DP83815,DP83816

AN-1351 MAC Address Programming for DP83816 MacPHYTER-II and DP83815 MacPHYTER

Literature Number: ZHCA168

DP83816 MacPHYTER™II 和DP83815 MacPHYTER™ 的MAC地址编程

美国国家半导体
应用注释 1351
2004年11月

1.0 目的

本应用注释解释了什么是MAC (介质访问控制) 地址，如何得到MAC地址，以及如何在DP83816 MacPHYTER™II 和 DP83815 MacPHYTER™器件中编程MAC地址。为了对地址编程，本文讨论了两种可行的方法。但首先要定义何谓MAC地址。

2.0 MAC 地址定义

MAC (介质访问控制)地址是连到网络的设备的唯一的硬件或物理地址。MAC地址唯一地确定了网络上的每个结点。特别地，在以太网上MAC地址被称之为以太网地址，即电脑中以太网设备的唯一标识(ID)串联编号。MAC地址用于计算机之间通信用的局地网(LAN)中。

更一般地，MAC地址是标准化的数据链路层(第二层)地址，这是每个链接到LAN的端口或设备都必需的。网络中的其它设备采用这些地址来定位网络中的具体端口，并创建和升级路由表和数据结构。

MAC地址为6个字节长，并由IEEE控制。它们由48位的十六进制(hex)编号(12 个字符)组成，格式如XX-XX-XX-XX-XX-XX，其中X's 是从A到F的任意数字或字母。

地址的24个最高有效位也被称做"组织的唯一标识(OUI)"，或"供应商标识"。当请求一个MAC地址时，大多数供应商都有一个被赋予的固定的唯一OUI。这部分与另外24位连接以形成一个唯一的MAC地址，亦称之为硬件地址，MAC层地址或物理地址。

美国国家半导体公司(NSC)分配到的24位OUI以16进制表示为:

08-00-17

美国国家半导体公司产品的一个MAC地址实例以16进制表示为:

08-00-17-0B-62-35

3.0 获取一个MAC地址

若您的机构欲得到MAC地址，请参考下列IEEE链接：

http://standards.ieee.org/regauth/oui/forms/

http://standards.ieee.org/regauth/oui/index.shtml

4.0 MAC地址的编程

在DP83816和DP83815上，可用两种方法之一种将MAC地址编程到器件上。

在第一种方法中，首先将MAC地址编入EEPROM，然后加载到器件。在第二种方法中，采用RFCR (接收滤波器/匹配控制寄存器-偏移48h) 和RFDR (接收滤波器/匹配数据寄存器 - 偏移4Ch) 寄存器将MAC地址直接编程到器件中。

具体采用哪一种方法取决于器件的目标应用（参见第5.0节）。

4.1 使用EEPROM

第一种方法包括将MAC地址转换为一个特定格式，使之可从EEPROM下载到器件上。在转换MAC地址之后，可将它编程到EEPROM的相应寄存器位上，如表1所示。当EEPROM加载发生时，读取地址并放到器件的完全匹配寄存器(PMATCH)中，如表1所示。

表1.

EEPROM 寄存器地址 / bit(s)	PMATCH bit(s)
0006h / bit [0]	PMATCH [0]
0007h / bits [15:0]	PMATCH [1:16]
0008h / bits [15:0]	PMATCH [17:32]
0009h / bits [15:1]	PMATCH [33:47]

当器件上电时会自动进行EEPROM加载。此外，通过设定PTSCR (PCI 测试控制寄存器 -偏移0Ch)的第2位，可以强制或者启动EEPROM的再加载,拟或手动加载。

第4.1.1节解释了如何进行转换。

4.1.1 MAC地址转换

考察下列用十六进制格式表示的MAC地址：

08-00-17-0B-62-35

其中字节(Octet)0 或最高有效位(MSB)为08h，字节(Octet)5 或最低有效位(LSB)为35h，如表2所示。

表2.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5
08	00	17	0B	62	35

表3. 表示八进制的MAC地址转化为二进制，并列出了每一位的数值。

表3.

Octet	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Octet 0	0	0	0	0	1	0	0	0
Octet 1	0	0	0	0	0	0	0	0
Octet 2	0	0	0	1	0	1	1	1
Octet 3	0	0	0	0	1	0	1	1
Octet 4	0	1	1	0	0	0	1	0
Octet 5	0	0	1	1	0	1	0	1

在表4中，在各自的八进制中交换位的方向，如下所示：

表4.

Octet	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Octet 0	0	0	0	1	0	0	0	0
Octet 1	0	0	0	0	0	0	0	0
Octet 2	1	1	1	0	1	0	0	0
Octet 3	1	1	0	1	0	0	0	0
Octet 4	0	1	0	0	0	1	1	0
Octet 5	1	0	1	0	1	1	0	0

最后, 所有的位都左移1位，并映射到PMATCH (PM)寄存器，如表5所示：

表5.

