Engineer-to-Engineer Note EE-315

Technical notes on using Analog Devices DSPs, processors and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Changing the PHY in the Ethernet Driver for Blackfin® Processors

Contributed by Jiang Wu

Rev 1 – June 21, 2007

Introduction

This EE-Note describes how to achieve networking functionality with ADSP-BF536/ADSP-BF537 Blackfin® processors together with MII-compatible Ethernet Physical Layer Transceivers (PHYs) other than the SMSC LAN83C185, which is populated on the ADSP-BF537 EZ-KIT Lite® evaluation system. The procedure is based on the VisualDSP++® 4.5 development tool suite and its Ethernet MAC driver for the ADSP-BF537 processor. The work was verified on an SMSC LAN8187 and on a National Semiconductor DP83848.

Overview

ADSP-BF536/ADSP-BF537 Blackfin processors include a built-in Ethernet MAC controller, providing an IEEE 802.3-2002-compliant MII interface for easy connection to any MII-compatible PHY[1]. In addition, the VisualDSP++ 4.5 development tools are shipped with a driver for the MAC controller, a TCP/IP stack (`LwIP`), and a project template for TCP/IP applications[2]. This allows you to start network-capable applications right away by using the standard BSD socket API. This network solution is illustrated in Figure 1.

The MAC controller driver provided by the VisualDSP++ 4.5 tools is designed primarily for the SMSC LAN83C185 PHY device populated on the ADSP-BF537 EZ-KIT Lite evaluation board[3]. If you choose to use this development system, you can code network applications without having to consider any hardware details[4]. However, if you use other PHYs, you must modify the MAC driver, because the MAC driver uses unique vendor-specific features in addition to MII standard PHY features.

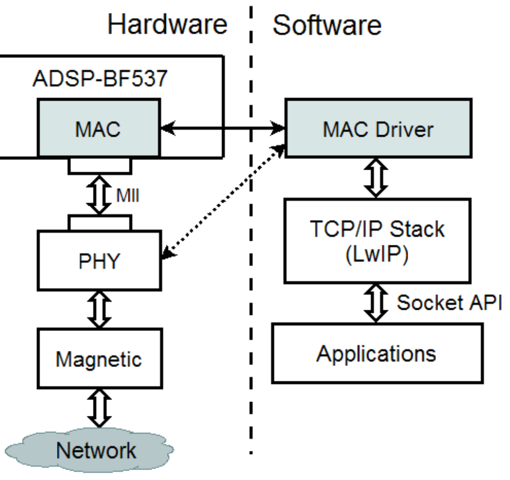


Figure 1. Overview of ADSP-BF537 based network applications

The source code of the MAC driver consists of three files.

The project file (`ADI_ETHER_BF537.dpj`) and the C source file (`ADI_ETHER_BF537.c`) are located in the following directory:

```
<install_path>\Blackfin\lib\src\drivers
\ethernet\ADI_ETHER_BF537\
```

The `ADI_ETHER_BF537.h` file is located in:

```
<install_path>\Blackfin\include\drivers
\ethernet\
```

Copyright 2007, Analog Devices, Inc. All rights reserved. Analog Devices assumes no responsibility for customer product design or the use or application of customers' products or for any infringements of patents or rights of others which may result from Analog Devices assistance. All trademarks and logos are property of their respective holders. Information furnished by Analog Devices applications and development tools engineers is believed to be accurate and reliable, however no responsibility is assumed by Analog Devices regarding technical accuracy and topicality of the content provided in Analog Devices Engineer-to-Engineer Notes.

These files are copied automatically when the VisualDSP++ 4.5 tools are installed. Opening the project file and building it with the VisualDSP++ 4.5 development tools generates a library file named `ADI_ETHER_BF537.dlb`, which can then be linked to the network application[5].

The MAC driver is the connecting bridge between the physical MAC controller/PHY and the software TCP/IP stack (`LwIP`). On one side, it uses two dedicated DMA channels (`DMA1` and `DMA2`) for the network data exchange between the MAC controller and the system memory. On the other side, it communicates with the TCP/IP stack through a message box mechanism.

