. Writes to a MSC8122 DMA register to show how the MSC8103 may initialize MSC8122 modes of operation using the syntax from MSC812xDSIHost_RegMemMap.
4. MSC8103 writes and then reads a block of MSC8122 M2 and M1 memory.

The MSC8103 device avoids contention by employing semaphores to ensure that the memory regions inside the MSC8122 are not in use by another master. In this example, no other master is active, but semaphores are generally required in such a multi-master system and are shown here to make the example more complete.

The project for this example is located in the CodeWarrior for StarCore installation at: ..\CodeWarrior\(CodeWarrior_Examples)\StarCore_Examples\DSI.

# 2 Example Host Code

```
#include "MSC812xDSIHost_RegMemMap.h"
#include "msc8103.h"
#define DSI_BASE 0x25e00000
#define IMM_BASE 0x14700000

void main(void)
{
   stMsc812xDSIHost_RegMemMap *pstDSI;     /* DSI bus pointer */
   t_8103IMM *pstIMM;                      /* pointer to MSC8103 internal memory map */
   VUWord32 temp,i;
   VUByte data[8]={0xA1,0xB2,0xC3,0xD4,0xE5,0xF6,0x07,0x98};
   VUByte readdata[8]={0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0};

   pstIMM = (t_8103IMM *)(IMM_BASE);             /* MSC8103 internal register map */
   pstDSI = (stMsc812xDSIHost_RegMemMap *)(DSI_BASE); /* DSI interface memory map */
   //*********************************
   // Access MSC8103 System Bus Register
   //*********************************
   pstIMM->dchcr[0]=0x40020040;                                  /* IMM_BASE+0x10700 */
```

```
   //*********************************
   // Access MSC812x System Bus Register
   //*********************************
   pstDSI->DSISystemRegs.avuliDCHCR[0]=0x00000040;               /* DSI_BASE+0x1D0700 */

   //*****************************
   // Access MSC812x IPBus Register
   // Access MSC812x M1&M2 memories
   //*****************************
   // Write data to MSC812x M2 Mem,
   // check/lock semaphore before accessing MSC812x resource
   pstDSI->DSIIPBus1Regs.astHSMPR[0].vuliHSMPR=0x00000012;       /* DSI_BASE+0x1BC100 */
   temp = pstDSI->DSIIPBus1Regs.astHSMPR[0].vuliHSMPR;
   if (temp == 0x00000012)
      for (i=0; i<4; i++)
         pstDSI->avucDSIM2Mem[i*4] = (VUByte) data[i];            /* DSI_BASE+0x0 */
   // clear semaphore when write to MSC812x resource is complete
   pstDSI->DSIIPBus1Regs.astHSMPR[0].vuliHSMPR = 0x0;

   // Write data to MSC812x M1 Mem
   pstDSI->DSIIPBus1Regs.astHSMPR[6].vuliHSMPR=0x00000012;       /* DSI_BASE+0x1BC130 */
   temp = (VUWord32) pstDSI->DSIIPBus1Regs.astHSMPR[6].vuliHSMPR;
   if (temp == 0x00000012)
      for (i=4; i<8; i++)
         pstDSI->astDSICore[0].avucM1Mem[i*4] = (VUByte) data[i]; /* DSI_BASE+0x80000 */
   pstDSI->DSIIPBus1Regs.astHSMPR[6].vuliHSMPR = 0x0;

   // Read MSC812x M2 and M1 Mem
   // Not necessary to use a semaphore when reading a shared resource
   for (i=0; i<4; i++)
      readdata[i] = (VUByte) pstDSI->avucDSIM2Mem[i*4];
   for (i=4; i<8; i++)
      readdata[i] = (VUByte) pstDSI->astDSICore[0].avucM1Mem[i*4];

   asm(" debug");
   asm(" nop");

} /* end main() */
```

# 3 Header File Code

```
/*************************************************************************
* Freescale Semiconductor Inc. ******************************************/
/*************************************************************************
*
* MODULE NAME: Msc812xDSIHost_RegMemMap.h
*
*************************************************************************
* DESCRIPTION:
*       Internal Register Structures for the MSC812x accessible from a
*       host via the MSC812x DSI interface.
*       These include: M1 and M2 memories, IP Bus 1 structure, the System
*       Registers, and IP Bus 2 structure.
*
* VERSION:    v1.0
*
* COMPILER:   Metrowerks, CodeWarrior for Starcore v 2.6
*************************************************************************/
#ifndef __DSIHOSTMAP_H
#define __DSIHOSTMAP_H

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

```
* Include files
*****************************************************************************/
#include "typedefs.h"
/*---------------------------------------------------------------------------*/
/*     DEFINITION OF MSC812x Register Structures via DSI Interface           */
/*---------------------------------------------------------------------------*/
/*****************************************************************************
* MSC812x registers at the DSI offset are defined in a data structure called
* "stMsc812xDSIHost_RegMemMap".
* To access the MSC812x registers using this structure do the following:
*
* 1. Declare a global DSI pointer.
*    stMsc812xDSIHost_RegMemMap *pstDSI; // MSC812x register structure pointer
*
* 2. Initialize the DSI pointer:
*    pstDSI = (stMsc812xDSIHost_RegMemMap *)(DSI_BASE);
*    where
*    #define DSI_BASE    0x25e00000     // DSI map base address
*    Note that this base value is the default value for the MSC812xADS using
*    CodeWarrior2.6. If another base address is used by the host be sure to
*    change this #define value. (Check the MSC812x CHIP_ID register setting
*    to verify the correct address).
*
* 3. Access the DSI registers in the application code using pointer to the
*    DSI structure, for example:
*    pstDSI->DSIIPBus1Regs.astTDM[2].vuliTDMRDBFT = 0x129f0001;
*    pstDSI->DSISystemRegs.astDSIDMAChPRAM[0].vuliBDAddr = 0x20005032;
*****************************************************************************/