PM bits	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
[0]								0
[1:8]	0	0	1	0	0	0	0	0
[9:16]	0	0	0	0	0	0	0	1
[17:24]	1	1	0	1	0	0	0	1
[25:32]	1	0	1	0	0	0	0	0
[33:40]	1	0	0	0	1	1	0	1
[41:47]	0	1	0	1	1	0	0	

然后八进制被转化回十六进制格式，并分配到正确的EEPROM地址，如表6所示：

表6.

EEPROM 寄存器地址 / bits	PMATCH bit(s)	转化后的 MAC 地址
0006h / bit [0]	PMATCH [0]	0
0007h / bits [15:0]	PMATCH [1:16]	2001
0008h / bits [15:0]	PMATCH [17:32]	D1A0
0009h / bits [15:0] (假设 bit 0 = 0)	PMATCH [33:47]	8D58

通过下列链接可在线索取执行上述转换计算的excel表格。

http://www.national.com/feedback/newfeed.nsf/Techsupport?openform

或者，用美国国家半导体的DOS控件诊断工具对EEPROM寄存器进行编程，包括MAC地址，可以由下列网站下载得到：

http://www.national.com/appinfo/networks/files/diag_exec.zip

4.1.2 EEPROM 的编程

为了对EEPROM编程，可采用两种方法之一。可以选用EEPROM程序或采用DP83816 (或 DP83815)本身来进行编程。

在DP83816 (或 DP83815)中，MEAR (EEPROM 访问寄存器 - 偏移 08h)控制器件上的EEPROM引脚。MEAR[3] (EESEL: EEPROM 片选位)控制EESEL引脚的数值(引脚128)。MEAR[2] (EECLK: EEPROM 串行时钟位)控制EECLK引脚(引脚2)的数值。MEAR [1] (EEDO: EEPROM 数据输出位)返回EEDO引脚(引脚138)的当前状态。MEAR [0] (EEDI: EEPROM数据输入位)控制EEDI引脚的数值(引脚1)。

使用那些位的实例如下所示：

将00000008h写到 MEAR ** 片选高电平
将00000008h写到MEAR ** 时钟低电平且数据输入为 0
将0000000Ch写到MEAR ** 时钟高电平且数据输出为 0
将00000009h写到MEAR ** 时钟低电平且数据输入为1
将0000000Eh写到MEAR ** 时钟高电平且数据输出为1
将00000000h写到MEAR ** 片选低电平

请参考数据手册中的第4.2.3节，获得关于MEAR寄存器的更详细的信息。附录A包含C++代码的样板程序，说明如何通过器件访问EEPROM。下列网站链接提供了诊断工具代码的子集：

http://www.national.com/appinfo/networks/macphyter2.html

在 "DOS 控件诊断(目标和源文件)"下方。

4.1.3 参考 EEPROM

请使用DP83815和DP83816数据手册中提到的FM93C46 EEPROM或者类似的器件。这种EEPROM具有MICROWIRE™ 同步总线接口，并采用一套特殊的指令组与器件进行通信。

4.2 采用 RFCR 和 RFDR

第二种方法采用RFCR和RFDR寄存器将MAC地址写入如下实例所示的器件中：

1) 考虑采用十六进制表示的相同MAC地址：

08-00-17-0B-62-35

其中字节0或最高有效位(MSB)为08h，字节5或者最低有效位(LSB)为35h，如表2所示。

表7.

Octet 0	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5
08	00	17	0B	62	35

2) 为了将地址编程到PMATCH寄存器:

- 将0000h 写到 RFCR (偏移48h) (为字节1-0选择合适的内部接收滤波器)
- 将0008h (字节 1-0) 写到RFDR (偏移4Ch)
- 将0002h写到RFCR (偏移48h) (为字节3-2选择合适的内部接收滤波器)
- 将0B17h (字节3-2) 写到RFDR (偏移4Ch)
- 将0004h写到RFCR (偏移48h) (为字节5-4选择合适的内部接收滤波器)
- 将3562h (字节5-4) 写到RFDR (偏移4Ch)

3) 为了从器件中读取地址:

- 将0000h写到RFCR (偏移48h)
- 读取RFDR (偏移4Ch), 这将提供MAC地址的字节1-0 (翻转它们以读取字节0-1)
- 将0002h 写到RFCR (偏移48h)
- 读取 RFDR (偏移4Ch), 这将提供MAC地址的字节3-2 (翻转它们以读取字节 2-3)
- 将0004h写到RFCR (偏移48h)
- 读取 RFDR (偏移4Ch), 这将提供MAC地址的字节5-4 (翻转它们以读取字节 4-5)

