



XAPP1042 (v1.0.1) May 2, 2008

## Reference System: Ethernet PHY Register Access With GPIO

Author: Brian Hill

### Abstract

The XPS Ethernetlite peripheral does not provide any mechanism to access the Ethernet PHY registers. These registers are used to configure auto negotiation parameters and to obtain PHY status. This application note provides reference systems and associated software to access PHY registers by connecting the serial management bus signals MDC and MDIO to GPIOs which the software controls directly.

### Included Systems

Included with this application note are two reference systems:

[www.xilinx.com/support/documentation/application\\_notes/xapp1042\\_ppc405.zip](http://www.xilinx.com/support/documentation/application_notes/xapp1042_ppc405.zip)

[www.xilinx.com/support/documentation/application\\_notes/xapp1042\\_mb.zip](http://www.xilinx.com/support/documentation/application_notes/xapp1042_mb.zip)

### Introduction

The modern Ethernet PHY is highly configurable. The devices used on the Xilinx boards pertinent to this application note are capable of 10MB, 100MB, or 1000MB operation at full duplex and half duplex. A PHY will, by default, auto negotiate to the highest possible link speed it has in common with the peer Ethernet PHY to which it is connected. This behavior is modified by accessing the appropriate PHY configuration registers.

Many Ethernet MACs will provide a mechanism for the software to access the registers of the PHY to which it is connected. Because the goal of the XPS Ethernetlite core is to provide a small, simple Ethernet controller, it does not provide access to the PHY registers. Furthermore, this MAC is capable of 10/100MB link speeds, not 1000MB. When XPS Ethernetlite is coupled to a gigabit PHY which is cabled to a PC with a gigabit NIC the auto negotiation defaults will select a link speed of 1000MB, a speed with which Ethernetlite is not compatible. For proper operation it is best to configure the PHY so that it will negotiate only 100MB links.

This application note provides a reference system where the PHY serial management interface signals (MDC, MDIO) are connected to an XPS GPIO peripheral. The PHY registers are accessed by the software directly manipulating this serial bus clock (MDC) and bidirectional data (MDIO). Software is provided to configure the onboard PHY so that it will not auto negotiate to 1000MB.

### Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx ML403 Development board (PPC405 system)
- Xilinx SP3ADSP-1800 Development board (MicroBlaze™ system)
- Xilinx Platform USB Cable or Parallel IV Cable
- RS232 Cable
- Serial Communications Utility Program (such as HyperTerminal)
- Xilinx Platform Studio 10.1i
- Xilinx Integrated Software Environment (ISE®) 10.1i

## ML403 Reference System Specifics

The included ML403 PPC405 reference system is shown in Figure 1. Note the connection of the XPS GPIO peripheral to the Ethernet PHY serial management bus. Refer to Table 1 for the processor address map.