/*---------------------------------------------------------------------------*/
/*             Ethernet Controller STRUCTURE                                 */
/*---------------------------------------------------------------------------*/
//size=8bytes: align on 8-byte boundary: start addr ends in 0 or 8
typedef __PACK__ struct stBufferDescriptor
{
    VUWord16  vusiControlStatus;    /* control and status */
    VUWord16  vusiLength;           /* transfer length */
    void*     pvBufferAddress;      /* buffer address */
} stBufferDescriptor;

// OPTIONAL: 32-byte Pattern Insertion Transmit Buffer Descriptor
typedef __PACK__ struct stTxBufferDescriptor
{
stBufferDescriptor stBD;
    volatile  void*     pvDataBufferAddress;    /* buffer address */
    VUWord32  vuliReserved2;
    volatile  void*     pvInsertBufferAddress;  /* buffer address */
    VUWord16  vusiReserved3;
    VUWord16  vusiInsertCtrl;
    VUWord16  vusiInsertIndex;
    VUWord16  vusiInsertLength;
    VUWord32  vuliReserved4;
} stTxBufferDescriptor;

// OPTIONAL: 32-byte Pattern Matching Receive Buffer Descriptor
typedef __PACK__ struct stRxBufferDescriptor
{
stBufferDescriptor stBD;
    volatile  void*     pvDataBufferAddress;    /* buffer address */
    VUWord32  vuliReserved2;
    VUWord32  vuliReserved3;
    VUWord16  vusiPatternCtrl;
    VUWord16  vusiReserved4;
    VUWord32  vuliReserved5;
```

```
    VUWord16  vusiReserved6;
    VUWord16  vusiByteCount;
} stRxBufferDescriptor;


typedef struct stEnet
{
    VUByte    avucReserved1[0xC];
    VUWord32  vuliIEVENT;/* Interrupt event register */
    VUWord32  vuliIMASK;/* Interrupt mask register */
    VUWord32  vuliEDIS;/* Error disabled register */
    VUByte    avucReserved2[4];
    VUWord32  vuliECNTRL;/* Ethernet control register */
    VUWord32  vuliMINFLR;/* Minimum frame length */
    VUWord32  vuliPTV;/* Pause time value register */
    VUWord32  vuliDMACTRL;/* DMA control register */
    VUWord32  vuliTBIPA;/* TBI PHY address register */
    VUByte    avucReserved3[4];
    VUWord32  vuliDMAMR;/* DMA Maintenance register */
    VUByte    avucReserved31[0xC];

/*  FIFO CONTROL AND STATUS REGISTERS   */
    VUWord32  vuliFRXSTATR;/* FIFO Receive Status Register */
    VUWord32  vuliFRXCTRLR;/* FIFO Receive Control Register */
    VUWord32  vuliFRXALAR;/* FIFO Receive Alarm Register */
    VUWord32  vuliFRXSHR;/* FIFO Receive Alarm Shutoff Register */
    VUWord32  vuliFRXPAR;/* FIFO Receive Panic Register */
    VUWord32  vuliFRXPSR;/* FIFO Receive Shutoff Register */
    VUByte    avucReserved4[0x18];
    VUWord32  vuliFTXSTATR;/* FIFO Transmit Status Register */
    VUByte    avucReserved41[0x10];
    VUWord32  vuliFTXTHR;/* FIFO Transmit Threshold Register */
    VUByte    avucReserved42[0x4];
    VUWord32  vuliFTXSPR;/* FIFO Transmit Space Available Register */
    VUWord32  vuliFTXSR;/* FIFO Transmit Starve Register */
    VUWord32  vuliFTXSSR;/* FIFO Transmit Starve Shutoff Register */
    VUByte    avucReserved43[0x60];

/*  TSEC TRANSMIT   */
    VUWord32  vuliTCTRL;/* Transmit control register */
    VUWord32  vuliTSTAT;/* Transmit status register */
    VUByte    avucReserved5[4];
    VUWord32  vuliTBDLEN;/* TxBD data length */
    VUByte    avucReserved6[0x14];

    volatile  stBufferDescriptor* pstCTBPTR;         /* Current TxBD pointer */
    VUByte    avucReserved7[0x5c];
    volatile  stBufferDescriptor* pstTBPTR;          /* TxBD pointer */
    VUByte    avucReserved8[0x7c];
    volatile  stBufferDescriptor* pstTBASE;          /* TxBD base buffer descriptor */
    VUByte    avucReserved9[0xa8];
    volatile  stTxBufferDescriptor stOSTBD; /* Out-of-sequence TxBD register */
    VUByte    avucReserved10[0x30];/* {OSTBD,OSTBDP,OS32TBDP,OS32IPTRL,OS32TBDR,OS32IIL}*/

/*  TSEC RECEIVE    */
    VUWord32  vuliRCTRL;/* Receive control register */
    VUWord32  vuliRSTAT;/* Receive status register */
    VUByte    avucReserved11[4];
    VUWord32  vuliRBDLEN;/* RxBD data length */
    VUByte    avucReserved12[0x14];
    VUWord32  vuliCRBPTRL;/* Current RxBD pointer */
    VUByte    avucReserved13[0x18];
    VUWord32  vuliMRBLR0R1;/* Maximum receive buffer length R0R1 register*/
    VUWord32  vuliMRBLR2R3;/* Maximum receive buffer length R2R3 register */
```