注意到读取的MAC地址如下所示：

00-08-0B-17-35-62

必须正确地翻转每组字节，因而得到的实际地址为08-00-17-0B-62-35。

由于RFCR也控制接收滤波器，一旦完成MAC地址的编程，需要用正确的设定值对RFCR重新配置，使得接收工作正常。

5.0 总结

第一种方法采用EEPROM，主要针对使用普通的、非唯一的软件的大多数应用场合，通常不需要任何软件修改。

第二种方法采用RFCR和RFDR寄存器，用于每个器件都需要唯一的软件代码的应用或系统。

因为IEEE标准要求每个单结点都具有唯一的MAC地址，所以由应用来限定所用的最佳编程方法。

附录 A

```
/**** **** **** **** *** **** **** *** **** **** *** **** **** *** **** **** ****
*
* EEP ROM.C
*
* 这是用于 MacPhyte 芯片与EEPROM
* 进行接口的程序
*
*** **** *** **** **** **** *** **** **** *** **** **** *** **** **** **** **** */

// Include files

#include <dos.h>
#include <time.h>
#include <sys\timeb.h>
#include <stdio.h>
#include <conio.h>
#include <memory.h>
#include <time.h>
#include "GLOBAL.H"
#include "NICSTUFF.H"
#include "PCIBIOS.H"

// Function prototypes

#include "FUNCTION.H"

UINT16 eepromDefaultValue[11] = {
                0xD008,         // 0000h, Subsystem Vendor ID
                0x0400,         // 0001h, Subsystem Device ID
                0x2CD0,         // 0002h, Minimum Grant, Maximum Latency
                0xCF82,         // 0003h, Configuration/Power Management Stuff
                0x0000,         // 0004h, SecureOn Password
                0x0000,         // 0005h, SecureOn Password
                0x0000,         // 0006h, SecureOn Password
                0x0000,         // 0007h, Ethernet ID
                0x0000,         // 0008h, Ethernet ID
                0x0000,         // 0009h, Ethernet ID
                0xA098          // 000Ah, WOL, Receive Filter/Match
};
```

```
UINT32 OUID; // Organizationally Unique Identifier
UINT32 startMACAddr;
UINT32 endMACAddr;
UINT32 currentMACAddr;

// Globals
extern unsigned short RegBase;
extern unsigned short batchMode;

extern UINT16 flip_16 (UINT16 us);

/**********************************************************************
*       功能: 显示 EEPROM
*       目的: 显示 EEPROM 内容
*       返回值: 空
**********************************************************************/
void DisplayEEPROM()
{
        int i=0;
        WORD value;

        for( i=0; i<0xC; i++ )
        {
                ReadEEWord( RegBase + MEAR, i, &value );

                printf( "\n%01X= %04X  ", i, value );
        }

        printf( "\n\n");
}

/**********************************************************************
*       功能: 读MAC地址
*       目的: 加载MAC地址的可读格式
*       返回值: 空
**********************************************************************/
void ReadMACAddress(  UINT16 *nicAddress )
{
        UINT32 i;
        UINT16 mask;
```

```
        UINT16 word1 = 0;
        UINT16 word2 = 0;
        UINT16 word3 = 0;
        UINT16 word4 = 0;

//
// 从 EEProm 中读取16位的字6到字9。
// 它们包含使用相当加密格式的硬件的MAC地址。
//
ReadEEWord( RegBase + MEAR, 0x06, &word1 );
ReadEEWord( RegBase + MEAR, 0x07, &word2 );
ReadEEWord( RegBase + MEAR, 0x08, &word3 );
ReadEEWord( RegBase + MEAR, 0x09, &word4 );

//
// 将加密格式解码为我们每次可使用一个字
//
nicAddress[ 0 ] = word1 & 1;
nicAddress[ 1 ] = word2 & 1;
nicAddress[ 2 ] = word3 & 1;

i = 15;
mask = 0x2;
while ( i-- )
        {
                if ( word2 & 0x8000 )
                {
                        nicAddress[ 0 ] |= mask;
                }
                word2 = word2 << 1;
                mask = mask << 1;
}

i = 15;
mask = 0x2;
        while ( i-- )
        {
                if ( word3 & 0x8000 )
                {
                        nicAddress[ 1 ] |= mask;
                }
```

```
                word3 = word3 << 1;
                mask = mask << 1;
    }

    i = 15;
    mask = 0x2;
    while ( i-- )
        {
                if ( word4 & 0x8000 )
                {
                        nicAddress[ 2 ] |= mask;
                }
                word4 = word4 << 1;
                mask = mask << 1;
    }
}

/***** ***** ***** ***** *** ***** ***** *** ***** ***** *** ***** ***
*       功能: 写 MAC 地址
*       目的: 将MAC地址的可读形式写入EEPROM
*       返回值: 如果失败为1, 如果成功为0
*** ***** *** ***** ***** ***** *** ***** ***** *** ***** ***** *** **/
int writeMacAddress( unsigned *nicAddress )
{
        UINT32      mask;
        UINT32      l;
        int         nbits = 9;
        unsigned    value, sixOff, nineOff, checkSum = 0, word1, word2, word3;