The MAC controller talks to the PHY via the MII interface, which includes two parts, the network data exchange (signals `RxD[3:0]`, `Rx_DV`, `Rx_CLK`, and `Rx_ER` for receiving; and `TxD[3:0]`, `Tx_EN`, `Tx_CLK`, and `Tx_ER` for transmitting), and the management data exchange (signals `MDC` and `MDIO`). The driver controls the MAC controller by writing/reading its memory-mapped registers (MMRs). It also manages the PHY by commanding the MAC controller to write/read the 32 PHY registers via the MII management data exchange channel.

The first two of the 32 PHY registers, the Control Register (register 0) and the Status Register (register 1), are mandatory, according to the MII specification, while the following 13 registers (registers 2-15) are the optional extended register set. The remaining registers (registers 16-31) are vendor-specific.

In the following section, the required modifications to the MAC driver source code to accommodate different PHYs will be described.

Modifying the MAC Driver Code

To enable the MAC driver for the ADSP-BF536/ADSP-BF537 processor to work with different PHYs, you must first modify the `ADI_ETHER_BF537.c` source code file as described in the following sections.

Required changes to the source code are explained in RED font in the code shown in the following steps.

To make it easy to find the change locations, a modified version of `ADI_ETHER_BF537.c` is included in the `.ZIP` file associated with this EE-Note[6], in which all the change locations are marked with a comment `/*PHY_CHANGE_LOCATION*/`.

static u32 adi_pdd_Open()

This function determines all the default settings of the PHY, such as PHY address, full/half duplex mode, speed, and auto-negotiation. They can be set here or in the application code by calling the device control function `adi_dev_Control()` with the corresponding `COMMAND`. These settings must be done only after `adi_dev_Open()` and before `adi_dev_Control(…, ADI_ETHER_CMD_START, …)`.

```
dev->PhyAddr = 0x01;  // 0x01 1.1 EZ kit and BUB, Changed to reflect the new PHY's
           address. The PHY's address is used by the MAC's MII management interface
           to identify each PHY, since MII is able to manage up to 32 PHYs with the
           same interface. It is usually determined by the strap option pins of the
           PHYs. The values of these pins are sampled during PHY reset and are used
           to strap the device into specific addresses.
dev->CLKIN = 25;  //Ezkit, Changed to reflect the new application system's
           oscillator/crystal clock in unit of MHz. It will later be used for
           setting up the MDC frequency. According to the MII standard, the minimum
           high and low times for the MDC signal shall be 160 ns each, and the
           minimum period for MDC shall be 400 ns. The driver uses this CLKIN value
           to set up a MDC clock of 2.5MHz.
dev->FullDuplex=false; //Changed to reflect the new PHY's Full Duplex capability
dev->Negotiate = true; //Changed to reflect the new PHY's AUTONEGOTIATION
           capability. If the PHY doesn't support it, the driver code must be
           changed to set the PHY's capability correctly, which includes dev-
           >FullDuplex and dev->Port10. A TRUE value of dev->Port10 means 10MBase,
           otherwise it is 100MBase.
```

static SetPhy()

This function is called when the application issues the `ADI_ETHER_CMD_START` command to the MAC driver by calling `adi_dev_Control` (…, `ADI_ETHER_CMD_START`, …). It initializes the PHY using the default parameters set in the `adi_pdd_Open()`function.

Set software reset

```
// issue a reset
RawWrPHYReg(dev->PhyAddr, PHYREG_MODECTL, 0x8000);
// wait half a second
period = 30000000; // assume 600 MHZ, This provides a pure delay to allow the PHY
           to complete a software RESET. It needs to be changed to reflect the new
           PHY's requirements and the application system's clock. The MII standard
           requires the RESET process be completed within 0.5 seconds from the
           issuing of a software reset.
ndtime = clock()+period;
while (clock()<ndtime);
```

Configure settling time

```
WrPHYReg(dev->PhyAddr, PHYREG_MODECTL, phydat);
period = 100000000; // assume 600 MHZ, Similar to step 2, it provides a pure delay
           to allow the new control settings to take effect, changed to reflect the
           new PHY's requirements
ndtime = clock()+period;
while (clock()<ndtime);
```

Check PHY ID

```
if ((RdPHYReg(dev->PhyAddr, PHYREG PHYID1) == 0x7) && ((RdPHYReg(dev->PhyAddr,
           PHYREG PHYID2)&0xfff0 ) == 0xC0A0)) { // check PHYID, It checks the ID
           of the PHY to ensure the connected PHY is LAN83C185(the PHY on the ADSP-
           BF537 EZ-Kit Lite evaluation platform) by reading the 2 PHYID registers
           of the PHY. It needs to be changed to reflect the new PHY's ID, which
           can be found in the datasheet of the PHY.
```