```
    VUByte    avucReserved14[0x38];
    struct astRBPTR
    {
 VUByte   avucReserved14a[0x4]; /* {RBPTR0, RBPTR1, RBPTR2, RBPTR3}*/
 volatile stBufferDescriptor* pstRBPTR;       /* RxBD pointer */
} astRBPTR[4];
    VUByte    avucReserved15[0x60];
    struct astRBASE
    {
 VUByte   avucReserved15a[0x4]; /* {RBASE0, RBASE1, RBASE2, RBASE3}*/
 volatile stBufferDescriptor* pstRBASE;       /* RxBD base address */
} astRBASE[4];
    VUByte    avucReserved16[0xe0];

/*  MAC REGISTERS   */
    VUWord32  vuliMACCFG1R;/* MAC configuration #1 */
    VUWord32  vuliMACCFG2R;/* MAC configuration #2 */
    VUWord32  vuliIPGIFGIR;/* IPG/IFG */
    VUWord32  vuliHAFDUPR;/* Half-duplex */
    VUWord32  vuliMAXFRMR;/* Maximum frame */
    VUByte    avucReserved17[0xc];
    VUWord32  vuliMIIMCFGR;/* MII Mgmt:configuration */
    VUWord32  vuliMIIMCOMR;/* MII Mgmt:command */
    VUWord32  vuliMIIMADDR;/* MII Mgmt:address */
    VUWord32  vuliMIIMCONR;/* MII Mgmt:control */
    VUWord32  vuliMIIMSTATR;        /* MII Mgmt:status */
    VUWord32  vuliMIIMINDR;/* MII Mgmt:indicators */
    VUByte    vuliReserved18[4];
    VUWord32  vuliIFSTATR;/* Interface status */
    VUWord32  vuliMACSTADDR1R;/* Station Address, part 1 */
    VUWord32  vuliMACSTADDR2R;/* Station Address, part 2 */
    VUByte    avucReserved19[0x138];

/*  RMON MIB REGISTERS  */
/*  TRANSMIT and RECEIVE COUNTERS */
    VUWord32  vuliTR64;             /* transmit and receive 64 byte frame counter */
    VUWord32  vuliTR127;/* transmit and receive 65 to 127 byte frame counter */
    VUWord32  vuliTR255;/* transmit and receive 128 to 255 byte frame counter */
    VUWord32  vuliTR511;/* transmit and receive 256 to 511 byte frame counter */
    VUWord32  vuliTR1K;             /* transmit and receive 512 to 1023 byte frame counter */
    VUWord32  vuliTRMAX;/* transmit and receive 1024 to 1518 byte frame counter */
    VUWord32  vuliTRMGV;/* transmit and receive 1519 to 1522 byte good VLAN frame count */

/* RECEIVE COUNTERS */
    VUWord32  vuliRBYT;             /* receive byte counter */
    VUWord32  vuliRPKT;             /* receive packet counter */
    VUWord32  vuliRFCS;             /* receive FCS error counter */
    VUWord32  vuliRMCA;             /* RMCA receive multicast packet counter */
    VUWord32  vuliRBCA;             /* receive broadcast packet counter */
    VUWord32  vuliRXCF;             /* receive control frame packet counter */
    VUWord32  vuliRXPF;             /* receive PAUSE frame packet counter */
    VUWord32  vuliRXUO;             /* receive unknown OP code counter */
    VUWord32  vuliRALN;             /* receive alignment error counter */
    VUWord32  vuliRFLR;             /* receive frame length error counter */
    VUWord32  vuliRCDE;             /* receive code error counter */
    VUWord32  vuliRCSE;             /* receive carrier sense error counter */
    VUWord32  vuliRUND;             /* receive undersize packet counter */
    VUWord32  vuliROVR;             /* receive oversize packet counter */
    VUWord32  vuliRFRG;             /* receive fragments counter */
    VUWord32  vuliRJBR;             /* receive jabber counter */
    VUWord32  vuliRDRP;             /* receive drop */

/* TRANSMIT COUNTERS */
    VUWord32  vuliTBYT;             /* transmit byte counter */
```

```
    VUWord32  vuliTPKT;             /* transmit packet counter */
    VUWord32  vuliTMCA;             /* transmit multicast packet counter */
    VUWord32  vuliTBCA;             /* transmit broadcast packet counter */
    VUWord32  vuliTXPF;             /* transmit PAUSE control frame counter */
    VUWord32  vuliTDFR;             /* transmit deferral packet counter */
    VUWord32  vuliTEDF;             /* transmit excessive deferral packet counter */
    VUWord32  vuliTSCL;             /* transmit single collision packet counter */
    VUWord32  vuliTMCL;             /* transmit multiple collision packet counter */
    VUWord32  vuliTLCL;             /* transmit late collision packet counter */
    VUWord32  vuliTXCL;             /* transmit excessive collision packet counter */
    VUWord32  vuliTNCL;             /* transmit total collision counter */
    VUWord32  vuliTPFH;             /* transmit PAUSE frames honored counter */
    VUWord32  vuliTDRP;             /* transmit drop frame counter */
    VUWord32  vuliTJBR;             /* transmit jabber frame counter */
    VUWord32  vuliTFCS;             /* transmit FCS error counter */
    VUWord32  vuliTXCF;             /* transmit control frame counter */
    VUWord32  vuliTOVR;             /* transmit oversize frame counter */
    VUWord32  vuliTUND;             /* transmit undersize frame counter */
    VUWord32  vuliTFRG;             /* transmit fragments frame counter */