        unsigned    sword[4];

    ReadEEWord( RegBase + MEAR, 0x06, &sixOff );
        sixOff &= ~1;

    ReadEEWord( RegBase + MEAR, 0x09, &nineOff );
        nineOff &= 1;

    word1 = nicAddress[ 0 ];
    word2 = nicAddress[ 1 ];
    word3 = nicAddress[ 2 ];

    writeEeEnable();
```

```
        value = sixOff;
        if( word1 & 1 ) value |= 1;
sword[0] = value;
WriteEEWord( RegBase + MEAR, 0x06, value );

        value = 0;
l = 15;
mask = 0x2;
        while( l-- )
        {
                if( word1 & 0x8000 ) value |= mask;
                mask <<= 1;
                word1 <<= 1;
        }

if( word2 & 1 ) value |= 1;
sword[1] = value;
WriteEEWord( RegBase + MEAR, 0x07, value );

        value = 0;
l = 15;
mask = 0x2;
        while( l-- )
        {
                if( word2 & 0x8000 ) value |= mask;
                mask <<= 1;
                word2 <<= 1;
        }

if( word3 & 1 ) value |= 1;
sword[2] = value;
WriteEEWord( RegBase + MEAR, 0x08, value );

value = nineOff;
l = 15;
mask = 0x2;
        while( l-- )
        {
                if( word3 & 0x8000 ) value |= mask;
                mask <<= 1;
```

```
                word3 <<= 1;
        }

    sword[3] = value;
    WriteEEWord( RegBase + MEAR, 0x09, value );

    writeEeDisable();

    for( l=0; l<4; l++ )
    {
        ReadEEWord( RegBase + MEAR, 6+l, &value );
                if( value != sword[l] )
                {
                        printf( "\nEEPROM programming failed!  Enter Alternate MAC address.\n" );
                        break;
                }
    }

    if (doEeCheckSum())
        {
                return 1;
        }

        if (batchMode)
        {
                return writeMacFile();
        }

        return 0;
}

int writeID( int ssId, BYTE venDev )
{
        int idOffset = 0;

        // 计算偏差值：
        if (venDev == 'D') idOffset = 1;

    writeEeEnable();
    WriteEEWord( RegBase + MEAR, idOffset, ssId );
    writeEeDisable();
```

```
        return doEeCheckSum();
}

/***** ***** ***** ***** *** ***** ***** *** ***** ***** *** ***** ***
*       功能: 写 EEWord
*       目的: 将一个16位数值写入指定的EEPROM位置
*       返回值: 空
*** ***** *** ***** ***** ***** *** ***** ***** *** ***** ***** *** **/
void WriteEEWord( unsigned short pMear, UINT32 wordOffset, UINT16 value )
{
        UINT32 mask;
        UINT32 l;
        int     nbits = 9;
        UINT32 invalue = (UINT32) value;

    outpd( RegBase + MEAR, 0 );         /* clock out CS low */
    mdelay( 5 );

    outpd( RegBase + MEAR, EECLK );
    mdelay( 5 );

    invalue = ((UINT32)( 0x0140|wordOffset) << 16 ) | invalue;

    nbits = 25;

    mask = ((UINT32)1) << (nbits-1);

    while (nbits--)
        {
      /* assert chip select, and setup data in */
      l = ( invalue & mask ) ? EECS | EEDI : EECS;

      outpd( RegBase + MEAR, l );
          mdelay( 5 );

      outpd( RegBase + MEAR, l | EECLK );
      mdelay( 5 );

      mask >>= 1;
    }
```

```
    // IOW32( mear, 0 );          /*  终止写操作  */
    outpd( RegBase + MEAR, 0 );              // 工作终止
    mdelay( 5 );

    /* 等待工作完成 */
    outpd( RegBase + MEAR, EE CS );        /* assert CS */
    mdelay( 5 );

    outpd( RegBase + MEAR, EEC S | EECLK );
    mdelay( 5 );

    for (I = 0; I < 10; I++)
            {
       mdelay( 5 );
       if( inpd( RegBase + MEAR ) & EEDO )
          break;
    }

    outpd( RegBase + MEAR, 0 );          /* clock outCS low */
    mdelay( 5 );

    outpd( RegBase + MEAR, EEC LK );
    mdelay( 5 );
}

/**** **** **** **** *** **** **** *** **** **** *** **** ***
*          功能: 检查求和检验
*          目的: 采用MacPhyter 上的EEBIST 位使EEPROM求和检验生效
*          返回值: 如果失败为1, 如果成功为0
*** **** *** **** **** **** *** **** **** *** **** **** *** **/
int checkSumCheck()
{
          unsigned long pciTestReg = EEBIST _EN;

          outpd( RegBase + PCITEST_CNTRL,  EEBIST_E N );

          while (pciTestReg & EEB IST_EN)
          {
                    pciTestReg = inpd( RegBase + PCITEST _CNTRL );
```

```
        }

        if (inpd( RegBase + PCITEST_CNTRL )  & EEBIST_FAIL)
        {
                printf( "\n!!WARNING:  Checksum FAILED!\n" );