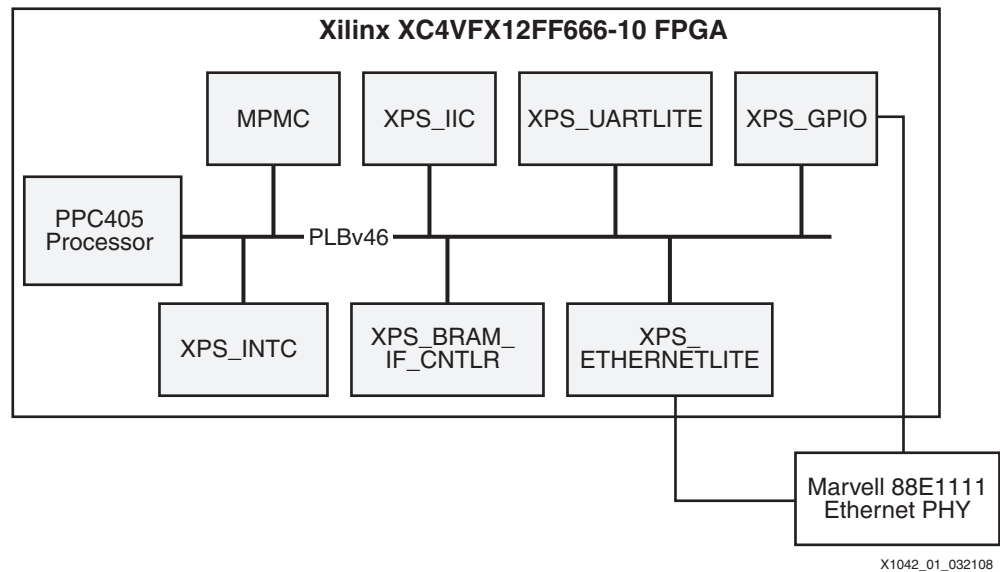


Figure 1: ML403 Reference System Block Diagram

## Address Map

Table 1: ML403 PPC405 Reference System Address Map

Instance	Peripheral	Base Address	High Address
DDR_SDRAM	mpmc	0x00000000	0x03FFFFFF
Ethernet_MAC	xps_ethernetlite	0x81000000	0x8100FFFF
MII_MDC_MDIO	xps_gpio	0x81400000	0x8140FFFF
IIC_EEPROM	xps_iic	0x81600000	0x8160FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
RS232_Uart	xps_uartlite	0x84000000	0x8400FFFF
xps_bram_if_cntlr_1	xps_bram_if_cntlr	0xFFFFE000	0xFFFFFFFF

The basic system, created with EDK Base System Builder, is modified slightly to include an xps\_gpio with ports connected to the PHY serial management bus as indicated in the excerpts below.

```
system.mhs:
PORT fpga_0_MII_MDC_MDIO_GPIO_IO_pin = fpga_0_MII_MDC_MDIO_GPIO_IO, DIR =
IO, VEC = [1:0]

BEGIN xps_gpio
PARAMETER INSTANCE = MII_MDC_MDIO
PARAMETER HW_VER = 1.00.a
PARAMETER C_GPIO_WIDTH = 2
PARAMETER C_BASEADDR = 0x81400000
PARAMETER C_HIGHADDR = 0x8140ffff
BUS_INTERFACE SPLB = plb
```

```

PORT GPIO_IO = fpga_0_MII_MDC_MDIO_GPIO_IO
END

```

```

data/system.ucf:
# MDC. For ML403, this is location D1
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<0> LOC = D1;
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<0> IOSTANDARD = LVCMOS25;

# MDIO. For ML403, this is location G4
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<1> LOC = G4;
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<1> IOSTANDARD = LVCMOS25;

```

## SP3ADSP-1800 Reference System Specifics

The included SP3ADSP-1800 MicroBlaze reference system is shown in [Figure 2](#). Note the connection of the XPS GPIO peripheral to the Ethernet PHY serial management bus. Refer to [Table 2](#) for the processor address map.