/* GENERAL REGISTERS */
    VUWord32  vuliCAR1;             /* carry register one register* */
    VUWord32  vuliCAR2;             /* carry register two register* */
    VUWord32  vuliCAM1;             /* carry register one mask register */
    VUWord32  vuliCAM2;             /* carry register two mask register */
    VUByte    avucReserved20[0xc0];

/*  HASH FUNCTION REGISTERS */
    VUWord32  avuliIADDR[8];        /* Individual address registers 0-7 */
    VUByte    avucReserved21[0x60]; /* {IADDR0, IADDR1, IADDR2, IADDR3, IADDR4, IADDR5, IADDR6,
IADDR7}*/
    VUWord32  avuliGADDR[8];        /* Group address registers 0-7 */
    VUByte    avucReserved22[0x60]; /* {GADDR0, GADDR1, GADDR2, GADDR3, GADDR4, GADDR5, GADDR6,
GADDR7}*/

/*  PATTERN MATCHING REGISTERS  */
    struct astPMRegs
    {
VUWord32  vuliPMD;          /* Pattern match data  */
VUByte    avucReserved23[4];
VUWord32  vuliPMASK;        /* Pattern mask register */
VUByte    avucReserved24[4];
VUWord32  vuliPCNTRL;       /* Pattern control register */
VUByte    avucReserved25[4];
VUWord32  vuliPATTRB;       /* Pattern attributes register */
VUWord32  vuliPATTRBELI;    /* Pattern attributes EL & EI register */
    } astPMRegs[16];

    VUByte    avucReserved71[0xf8];
    VUWord32  vuliDATTR;            /* Default attribute register */

/*  FUTURE EXPANSION SPACE  */
    VUByte    avucReserved72[0x404];

/*  MII GASKET REGISTERS  */
    VUWord32  vuliMIIGSK_CFGR;
    VUWord32  vuliMIIGSK_GPR;
    VUWord32  vuliMIIGSK_ENR;
    VUWord32  vuliMIIGSK_SMII_SYNCDIR;
    VUWord32  vuliMIIGSK_TIFBR;
    VUWord32  vuliMIIGSK_RIFBR;
    VUWord32  vuliMIIGSK_ERIFBR;
    VUWord32  vuliMIIGSK_IEVENT;
    VUWord32  vuliMIIGSK_IMASK;
```

```
}stEnet;