Enable PHY interrupt

```
WrPHYReg(dev->PhyAddr, 30, 0x0ff); // enable interrupt, It enables the interrupt of
           the PHY by writing a vendor-specific register (#30). It needs to be
           changed to new PHY's counterpart. The enabling of the PHY interrupt also
           requires the physical connection of the PHY's interrupt pin to the ADSP-
           BF537 processor's PHY INT pin. If the developer decides not to use the
           interrupt, this instruction can be commented out.
```

static ADI_INT_HANDLER(EtherInterruptHandler)

This function is the interrupt service routine (ISR) for the MAC controller interrupt. The interrupt may come from the PHY, the MAC management counter, the `Rx` frame status, the `Tx` frame status, and so on[1]. However, the current version of the driver processes only the PHY interrupt and the MAC management counter interrupt. The PHY interrupt is indicated by the assertion of the `INT` pin of the PHY and is sensed by the ADSP-BF537 processor at the `PHY_INT` pin if these two pins are connected. If you choose to use the PHY interrupt, you must provide your own service code in the `if(systat&0x01){…}` block in this function. At a minimum, the service code should read and clear the PHY interrupt status bits. Note that different PHYs usually have different interrupt structure and capability. The current MAC driver takes auto-negotiation mode by default, so it sets the full/half duplex mode and PAUSE capability according to its link partners' ability.

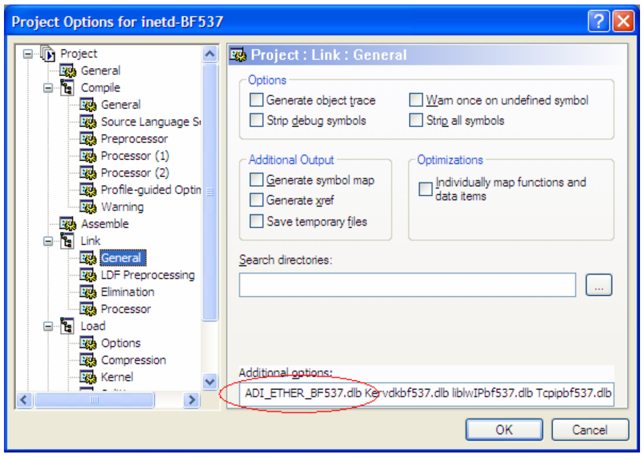
```
if (systat&0x01) {
    //PHY_INT
    int full=0;
    u32 opmode;
    u16 reg3,reg2,reg,phydat;
    reg2 = RdPHYReg(dev->PhyAddr,2); // read PHY id 1
    reg3 = RdPHYReg(dev->PhyAddr,3); // read PHY id 2
    if ((reg2 == 0x07) && ((reg3>>4) == 0xc0a)) { // Similar to step 3, check the ID
           of the PHY to ensure the connected PHY is LAN83C185 (the PHY on the
           ADSP-BF537 EZ-Kit Lite evaluation platform). Change to reflect the new
           PHY's ID.
         // SMSC LAN83C185
         reg = RdPHYReg(dev->PhyAddr,31); // read special status, this is to read
           the PHY's speed indication to determine whether full or half duplex is
           being used. Change this instruction and the next one to reflect the new
           PHY.
         full = (reg&0x10);
         if (full) {
              // does remote link support flow control
              phydat = RdPHYReg(dev->PhyAddr,PHYREG_ANLPAR);
              dev->FlowControl = (phydat &0x0400); // get the PAUSE capability of
                   the link partner and set the PAUSE option of the MAC controller
                   accordingly in the following instruction
              if (dev->FlowControl) {
                  // we enable flow control
                  *pEMAC_FLC = FLCE;            /* flow control enabled */
              // advertize flow control supported
              }
         }
    }
    opmode = *pEMAC_OPMODE;
    if (full) {
         opmode |= 0x04000000;
    } else {
         opmode &= 0xfbffffff;
    }
    *pEMAC OPMODE = opmode; // Set the Full/half duplex mode of the MAC controller
           according to the PHY's current mode

    systat = RdPHYReg(dev->PhyAddr,29); // read interrupt sources, read the source
           of the current PHY interrupt and clear it. Change to reflect the new
           PHY. The interrupt event and the status (systat) are to be passed to the
           interrupt callback function. However, the callback of the current
           version of the driver does not do any further processing.
    event = ADI_ETHER_EVENT_INTERRUPT_PHY;
    result = ADI_INT_RESULT_PROCESSED;
}
```