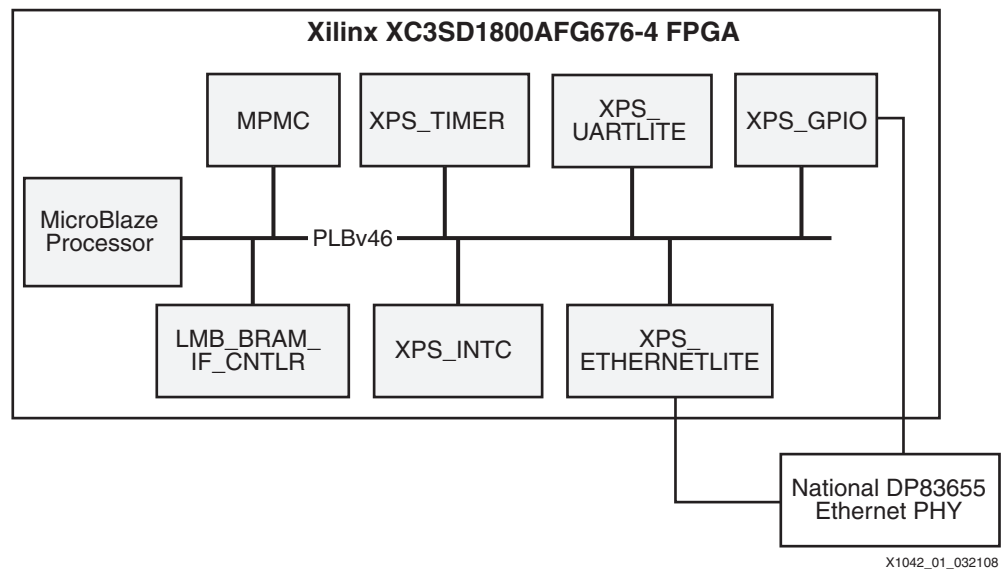


Figure 2: SP3ADSP-1800 Reference System Block Diagram

## Address Map

Table 2: SP3ADSP-1800 MicroBlaze Reference System Address Map

Instance	Peripheral	Base Address	High Address
DDR2_SDRAM	mpmc	0x00000000	0x03FFFFFF
Ethernet_MAC	xps_ethernetlite	0x81000000	0x8100FFFF
MII_MDC_MDIO	xps_gpio	0x81400000	0x8140FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
xps_timer_1	xps_timer	0x83C00000	0x83C0FFFF
RS232_Uart_1	xps_uartlite	0x84000000	0x8400FFFF
xps_bram_if_cntlr_1	xps_bram_if_cntlr	0xFFFFE000	0xFFFFFFFF

The basic system, created with EDK Base System Builder, is modified slightly to include an xps\_gpio with ports connected to the PHY serial management bus as indicated in the excerpts below.

```

system.mhs:
PORT fpga_0_MII_MDC_MDIO_GPIO_IO_pin = fpga_0_MII_MDC_MDIO_GPIO_IO, DIR =
IO, VEC = [2:0]

BEGIN xps_gpio
  PARAMETER INSTANCE = MII_MDC_MDIO
  PARAMETER HW_VER = 1.00.a
  PARAMETER C_GPIO_WIDTH = 3
  PARAMETER C_BASEADDR = 0x81400000
  PARAMETER C_HIGHADDR = 0x8140ffff
  BUS_INTERFACE SPLB = mb_plb
  PORT GPIO_IO = fpga_0_MII_MDC_MDIO_GPIO_IO
END

data/system.ucf:
# MDC. For SP3ADSP-1800, this is location F4
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<0> LOC = F4;
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<0> IOSTANDARD = LVCMOS18;

# MDIO. For SP3ADSP-1800, this is location F5
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<1> LOC = F5;
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<1> IOSTANDARD = LVCMOS18;

# PHY Reset. For SP3ADSP-1800, this is location G4
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<2> LOC=G4;
Net fpga_0_MII_MDC_MDIO_GPIO_IO_pin<2> IOSTANDARD = LVCMOS18;

```

Note that a system generated with Base System Builder will connect the PHY reset to the xps\_ethernetlite PHY\_rst\_n port. In the included SP3ADSP-1800 reference system, the PHY reset is connected to a GPIO, as shown above. This is due to the long reset time of 150 microseconds required by the National DP83865 Ethernet PHY, far longer than xps\_ethernetlite asserts this reset.

# PHY Serial Management Bus

The serial management bus used to access PHY registers is a 2-wire bus with clock (MDC) and bidirectional data (MDIO). The bus protocol is described in IEEE 802.3u. This bus allows for multiple PHY devices to be present. Each PHY has a unique 5-bit address, determined by device strapping. The register address space is also 5 bits, which allows for a maximum of 32 registers. Registers are 16 bits. Data are written a bit at a time on the data line MDIO to be clocked on the rising edge of MDC.

The PHY requires that at least one **preamble** occur before any registers are accessed. The preamble is 32 bits of '1' sent by the controller to the PHY. The software included in the application note will always send a preamble for each register access (preamble suppression is not used). Note that the preamble is not shown in the timing diagrams supplied within this application note, even though the software will produce one for every register access.

The MDIO is generally a high value (logic '1') between operations because a pullup resistor on this signal. Transactions are initiated by the controller sending a start value of '01'.

The serial bus is bidirectional. The host writes the desired operation (read/write), the PHY address, and the desired register in that PHY. If the operation is a read, the PHY will drive the contents of the indicated register out on the MDIO signal which the controller will read after a two bit-time "Turn Around" period. During the first clock of this period, neither the PHY nor the controller shall drive MDIO. The PHY is expected to drive MDIO '0' during the second clock of Turn Around. See [Figure 3](#) for an example diagram of a read operation. For writes, the host sends the 16 bit register contents on the MDIO line. In this instance, the controller will continue to drive MDIO during the Turn Around period. The controller will send '10' for these two bit-times. See [Figure 4](#) for an example diagram of a write operation.