/*------------------------------------------------------------------------*/
/*                        TDM  Structure                                  */
/*------------------------------------------------------------------------*/
typedef struct stTDM
{
    VUByte    avucTDMRXLCLMEM[2048];    /* TDM Receive Local Memory*/
    VUByte    avucReserved1[2048];
    VUWord32  avuliTDMRCPR[256];        /* TDM Receive Channel Parameters Register n. (n=0-255)*/
    VUByte    avucReserved2[1024];      /* {TDM0RCPR[0-255], TDM1RCPR[0-255],TDM2RCPR[0-255],
TDM3RCPR[0-255]}*/
    VUByte    avucTDMTXLCLMEM[2048];    /* TDM Transmit Local Memory*/
    VUByte    avucReserved3[2048];
    VUWord32  avuliTDMTCPR[256];        /* TDM Transmit Channel Parameters Register n. (n=0-255)*/
    VUByte    avucReserved4[0x1320];    /* {TDM0TCPR[0-255], TDM1TCPR[0-255],TDM2TCPR[0-255],
TDM3TCPR[0-255]}*/
    VUWord32  vuliTDMTSR;               /* TDM Transmit Status Register*/
    VUByte    avucReserved6[4];         /* {TDM0TSR, TDM1TSR,TDM2TSR,    TDM3TSR}*/
    VUWord32  vuliTDMRSR;               /* TDM Receive Status Register*/
    VUByte    avucReserved7[4];         /* {TDM0RSR, TDM1RSR,TDM2RSR,    TDM3RSR}*/
    VUWord32  vuliTDMASR;               /* TDM Adaptation Status Register*/
    VUByte    avucReserved8[4];         /* {TDM0ASR, TDM1ASR,TDM2ASR,    TDM3ASR}*/
    VUWord32  vuliTDMTER;               /* TDM Transmit Event Register*/
    VUByte    avucReserved9[4];         /* {TDM0TER, TDM1TER,TDM2TER,    TDM3TER}*/
    VUWord32  vuliTDMRER;               /* TDM Receive Event Register*/
    VUByte    avucReserved10[4];        /* {TDM0RER, TDM1RER,TDM2RER,    TDM3RER}*/
    VUWord32  vuliTDMTNB;               /* TDM Transmit Number of Buffers*/
    VUByte    avucReserved11[4];        /* {TDM0TNR, TDM1TNR,TDM2TNR,    TDM3TNR}*/
    VUWord32  vuliTDMRNB;               /* TDM Receive Number of Buffers*/
    VUByte    avucReserved12[4];        /* {TDM0RNR, TDM1RNR,TDM2RNR,    TDM3RNR}*/
    VUWord32  vuliTDMTDBDR;             /* TDM Transmit Data Buffer Displacement Register*/
    VUByte    avucReserved13[4];        /* {TDM0TDBDR, TDM1TDBDR, TDM2TDBDR,  TDM3TDBDR}*/
    VUWord32  vuliTDMRDBDR;             /* TDM Receive Data Buffer Displacement Register*/
    VUByte    avucReserved14[4];        /* {TDM0RDBDR, TDM1RDBDR, TDM2RDBDR,  TDM3RDBDR}*/
    VUWord32  vuliTDMASDR;              /* TDM Adaptation Sync Distance Register*/
    VUByte    avucReserved15[4];        /* {TDM0ASDR, TDM1ASDR,  TDM2ASDR,   TDM3ASDR}*/
    VUWord32  vuliTDMTIER;              /* TDM Transmit Interrupt Enable Register*/
    VUByte    avucReserved16[4];        /* {TDM0TIER, TDM1TIER,  TDM2TIER,   TDM3TIER}*/
    VUWord32  vuliTDMRIER;              /* TDM Receive Interrupt Enable Register*/
    VUByte    avucReserved17[4];        /* {TDM0RIER, TDM1RIER,  TDM2RIER,   TDM3RIER}*/
    VUWord32  vuliTDMTDBST;             /* TDM Transmit Data Buffer Second Threshold*/
    VUByte    avucReserved18[4];        /* {TDM0TDBST, TDM1TDBST, TDM2TDBST,  TDM3TDBST}*/
    VUWord32  vuliTDMRDBST;             /* TDM Receive Data Buffer Second Threshold*/
    VUByte    avucReserved19[4];        /* {TDM0RDBST, TDM1RDBST, TDM2RDBST,  TDM3RDBST}*/
    VUWord32  vuliTDMTDBFT;             /* TDM Transmit Data Buffer First Threshold*/
    VUByte    avucReserved20[4];        /* {TDM0TDBFT, TDM1TDBFT, TDM2TDBFT,  TDM3TDBFT}*/
    VUWord32  vuliTDMRDBFT;             /* TDM Receive Data Buffer First Threshold*/
    VUByte    avucReserved21[4];        /* {TDM0RDBFT, TDM1RDBFT, TDM2RDBFT,  TDM3RDBFT}*/
    VUWord32  vuliTDMTCR;               /* TDM Transmit Control Register*/
    VUByte    avucReserved22[4];        /* {TDM0TCR, TDM1TCR,TDM2TCR,    TDM3TCR}*/
    VUWord32  vuliTDMRCR;               /* TDM Receive Control Register*/
    VUByte    avucReserved23[4];        /* {TDM0RCR, TDM1RCR,TDM2RCR,    TDM3RCR}*/
    VUWord32  vuliTDMACR;               /* TDM Receive Control Register*/
    VUByte    avucReserved24[4];        /* {TDM0ACR, TDM1ACR,TDM2ACR,    TDM3ACR}*/
    VUWord32  vuliTDMTGBA;              /* TDM Transmit Global Base Address*/
    VUByte    avucReserved25[4];        /* {TDM0TGBA, TDM1TGBA,  TDM2TGBA,   TDM3TGBA}*/
    VUWord32  vuliTDMRGBA;              /* TDM Receive Global Base Address*/
    VUByte    avucReserved26[4];        /* {TDM0RGBA, TDM1RGBA,  TDM2RGBA,   TDM3RGBA}*/
    VUWord32  vuliTDMTDBS;              /* TDM Transmit Data Buffer Size*/
    VUByte    avucReserved27[4];        /* {TDM0TDBS, TDM1TDBS,  TDM2TDBS,   TDM3TDBS}*/
    VUWord32  vuliTDMRDBS;              /* TDM Receive Data Buffer Size*/
    VUByte    avucReserved28[4];        /* {TDM0RDBS, TDM1RDBS,  TDM2RDBS,   TDM3RDBS}*/
```

```
    VUWord32  vuliTDMTFP;               /* TDM Transmit Frame Parameters*/
    VUByte    avucReserved29[4];        /* {TDM0TFP, TDM1TFP,TDM2TFP,    TDM3TFP}*/
    VUWord32  vuliTDMRFP;               /* TDM Receive Frame Parameters*/
    VUByte    avucReserved30[4];        /* {TDM0RFP, TDM1RFP,TDM2RFP,    TDM3RFP}*/
    VUWord32  vuliTDMTIR;               /* TDM Transmit Interface Register*/
    VUByte    avucReserved31[4];        /* {TDM0TIR, TDM1TIR,TDM2TIR,    TDM3TIR}*/
    VUWord32  vuliTDMRIR;               /* TDM Receive Interface Register*/
    VUByte    avucReserved32[4];        /* {TDM0RIR, TDM1RIR,TDM2RIR,    TDM3RIR}*/
    VUWord32  vuliTDMGIR;               /* TDM General Interface Register*/
    VUByte    avucReserved33[4];        /* {TDM0GIR, TDM1GIR,TDM2GIR,    TDM3GIR}*/
}stTDM;

/*-------------------------------------------------------------------------*/
/*DEFINITION OF MSC812x IPBus1 REGISTER STRUCTURE                 */
/*-------------------------------------------------------------------------*/
typedef struct stMsc812xIPBus1
{
/*  TDM 0,1,2,3    - 0x0000 */
    stTDM     astTDM[4];
    VUByte    avucReserved0[0x28004];

/*  Ethernet Controller - 0x38000 */
    stEnet    stEnet;
    VUByte    avucReserved0a[0x1FDC];

/* IP Master - 0x3B000 */
    VUWord32  vuliSCR;              /* Stop Control Register */
    VUByte    avucReserved1a[4];
    VUWord32  vuliSASR;             /* Stop Acknowledge Status Register */
    VUByte    avucReserved1b[0xFF4];