                return( 1 );
        }
        else
        {
                printf( "\nEEBIST Test Passes!\n" );
        }

        return( 0 );
}

/*********************************************************************
*       功能: 读 EEWord
*       目的: 从指定的位置读取16位的数值
*       返回值: 空
*********************************************************************/
void ReadEEWord( unsigned short pMear, UINT32 offset, UINT16* pEeData )
{
        UINT16 value;

   outpd( RegBase + MEAR, 0 );
   mdelay( 5 );

   outpd( RegBase + MEAR, EECLK );       // clock outno chipselect
   mdelay( 5 );

   eeput( (UINT32)(EEread|offset), 9 );

   value = eeget( 16 );

   outpd( RegBase + MEAR, 0 );          // terminate read
   mdelay( 5 );

   outpd( RegBase + MEAR, EECLK );       // clock outno chipselect
   mdelay( 5 );

   *pEeData = ( UINT16 )value;
```

```
    outpd( RegBase + MEAR, 0 );          // terminate read
    mdelay( 5 );
}

/********** ********** ********** ********** ********** ********** **********
*           功能: EEPUT
*           目的: 用时钟输出数据到EEPROM
*           返回值: 空
********** ********** ********** ********** ********** ********** **********/
void eeput( UINT32 value, int nbits )
{
            UINT32 mask = ((UINT32)1) <<  (nbits-1);
            UINT32 l;

    while( nbits-- )
            {
        // 断言片选，并设置数据输入
        l = ( value & mask ) ? EE CS | EEDI :  EECS;
        outpd(RegBase + MEAR, l );
        mdelay( 5 );

        outpd( RegBase + MEAR, l | EECLK );
        mdelay( 5 );

        mask >>= 1;
    }
}

/********** ********** ********** ********** ********** ********** **********
*           功能: EEPUT
*           目的: 用时钟由EEPROM输入数据
*           返回值: 空
********** ********** ********** ********** ********** ********** **********/
UINT16 eeget( int nbits )
{
            UINT16 mask = 1 << (nbits-1);
            UINT16 value = 0;

    while( nbits-- )
            {
```

```
    // 断言片选，以及用时钟传入数据
    outpd(RegBase + MEAR, EECS );
    mdelay( 5 );

    outpd( RegBase + MEAR, EECS | EECLK );
          mdelay( 5 );

    if( inpd( RegBase + MEAR ) & EEDO )
         value |= mask;
          mdelay( 5 );

    mask >>= 1;
  }

  return( value );
}

/***** ***** ***** ***** *** ***** ***** *** ***** ***** *** ***** ***
*         功能: 写 EeEnable
*         目的: 激活EEPROM 写模式
*         返回值: 空
*** ***** *** ***** ***** ***** *** ***** ***** *** ***** ***** *** **/
void writeEeEnable()
{
         UINT32 mask;
         UINT32 I;
         int       nbits = 9;
  //
  // 采用片选0保持4微秒以启动序列
  //
  outpd( RegBase + MEAR, 0 );
  mdelay( 5 );

  outpd( RegBase + MEAR, EEC LK );
  mdelay( 5 );

  mask = ((UINT32)1) << (nbit s-1);

  while( nbits-- )
        {
    /* 断言片选，并设置数据输入 */
```

```
        l = ( 0x0130 & mask ) ? EECS | EEDI : EECS;
        outpd(RegBase + MEAR, l );
        outpd(RegBase + MEAR, l | EECLK );
        mask >>= 1;
    }

    outpd( RegBase + MEAR, 0 );              // terminate operation
    mdelay( 5 );

    //lOW32( mear, 0 );          /* clock outCS low */
    outpd( RegBase + MEAR, EEC LK );
    mdelay( 5 );
}

/********** ********** ********** ********** ********** ********** ********** ********** ***
*           功能: 写 EEDisable
*           目的: 禁止EEPROM写模式
*           返回值: 空
********** ********** ********** ********** ********** ********** ********** ********** **/
void writeEeDisable()
{
            UINT32 mask;
            UINT32 l;
            int         nbits = 9;

    //ed->eeput( (uint32)EEwrite Disable, 9 );
    nbits = 9;
    mask = ((UINT32)1) << (nbit s-1);

    while( nbits-- )
            {
        /* 断言片选，并设置数据输入 */
        l = ( 0x0100 & mask ) ? EECS | EEDI : EECS;
        outpd( RegBase + MEAR, l );
        outpd( RegBase + MEAR, l | EECLK );
        mask >>= 1;
    }

    outpd( RegBase + MEAR, 0 );              // terminate operation
}
```

```
/**** **** **** **** *** **** **** *** **** **** *** **** ***
*       功能: 执行   EeCheckSum()
*       目的: 读取   EEPROM 内容, 计算求和校验,
*       写入, 验证写入值并使其生效。
*       返回值: 失败为1，成功为0
*** **** *** **** **** **** *** **** **** *** **** **** **** ***/
int doEeCheckSum()
{