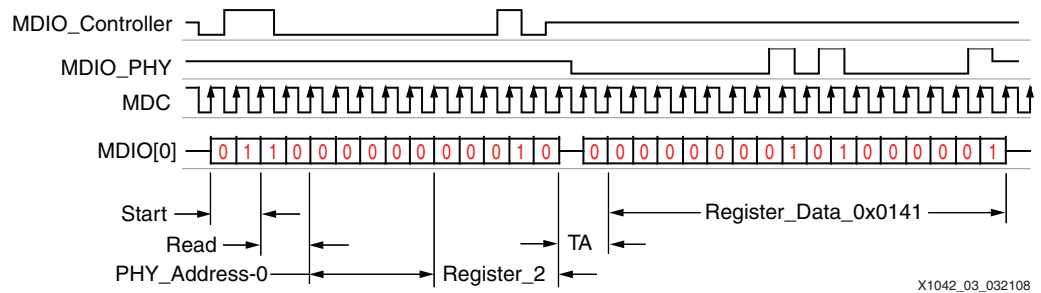


Figure 3: MDC/MDIO PHY Register Read

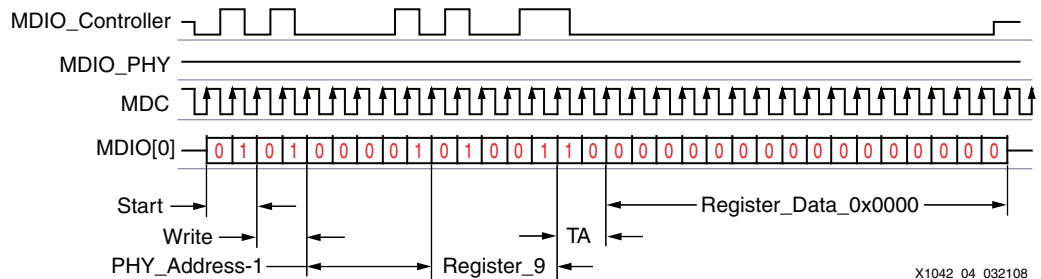


Figure 4: MDC/MDIO PHY Register Write

## PHY Registers

Each manufacturer defines their unique register specification. Which registers, and which bits in those registers control any given PHY configuration setting, vary from device to device. The one constant standard is the **PHY ID registers**, which are always **registers 2 and 3**. These registers contain the **Organizationally Unique Identifier (OUI)** of the manufacturer, and the model and revision unique to that manufacturer. See the specifications to determine how these registers are partitioned.

The software which accesses PHY registers must always determine which PHY is present, so that it will know the appropriate registers to access. This application note includes two reference systems, one for an ML403 board and the other an SP3ADSP-1800 board. The ML403 board has a Marvell 88E1111 PHY, and the SP3ADSP-1800 board has a National DP83865 PHY.

```
National DP83865
Register 2: 0x2000
Register 3: 0x5C7A
OUI: 0x20005C >> 2
OUI: 0x080017
```

```
Marvell 88E1111
Register 2: 0x0141
Register 3: 0x0CC1
OUI: 0x01410C >> 2
OUI: 0x005043
```

## Software Application

The included application, with a binary for each system, demonstrates the benefits of the ability to access PHY registers via software.

SP3ADSP-1800 output from Mii\_bitbang.elf:

```
Demonstration of PHY register access with GPIOs:
PHY Reset.
Read from phy 1
REG 2: 0x2000
REG 3: 0x5C7A
Waiting for auto-negotiation to complete.
PHY Status: LINK_OK SPEED-1000MB FULL-DUPLEX MDIX AUTONEG_CMPLT
Disabling 1000MB and initiating re-negotiation.
Waiting for auto-negotiation to complete.
PHY Status: LINK_OK SPEED-100MB FULL-DUPLEX MDI AUTONEG_CMPLT
```