/* GIC - 0x3C000 */
    VUWord32  vuliVIGR;             /* Virtual Interrupt Generation Register */
    VUByte    avucReserved2[4];
    VUWord32  vuliVISR;             /* Virtual Interrupt Status Register */
    VUByte    avucReserved3[4];
    VUWord32  vuliVNMIGR;           /* Virtual NMI Generation Register */
    VUByte    avucReserved4[4];
    VUWord32  vuliGICR;             /* GIC Interrupt Configuration Register */
    VUByte    avucReserved5[4];
    VUWord32  vuliGEIER;            /* GIC External Interrupt Enable Register */
    VUByte    avucReserved6[4];
    VUWord32  vuliGCIER;            /* GIC Core Interrupt Enable Register */
    VUByte    avucReserved7[4];
    VUWord32  vuliGISR;             /* GIC Interrupt Status Register */
    VUByte    avucReserved8[204];

/* HS - 0x3C100   */
    struct {
VUWord32  vuliHSMPR;        /* Hardware Semaphore Register */
VUByte    avucReserved[4];
    } astHSMPR[8];
    VUByte    avucReserved11[192];

/* GPIO   - 0x3C200 */
    VUWord32  vuliPODR;             /* Pin Open-Drain Register */
    VUByte    avucReserved12[4];
    VUWord32  vuliPDAT;             /* Pin Data Register */
    VUByte    avucReserved13[4];
    VUWord32  vuliPDIR;             /* Pin Direction Register */
    VUByte    avucReserved14[4];
    VUWord32  vuliPAR;              /* Pin Assignment Register */
    VUByte    avucReserved15[4];
    VUWord32  vuliPSOR;             /* Pin Special Options Register */
```

```
    VUByte    avucReserved17[0xDDC];

/* UART   - 0x3D00 */
    VUWord32  vuliSCIBR;            /* SCI Baud Rate Register */
    VUByte    avucReserved18[4];
    VUWord32  vuliSCICR;            /* SCI Control Register */
    VUByte    avucReserved19[4];
    VUWord32  vuliSCISR;            /* SCI Status Register */
    VUByte    avucReserved20[4];
    VUWord32  vuliSCIDR;            /* SCI Data Register */
    VUByte    avucReserved21[12];
    VUWord32  vuliSCIDDR;           /* SCI Data Direction Register */
    VUByte    avucReserved22[0xFD4];

/* DSI - 0x3E00 */
    VUWord32  vuliDCR;              /* DSI Control Register */
    VUByte    avucReserved24[4];
    VUWord32  vuliDSWBAR;           /* DSI Sliding Window Base Address Register */
    VUByte    avucReserved25[4];
    VUWord32  vuliDIBAR9;           /* DSI Internal Base Address Register Bank 9 */
    VUByte    avucReserved26[4];
    VUWord32  vuliDIBAR10;          /* DSI Internal Base Address Register Bank 10 */
    VUByte    avucReserved27[4];
    VUWord32  vuliDIBAR11;          /* DSI Internal Base Address Register Bank 11 */
    VUByte    avucReserved28[4];
    VUWord32  vuliDIAMR9;           /* DSI Internal Address Mask Register Bank 9 */
    VUByte    avucReserved29[4];
    VUWord32  vuliDIAMR10;          /* DSI Internal Address Mask Register Bank 10 */
    VUByte    avucReserved30[4];
    VUWord32  vuliDIAMR11;          /* DSI Internal Address Mask Register Bank 11 */
    VUByte    avucReserved31[4];
    VUWord32  vuliDCIR;             /* DSI Chip ID Register */
    VUByte    avucReserved32[4];
    VUWord32  vuliDDR;              /* DSI Disable Register */
VUByte    avucReserved33[0x14];
VUWord32  vuliDEXTBAR;         /* NEW for 8122: DSI External Sliding Window Base Address Register */
    VUByte    avucReserved33a[0x79C];
    VUWord32  vuliDSR;              /* DSI Status Register */
    VUByte    avucReserved34[0x4];
    VUWord32  vuliDER;              /* DSI Error Register */
    VUByte    avucReserved345[0x7F4];

/* Timers A,B   - 0x3F000 */
    struct astTimer
    {
struct
{
VUWord32  vuliTCFR;     /* Timer Configuration Register of Timer */
VUByte    avucReserved[4];
} astTCFRAB[16];
struct
{
VUWord32  vuliTCMP;     /* Timer Compare Register of Timer */
VUByte    avucReserved[4];
} astTCMPAB[16];
struct
{
VUWord32  vuliTCR;      /* Timer Control Register of Timer */
VUByte    avucReserved[4];
} astTCRAB[16];
struct
{
VUWord32  vuliTCNR;     /* Timer Count Register of Timer */
VUByte    avucReserved[4];
```

```
} astTCNRAB[16];
VUByte    avucReserved[0x180];
VUWord32  vuliTGCR;         /* Timer General Configuration Register */
VUByte    avucReserved1[4];
VUWord32  vuliTER;          /* Timer Event Register */
VUByte    avucReserved2[4];
VUWord32  vuliTIER;         /* Timer Interrupt Enable Register */
VUByte    avucReserved3[4];
VUWord32  vuliTSR;          /* Timer Status Register */
VUByte    avucReserved4[8];
VUByte    avucReserved5[8];
VUByte    avucReserved6[0x54];
    } astTimer[2];
    VUByte    avucReserved35[0x800];

} stMsc812xIPBus1;

/*-------------------------------------------------------------------------*/
/* DEFINITION OF MSC812x IPBus2 STRUCTURE (VCOP and TCOP) for MSC8126 only  */
/*-------------------------------------------------------------------------*/
typedef struct stMsc812xIPBus2
{
VUByte    avucReserved0[0xB000];
VUWord32  vuliSCR2;             /* Stop Control Register 2 */
    VUByte    avucReserved1[4];
VUWord32  vuliSASR2;            /* Stop Acknowledge Status Register 2 */
VUByte    avucReserved2[0x2FF4];