        int I;
        WORD value, checkSum;
        unsigned char csLow, csHigh, csSum, csVals[ 11 ];

  // 计算求和校验：
  for( I=0; I<11; I++ )
  {
        ReadEEWord( RegBase + MEAR, I, &value );

        csLow = value & 0xff;
                value >>= 8;
        csHigh = value & 0xff;

        csVals[ I ] = csLow + csHigh;
  }

  // 计算求和校验：
  csSum =  0;
  for( I=0; I<11; I++ )
        {
                csSum += csVals[ I ];
        }

  csSum += 0x55;
  checkSum = (~csSum) + 1;
  checkSum = ( checkSum << 8 ) + 0x55;

  writeEeEnable();
  WriteEEWord( RegBase + MEAR, 0x0B, checkSum );
  writeEeDisable();

  ReadEEWord( RegBase + MEAR, 0xB, &value );
  if (value != checkSum)
```

```
        {
                printf( "\n!!WARNING Checksum not programmed correctly!\n" );
                printf( "\n  checkSUM: %04X actual: %04X", checkSum, value );
        }

    if (checkSumCheck()) // Invalid checksum
        {
                return 1;
        }

        outpd( RegBase + PCITEST_ CNTRL, EELOAD_EN ); //  Reload configuration information from EEPROM
        delay( 1 );

        return 0;
}

/********** ********** ********** ********** ********** ********** ********** ********** ********** **********
*       功能: 写  EeDefaults:
*       目的: 无需接触MAC地址即启动EEPROM
*       返回值: 失败为1，成功为0
*
*     下列来自 EPMON原程序:
*
*     eeprom  0     d008        # subsystem vendor
*     eeprom  1     0400        # subsys id
*     eeprom  2     2cd0        # min gnt, max lat
*     eeprom  3     cf82        # config stuff
*     eeprom  4     0000        # secure on pw stuff
*     eeprom  5     0000        # secure on pw stuff
*     eeprom  6     0000        # LSb is part of Ethernet ID
*     eeprom  7     2001        # Ethernet ID
*     eeprom  8     d1a1        # Ethernet ID
*     eeprom  9     $Loc9       # Ethernet ID
*     eeprom  a     a098
*     eeprom  b     $value      # checksum
********** ********** ********** ********** ********** ********** ********** ********** ********** **********/
int writeEeDefaults()
{
        FILE *stream;
```

```
        char buf[120];
        UINT16 eepromValue[11];
        UINT32 i;
        UINT16 value;
        unsigned char inputString[25];
        int nicAddress[3];

        UINT16 Addressbit;

        // 如果文件存在，从文件中读取EEPROM数值，
        // 否则设置为默认值
if ((stream = fopen("EEPROM.TXT", "rt")) != NULL)
        {
                for (i = 0; i <= 0xA; i++)
                {
                        fgets(buf, sizeof(buf), stream);
                        sscanf(buf, "%04X", &eepromValue[i]);
                }
                fclose(stream);
        }
        else
        {
                for (i = 0; i <= 0xA; i++)
                {
                        eepromValue[i] = eepromDefaultValue[i];
                }
        }

        // 如果文件存在，即从文件中读取子系统的ID，
        // 并将ID转化为DP83815/816采用的可读格式
if ((stream = fopen("SUBSYSID.TXT", "rt")) != NULL)
        {
                for (i = 0; i <= 1; i++)
                {
                        fgets(buf, sizeof(buf), stream);
                        sscanf(buf, "%04X", &value);
                        eepromValue[i] = flip_16(value);
                }
                fclose(stream);
        }
```

```
writeEeEnable();

        // 写入 EEPROM
        for (i = 0; i <= 0xA; i++)
        {
                if (i <=5 || i == 0xA)
                {
                        WriteEEWord( RegBase + MEAR, i, eepromValue[i] );
                }
        }

// 清除MAC地址的最低有效位:
ReadEEWord( RegBase + MEAR, 0x6, &Addressbit );
Addressbit &= 1;
WriteEEWord( RegBase + MEAR, 0x6, Addressbit );

 // 清除MAC地址的最低有效位:
ReadEEWord( RegBase + MEAR, 0x9, &Addressbit );
Addressbit &= 0xfffe;
WriteEEWord( RegBase + MEAR, 0x9, Addressbit );

writeEeDisable();

        //检查EEPROM 值是否写入正确
        for (i = 0; i <= 0xA; i++)
        {
                if (i <= 5 || i == 0xA)
                {
                        ReadEEWord( RegBase + MEAR, i, &value );
                        if (value != eepromValue[i])
                        {
                                printf("EEPROM values  are not written correctly!\n");
                                return 1;