Once all the edits described above are made to the source code, rebuild the `ADI_ETHER_BF537.dpj` project and copy the generated driver library (`.dlb`) into:

`<install_path>\Blackfin\lib\`

Then, include the `.dlb` in the library list for the application project. The library list can be modified in the VisualDSP++ development tools via the `Project Options→Link` sub-tree in the `Additional options` dialog box, as shown in Figure 2.



Additional options: ADI_ETHER_BF537.dlb Kervdkbf537.dlb liblwIPbf537.dlb Tcpipbf537.dlb

Figure 2. Adding a library to the library list

Click `OK` to save the changes made to the `Project Options`, and your project is now set up to include the new driver.

Conclusion

Though the MAC controller driver provided by the VisualDSP++ 4.5 development tools is designed for the PHY on the ADSP-BF537 EZ-KIT Lite evaluation platform, it can be modified to work with other PHYs that are compatible with the MII standard. The modification involves changes to one of the source code files and rebuilding the MAC driver library. However, if you want to use of the extra unique features of the new PHY, you must make further modifications to the driver.

Appendix

Functions for PHY Register Access

These functions can be used for all PHYs.

```
static u16 RdPHYReg(u16 PHYAddr, u16 RegAddr)
static void RawWrPHYReg(u16 PHYAddr, u16 RegAddr, u32 Data)
static void WrPHYReg(u16 PHYAddr, u16 RegAddr, u32 Data)
```

Supported Control Commands

These control commands apply to all MII-compatible PHYs. The default settings are defined in the function `adi_pdd_Open()`. For new PHY drivers, developers can use control commands after the device is opened or use the `adi_pdd_Open()` function to configure the settings.

- `ADI_ETHER_CMD_BF537_CLKIN`
- `ADI_ETHER_CMD_BF537_SET_PHY_ADDR`
- `ADI_ETHER_CMD_SET_LOOPBACK`
- `ADI_ETHER_CMD_SET_NEGOTIATE`
- `ADI_ETHER_CMD_SET_FULL_DUPLEX`
- `ADI_ETHER_CMD_SET_SPEED`

MII PHY Registers Used by the Driver

The MAC driver uses the following PHY registers:

- Register 0 (basic): Control Register
- Register 1 (basic): Status Register
- Register 2 (extended): PHY ID 1 Register
- Register 3 (extended): PHY ID 2 Register
- Register 4 (extended): Auto-negotiation Advertisement Register
- Register 5 (extended): Auto-negotiation Link Partner Base Page Ability Register
- Register 29: Interrupt Source Flags
- Register 30: Interrupt Mask Register
- Register 31: PHY special Control/Status Register

In the MAC driver, the number of PHY registers and the register addresses are defined as macros (such as `NO_PHY_REGS`, `PHYREG_MODECTL`, and so on) at the top of `ADI_ETHER_BF537.c`. It is likely that these macros will be different for new PHYs. When developing new drivers, check the data sheet of the PHY to identify the equivalent registers, and change the code accordingly.

References

[1] *ADSP-BF537 Blackfin Processor Hardware Reference*. Rev 2.0, December 2005. Analog Devices, Inc.

[2] *LwIP User Guide* (in VisualDSP++ 4.5 package, <install_path>\Blackfin\lib\src\lwIP\docs\LWIP_UserGuide.doc). Analog Devices, Inc.

[3] *ADSP-BF537 EZ-KIT Lite Evaluation System Manual*. Rev 2.0, June 2006. Analog Devices, Inc.

[4] *Getting Started with ADSP-BF537 EZ-KIT Lite*. Rev 1.1, April 2006. Analog Devices, Inc.

[5] *VisualDSP++ 4.5 User's Guide*. Rev 2.0, April 2006. Analog Devices, Inc.

[6] Associated ZIP File for EE-315. Rev 1, May 31, 2007. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – June 21, 2007 by Jiang Wu	Initial release