/* TCOP   - 0xE000 */
struct stTCOP
{
VUWord32  vuliTCR1;         /* TCOP Control Register 1 */
VUWord32  vuliTCR2;         /* TCOP Control Register 2 */
VUWord32  vuliTSR;          /* TCOP status Register */
VUWord32  vuliGSCR;         /* TCOP Group Select Control Register */
VUWord32  vuliTCFR;         /* TCOP Correction Factor Register */
VUWord32  vuliTSCR;         /* TCOP Stop Condition Register */
VUByte    avucReserved1[4];
VUWord32  vuliYSBA;         /* Ys Base Address Register */
VUWord32  vuliYP1BA;        /* Yp1 Base Address Register */
VUWord32  vuliYP1ABA;       /* Yp1a Base Address Register */
VUWord32  vuliYP2BA;        /* Yp2 Base Address Register */
VUWord32  vuliYP2ABA;       /* Yp2a Base Address Register */
VUWord32  vuliLBA;          /* Lambda Base Address Register */
VUWord32  vuliIOBA;         /* Interleave Offset Base Address Register */
VUWord32  vuliBIBA;         /* Beta Intermediate Base Address Register */
VUWord32  vuliSRBA;         /* Soft Result Base Address Register */
} stTCOP;
    VUWord32  vuliBR;               /* Base Register */
VUByte    avucReserved3[0xfbc];

/* VCOP   - 0xF000 */
struct stVCOP
{
VUWord32  vuliVPOLR;        /* VCOP Polynomials Register */
VUWord32  vuliVPPR;         /* VCOP Puncture Pattern Register */
VUWord32  vuliVCNT;         /* VCOP Trellis Count Register */
VUWord32  vuliVOBAR;        /* VCOP Output Buffer Address Register */
VUWord32  vuliVIBSAR;       /* VCOP Input Buffer Start Address Register */
VUWord32  vuliVIBBLR;       /* VCOP Input Buffer Block Length Register */
VUWord32  vuliVPMFAR;       /* VCOPPM Fill Address Register */
VUWord32  vuliVPMISR;       /* VCOPPM Init State Register */
VUWord32  vuliVISRA;        /* VCOP Interim Stage Register (and S-parameters) A */
VUWord32  vuliVISRB;        /* VCOP Interim Stage Register (and S-parameters) B */
```

```
VUWord32  vuliVISRC;        /* VCOP Interim Stage Register (and S-parameters) C */
VUWord32  vuliVAADAR;       /* VCOP Algorithm Assist Data Dump Address Register */
VUByte    avucReserved1[0x0010];
VUWord32  vuliVCONFR;       /* VCOP Configuration Register */
VUWord32  vuliVSTR;         /* VCOP Status Register */
VUByte    avucReserved2[0x0EB8];
VUWord32  vuliVPCR;         /* VCOP Performance Monitor Control Register */
VUWord32  vuliVPCA;         /* VCOP Performance Monitor Counter A */
VUWord32  vuliVPCB;         /* VCOP Performance Monitor Counter B */
} stVCOP;

VUByte    avucReserved4[0x00F4];
} stMsc812xIPBus2; /* MSC8126 only */

/*------------------------------------------------------------------------*/
/*          DEFINITION OF MSC812x 60x SYSTEM BUS REGISTER STRUCTURE       */
/*------------------------------------------------------------------------*/
typedef struct stMsc812xSysRegs
{
    VUByte    avucReserved0 [0x10000];

/* General SIU - 0x10000 */
    VUWord32  vuliSIUMCR;           /* SIU Module Configuration Register*/
    VUWord32  vuliSYPCR;            /* System Protection Control Register*/
    VUByte    avucReserved13[0x6];
    VUWord16  vusiSWSR;             /* Software Service Register*/

/* 60x System and Local Buses - 0x10010 */
    VUByte    avucReserved14[0x14];
    VUWord32  vuliBCR;              /* Bus Configuration Register*/
    VUByte    vucPPCACR;            /* Arbiter Configuration Register*/
    VUByte    avucReserved15[0x3];
    VUWord32  vuliPPCALRH;          /* Arbitration Level Reg. (1st clients)*/
    VUWord32  vuliPPCALRL;          /* Arbitration Level Reg. (Next clients)*/
    VUByte    vucLCLACR;            /* LCL Arbiter Configuration Register*/
    VUByte    avucReserved16[0x3];
    VUWord32  vuliLCLALRH;          /* LCL Arbitration Lvl Reg (1st clients)*/
    VUWord32  vuliLCLALRL;          /* LCL Arbitration Lvl Reg (Next clients)*/
    VUWord32  vuliTESCR1;           /* PPC bus transfer err stat ctl reg 1*/
    VUWord32  vuliTESCR2;           /* PPC bus transfer err stat ctl reg 2*/
    VUWord32  vuliLTESCR1;          /* Local bus transfer err stat ctl reg 1*/
    VUByte    avucReserved5[0xC];
    VUWord32  vuliLGTDTEA;          /* Local bus GTD (Global TDM DMA) Transfer Error Addr*/
    VUByte    vucLGTDTEM;           /* Local bus GTD (Global TDM DMA) Transfer Error MSNUM*/
    VUByte    avucReserved18[0x3];
    VUWord32  vuliPDMTEA;           /* PPC bus DMA Transfer Error Addr*/
    VUByte    vucPDMTER;            /* PPC bus DMA Transfer Error RQNUM*/
    VUByte    avucReserved95[0x3];
    VUWord32  vuliLDMTEA;           /* Local PPC bus DMA Transfer Error Addr*/
    VUByte    vucLDMTER;            /* Local PPC bus DMA Transfer Error RQNUM*/
    VUByte    avucReserved96[0x93];