                        }
                }
        }

        // 如果MAC地址在批处理模式而且MAC地址文件存在
        // 则写MAC地址
        if (batchMode)
        {
```

```
                if (readMacFile(inputString))
                {
                        return 1;
                }
                nicAddress[0] = (UINT16)*( inputString+2 );
                nicAddress[0] = (nicAddress[0] << 8) + (UINT16)*( inputString );
                nicAddress[1] = (UINT16)*( inputString+6 );
                nicAddress[1] = (nicAddress[1] << 8) + (UINT16)*( inputString+4 );
                nicAddress[2] = (UINT16)*( inputString+10 );
                nicAddress[2] = (nicAddress[2] << 8) + (UINT16)*( inputString+8 );
                if (writeMacAddress(nicAddress))
                {
                        return 1;
                }

                // 再次调用 findNIC 程序以显示新地址
                if (findNIC(TRUE))
                {
                        return 1;
                }

                if (currentMACAddr == endMACAddr)
                {
                        return 1;
                }

                return 0;
        }
        else
        {
                return doEeCheckSum();
        }
}

/********** ********** ********** ********** ********** ********** **********
*       功能: 读Mac文件
*       目的: 打开文件, 找回Mac地址,
*       递增Mac地址并返回新数值。
*       返回值: 1 = 失败; 0 =成功;加载输入;
********** ********** ********** ********** ********** ********** **********/
int readMacFile( unsigned char *inputString )
```

```
{
        FILE *stream;
        char buf[120];

        // 打开cwd 中的文件 MACADDR.TXT
  if ((strea m = fopen("MACADDR.TXT", "rt")) != NULL)
        {
                fgets(buf, sizeof(buf), stream);
                sscanf(buf, "%06lX%06lX", &OUID, &startMACAddr);
                fgets(buf, sizeof(buf), stream);
                sscanf(buf, "%06lX%06lX", &OUID, &endMACAddr);
                fgets(buf, sizeof(buf), stream);
                sscanf(buf, "%06lX%06lX", &OUID, &currentMACAddr);
                fclose(stream);
        }
        else
        {
           printf("\nUnable to open up MACADDR.TXT in current working directory.");
           return 1;
        }

        // 检查MAC地址是否有效
        if (((OUID == 0x00 0000) && (c urrentMACAddr == 0x000000))
                || ((OUI D == 0xFFFFFF) &&  (currentMACAddr == 0xFFFFFF)))
        {
           printf("\nInvalid MAC address: %06lX%06lX", OUID, currentMACAddr);
                return 1;
        }
        else if ((currentMACAddr < startMACAddr) || (currentMACAddr > endMACAddr))
        {
           printf("\nMAC address is out of the range!");
                return 1;
        }

        sscanf(buf, "%02X%02X%02X%02X%02X%02X",
         inputString,
         (inputString+2),
         (inputString+4),
         (inputString+6),
         (inputString+8),
         (inputString+10)
```

```
        );

        return 0;
}


/*********************************************************************
*       功能: 写Mac文件
*       目的: 打开一个文件, 写入新数值。
*       返回值: 1 =失败; 0 = 成功; 加载此输入;
*********************************************************************/
int writeMacFile(void)
{
        FILE *stream;

        // 如果当前的MAC地址达到最终的MAC地址，则停止运行程序
        if (currentMACAddr < endMACAddr)
        {
                currentMACAddr++;

                // Open up MACADDR.TXT file in the cwd
                if ((stream = fopen("MACADDR.TXT", "wt")) != NULL)
                {
                        fprintf(stream, "%06lX%06lX\t\t// Start MAC Address\n", OUID, startMACAddr);
                        fprintf(stream, "%06lX%06lX\t\t//End MAC Address\n", OUID, endMACAddr);
                        fprintf(stream, "%06lX%06lX\t\t//Current MAC Address\n", OUID, currentMACAddr);
                        fclose(stream);
                }
                else
                {
                        printf("\nUnable to open up MACADDR.TXT in current working directory.");
                        return 1;
                }
        }
        else
        {
           printf("\nEnd MAC address is reached.\n");
        }

        return 0;
}
```