ML403 output from Mii\_bitbang.elf:

```
Demonstration of PHY register access with GPIOs:
Read from phy 0
REG 2: 0x0141
REG 3: 0x0CC1
Waiting for auto-negotiation to complete.
PHY Status: LINK_OK SPEED-1000MB FULL-DUPLEX SPD_DPLX_RSLVD MDI
Disabling 1000MB and initiating re-negotiation.
Waiting for auto-negotiation to complete.
PHY Status: LINK_OK SPEED-100MB FULL-DUPLEX SPD_DPLX_RSLVD MDI
```

In the above examples, the Xilinx board was connected to a PC with a Gigabit Ethernet NIC. As expected, the default result of auto negotiation is 1000MB, as shown in the preceding examples. Because the XPS Ethernetlite cannot operate at this link speed, the application configures the PHY so that it will not negotiate 1000MB links and initiates re-negotiation. The new result, 100MB, is the desired outcome.

Each PHY has its own register definitions. The first action the provided application will take prior to configuring the PHY is to verify that the expected device is present. This is done by

examining the PHY ID registers, as discussed previously. Two implementations are provided with this application note, **marvell\_88e1111.c** for the ML403 system, and **national\_dp83865.c** for the SP3ADSP-1800 system.

**Note:** PHY configuration is not standardized - the register specification for each Ethernet PHY is unique. The software must perform whatever operations are appropriate to the particular PHY connected to the XPS Ethernetlite.

The serial bus clock, MDC, is generated by software toggling the appropriate GPIO bit at the desired interval. The function **mdc\_clk\_1\_0()** shown below is used to generate this clock.

```

/*
 * mdc_clk_1_0:
 * Drive the MII Data Clock 1->0
 */
static inline void
mdc_clk_1_0 (XGpio *mii_gpio)
{
    mii_delay();
    mdc_drive_bit(mii_gpio, 1);
    mii_delay();
    mdc_drive_bit(mii_gpio, 0);
}

```

The PHY expects MDIO to have valid data on the rising edge of MDC. As such, MDIO is set immediately before calling this function. The clock frequency is determined by the delay between toggling the MDC signal. This delay is created by **mii\_delay()**.

The two reference systems achieve this differently.

The PPC405 includes a built-in timer. The standalone library implements the well-known function **udelay()** to pause for a desired number of microseconds using this timer. The MicroBlaze has no built-in timer.

The SP3ADSP-1800 MicroBlaze reference system includes an XPS Timer peripheral to serve this purpose. The **udelay()** function is implemented for this system in **xtmrctr\_udelay.c** which is provided with the application. The software uses a delay of 1 microsecond, which will clock the PHY at approximately 500KHz.

## Executing the Reference System

Configure a HyperTerminal or similar program to use the COM port. Connect the RS232 connector of the Xilinx board to the COM port via a serial cable. Set the HyperTerminal to Baud Rate of **9600**, Data Bits to **8**, Parity to **None**, and Flow Control to **None**.

### Executing the Reference System using the Pre-Built Bitstream and the Compiled Software Applications

To execute the system using files inside the `ready_for_download/` directory in the project root directory, follow these steps:

1. Change directories to the `ready_for_download` directory.
2. Use **iMPACT** to download the bitstream by using the following command:
 

```
impact -batch xapp1042.cmd
```
3. Invoke **XMD** and connect to the processor by using the following command:
 

```
xmd -opt xapp1042.opt
```
4. Download the executables by using the following command:
 

```
dow Mii_bitbang.elf
```

## Executing the Reference System from XPS

To execute the system using EDK, follow these steps:

1. Open `system.xmp` in EDK.
2. Use **Hardware** → **Generate Bitstream** to generate a bitstream for the system.
3. Download the bitstream to the board with **Device Configuration** → **Download Bitstream**.
4. Launch XMD with **Debug** → **Launch XMD...**
5. Right -click on the **Mii\_bitbang** application project, then select **Build Project** to generate the executable for the provided application.
6. Download the generated executable by using the following command:

```
down Mii_bitbang/executable.elf
```

The expected program output is shown in the “[Software Application](#)” section.

## Conclusion

This application note has demonstrated a simple and practical method for ensuring that systems which use the XPS Ethernetlite core will auto negotiate the PHY link to a configuration which is compatible with this core.

## Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
4/04/08	1.0	Initial Xilinx release.
5/02/08	1.0.1	Update document design file links

## Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.