/* Memory Controller - 0x10100 */
    struct astDSIMemCRegs           /* Note that BR8 and OR8 are reserved*/
    {
VUWord32  vuliBR;           /* Base Register */
VUWord32  vuliOR;           /* Option Register*/
    } astDSIMemCRegs[12];
    VUByte    avucReserved19[0x8];
    VUWord32  vuliMAR;              /* Memory Address Register*/
    VUByte    avucReserved20[0x4];
    VUWord32  vuliMAMR;             /* Machine A Mode Register*/
    VUWord32  vuliMBMR;             /* Machine B Mode Register*/
    VUWord32  vuliMCMR;             /* Machine C Mode Register*/
```

```
    VUByte    avucReserved21[0x8];
    VUWord16  vusiMPTPR;            /* Memory Periodic Timer Prescaler*/
    VUByte    avucReserved22[0x2];
    VUWord32  vuliMDR;              /* Memory Data Register*/
    VUByte    avucReserved23[0x4];
    VUWord32  vuliPSDMR;            /* PPC Bus SDRAM machine Mode Register*/
    VUByte    avucReserved97[0x4];
    VUByte    vucPURT;              /* PPC Bus assigned UPM Refresh Timer*/
    VUByte    avucReserved24[0x3];
    VUByte    vucPSRT;              /* PPC Bus assigned SDRAM Refresh Timer*/
    VUByte    avucReserved24a[0x3];
    VUByte    vucLURT;              /* PPC Bus assigned UPM Refresh Timer*/
    VUByte    avucReserved24b[0x3];
    VUByte    vucLSRT;              /* PPC Bus assigned SDRAM Refresh Timer*/
    VUByte    avucReserved25[0x3];
    VUWord32  vuliIMMR;             /* Internal Memory Map Register*/
    VUByte    avucReserved98[0x74];

/* System Integration Timers - 0x10200 */
    VUWord16  vusiTMCNTSC;          /* Time Counter Status & Control Reg*/
    VUByte    avucReserved30[0x2];
    VUWord32  vuliTMCNT;            /* Time Counter Register*/
    VUByte    avucReserved99[0x4];
    VUWord32  vuliTMCNTAL;          /* Time Counter Alarm Reg*/
    VUByte    avucReserved31[0x10];
    VUWord16  vusiPISCR;            /* Periodic Int Status & Control Reg*/
    VUByte    avucReserved32[0x2];
    VUWord32  vuliPITC;             /* Periodic Int Count Reg*/
    VUWord32  vuliPITR;             /* Periodic Int Timer Reg*/
    VUByte    avucReserved33[0x4b4];

/* DMA - 0x10700 */
    VUWord32  avuliDCHCR[16];       /* DMA Channel Configuration Register*/
    VUByte    avucReserved100[0x40];
    VUWord32  vuliDIMR;             /* DMA Internal Mask Register*/
    VUWord32  vuliDSTR;             /* DMA Status Register*/
    VUByte    vucDTEAR;             /* DMA TEA Status Register*/
    VUByte    avucReserved102[0x3];
    VUByte    vucDPCR;             /* DMA Pin Configuration Register*/
    VUByte    avucReserved103[0x3];
    VUWord32  vuliDEMR;             /* DMA External Mask Register*/
    VUByte    avucReserved104[0x6c];
    struct astDSIDMAChPRAM          /* DMA Channel Parameter RAM*/
    {
VUWord32  vuliBDAddr;       /* Buffer Descriptor Address*/
VUWord32  vuliBDSize;       /* Buffer Descriptor Transfer Size*/
VUWord32  vuliBDAttr;       /* Buffer Descriptor Attributes*/
VUWord32  vuliBDBSize;      /* Buffer Descriptor Base Size*/

    } astDSIDMAChPRAM[64];
    VUByte    avucReserved105 [136];

/* Clocks and Reset - 0x10C88 */
    VUWord32  vuliSCMSR;            /* System Clock Mode Register*/
    VUByte    avucReserved107[4];
    VUWord32  vuliRSR;              /* Reset Status Register*/
    VUByte    avucReserved106[108];

} stMsc812xSysRegs;

/*-------------------------------------------------------------------------------*/
/*        DEFINITION OF MSC812x INTERNAL MEMORY MAP FROM DSI HOST                */
/*-------------------------------------------------------------------------------*/
/*
```

```
0x000000-0x177fffM2 and M1 memories
0x180000-0x1bffffIPBus1 registers
0x1c0000-0x1cffff   Reserved
0x1d0000-0x1d0cffSystem registers
0x1d0d00-0x1edfff   Reserved
0x1ee000-0x1effff   IPBus2 registers
0x1f0000-0x1fffff   External memory access offset
*/
typedef struct stMsc812xDSIHost_RegMemMap
{
    VUByte            avucDSIM2Mem[0x76fff];
    VUByte            avucReserved1[0x9001];
    struct M1Core
    {
   VUByte         avucM1Mem[0x37fff];
   VUByte         avucReserved2[0x8001];
    } astDSICore[4];
    stMsc812xIPBus1   DSIIPBus1Regs;
    stMsc812xSysRegs  DSISystemRegs;
    VUByte            avucReserved4[0xf300];
    stMsc812xIPBus2   DSIIPBus2Regs;
} stMsc812xDSIHost_RegMemMap;

#endif __DSIHOSTMAP_H
```

**NOTES:**

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations not listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com



Document Order No.: AN2392
Rev. 1
8/2005