对于上述任何电路的使用，美国国家半导体公司不承担任何责任且不默示任何电路专利许可。美国国家半导体公司保留随时更改上述电路和规格的权利，恕不另行通知。
想了解最新的产品信息，请访问我们的网址：www.national.com。

生命支持策略

未经美国国家半导体公司的总裁和首席律师的明确书面审批，不得将美国国家半导体公司的产品作为生命支持设备或系统中的关键部件使用。特此说明：

1. 生命支持设备/系统指：（a）打算通过外科手术移植到体内的生命支持设备或系统；（b）支持或维持生命，依照使用说明书正确使用时，有理由认为其失效会造成用户严重伤害。
2. 关键部件是在生命支持设备或系统中，有理由认为其失效会造成生命支持设备/系统失效，或影响生命支持设备/系统的安全性或效力的任何部件。

禁用物质合规

美国国家半导体公司制造的产品和使用的包装材料符合《消费产品管理规范（CSP-9-111C2）》以及《相关禁用物质和材料规范（CSP-9-111S2）》的条款，不包含CSP-9-111S2限定的任何"禁用物质"。
无铅产品符合RoHS指令。

National Semiconductor
Americas Customer
Support Center
Email: new.feedback@nsc.com
Tel: 1-800-272-9959

www.national.com

National Semiconductor
Europe Customer Support Center
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor
Asia Pacific Customer
Support Center
Email: ap.support@nsc.com

National Semiconductor
Japan Customer Support Center
Fax: 81-3-5639-7507
Email: jpn.feedback@nsc.com
Tel: 81-3-5639-7560

重要声明

德州仪器(TI) 及其下属子公司有权在不事先通知的情况下, 随时对所提供的产品和服务进行更正、修改、增强、改进或其它更改，并有权随时中止提供任何产品和服务。客户在下订单前应获取最新的相关信息，并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的硬件产品的性能符合TI 标准保修的适用规范。仅在TI 保证的范围内，且TI 认为有必要时才会使用测试或其它质量控制技术。除非政府做出了硬性规定，否则没有必要对每种产品的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用TI 组件的产品和应用自行负责。为尽量减小与客户产品和应用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何TI 专利权、版权、屏蔽作品权或其它与使用了TI 产品或服务的组合设备、机器、流程相关的TI 知识产权中授予的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从TI 获得使用这些产品或服务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是TI 的专利权或其它知识产权方面的许可。

对于TI 的产品手册或数据表，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许进行复制。在复制信息的过程中对内容的篡改属于非法的、欺诈性商业行为。TI 对此类篡改过的文件不承担任何责任。

在转售TI 产品或服务时，如果存在对产品或服务参数的虚假陈述，则会失去相关TI 产品或服务的明示或暗示授权，且这是非法的、欺诈性商业行为。TI 对此类虚假陈述不承担任何责任。

TI 产品未获得用于关键的安全应用中的授权，例如生命支持应用（在该类应用中一旦TI 产品故障将预计造成重大的人员伤亡），除非各方官员已经达成了专门管控此类使用的协议。购买者的购买行为即表示，他们具备有关其应用安全以及规章衍生所需的所有专业技术和知识，并且认可和同意，尽管任何应用相关信息或支持仍可能由TI 提供，但他们将独力负责满足在关键安全应用中使用其产 品及TI 产品所需的所有法律、法规和安全相关要求。此外，购买者必须全额赔偿因在此类关键安全应用中使用TI 产品而对TI 及其 代表造成的损失。

TI 产品并非设计或专门用于军事/航空应用，以及环境方面的产品，除非TI 特别注明该产品属于“军用”或“增强型塑料”产品。只 有TI 指定的军用产品才满足军用规格。购买者认可并同意，对TI 未指定军用的产品进行军事方面的应用，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

TI 产品并非设计或专门用于汽车应用以及环境方面的产品，除非TI 特别注明该产品符合ISO/TS 16949 要求。购买者认可并同意，如果他们在汽车应用中使用任何未被指定的产品，TI 对未能满足应用所需要求不承担任何责任。

可访问以下URL 地址以获取有关其它TI 产品和应用解决方案的信息：

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP 机动性处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity		
	德州仪器在线技术支持社区	www.deyisupport.com	

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122