Interactive Graphical Program to access Ethernet PHY registers

Ding Yourun, Dmitry E. Gouriev

Abstract—**The MDIO interface is used in conjunction with the Ethernet MAC-PHY interfaces MII, RMII, SMII, GMII, RGMII, SGMII and provides an access to the internal registers of the Ethernet transceiver (which is also known as the Ethernet PHY). The only Linux operating system provides a unified API to access these registers. The only three programs provide user access to these registers, and all of these programs provide command-line user interface only. At the same time, there are calls from Ethernet hardware designers for access software tool of a some kind which could be able to provide interactive access to PHY registers in the form of windows, tables and input fields. The article describes the first program which meets these calls.**

Keywords—**Ethernet, MII, RMII, SMII, GMII, RGMII, SGMII, MDIO, PHY, MAC, autonegotiation, Linux, ioctl, phytool, mii-reg, mfectl, Qt library, SSH, 82574L, KSZ9021RL**

I. Introduction

The MDIO interface ([1], [2]) is used in conjunction with the Ethernet MAC-PHY interfaces MII, RMII, SMII, GMII, RGMII, SGMII and provides an access to the internal registers of the Ethernet transceiver (which is also known as the Ethernet PHY).

The MDIO interface is used mostly by Ethernet MAC-controllers and transceivers and its software, to configure Ethernet network interface to desired mode or, more commonly, to track the process of autonegotiation of the mode and the resulting mode of the network interface.

However, the needs for usage of the MDIO grow dramatically inside of the processes of designing, debugging and testing Ethernet hardware, including compliance testing.

The transceiver must be configured to special test modes for electrical measurements.

Loopbacks of various directions (local/remote) should be configured at various sublayers and/or subdevices.

Various configurations for fixed or autonegotiated modes are used in various test scenarios.

In many test scenarios the link-partner device must be configured as well as the device under test.

All these configuration procedures use the MDIO interface.

Manuscript received June 5, 2020.

Dmitry E. Gouriev is scientists in Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University, Moscow, Russia (email gouriev@oit.cmc.msu.ru).

Ding Yourun is student in Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University (email during1031@gmail.com).

The only Linux operating system provides a unified API to access Ethernet PHY registers, using special ioctl() calls SIOCGMIIPHY and SIOCSMIIPHY.

There are the only three programs which provide user access to these registers: phytool [3], mii-reg [4] and mfectl [5], and all of these programs provide command-line user interface only.

At the same time, there are calls from Ethernet hardware designers for access software tool of a some kind which could be able to provide interactive access to the PHY registers in the form of windows, tables and input fields.

The article is dedicated to the first step in this direction and describes the first program which meets these calls of hardware designers.

Design and implementation of the program were a part of a master thesis of Ding Yourun, done under direction of Dmitry E. Gouriev.

II. Some details of MDIO interface

The MDIO (Management Data Input/Output) interface connects Ethernet MAC-controller and Ethernet PHY (transceiver) for management purpose only. Data to send to the network or received from the network are exchanged via data interfaces like MII, RMII, SMII, GMII, RGMII or SGMII.

The MDIO interface consists of two signals: clock signal MDC and bidirectional data signal MDIO. The interface can operate under aperiodic clock in frequency range from 400 KHz to 2,5 MHz. The interface provides an exchange of interface frames of the following structure (see Figure 1):

1) 32-bit preamble filled with “1” bit value, optional,

2) 2-bit “start of frame” filled with value “01”,

3) 2-bit code of operation: “10” for reading and “01” for writing,

4) 5-bit PHY address (up to 32 PHYs on the single MDIO bus are allowed),

5) 5-bit PHY register address (up to 32 registers in the single PHY are allowed for direct access),

6) 2-bit pause for data source switching: the “10” bit sequence is transmitted by the MAC-controller in the write operation, whereas in the read operation the MAC-controller switches its MDIO signal to high impedance state and PHY transmits the “0” value bit in the second bit, resulting in “Z0” transmission,

7) 16-bit register value, in the write operation it is transmitted by the MAC-controller, whereas in the read operation it is transmitted by the PHY.

The interface provides direct access to 32 PHY registers. Registers 0-15 are defined in the standard [1], and registers 16-31 are defined by the manufacturer of the PHY.

Such a small number of the registers is usually unsufficient to control all aspects of the work of contemporary Ethernet PHYs. For this purpose the Ethernet standard provides an ability to access registers in additional register pages, or – in terms of the MDIO itself – MMDs (MDIO Manageable Devices). The standard [1] provides this ability using registers 13 and 14 for indirect access, and the standard [2] provides the same using an advanced MDIO frame structure (see Figure 2). However, some producers break the standard in this part and implement their own indirect access schemes (see Intel 82574L and Micrel KSZ9021RL as examples).

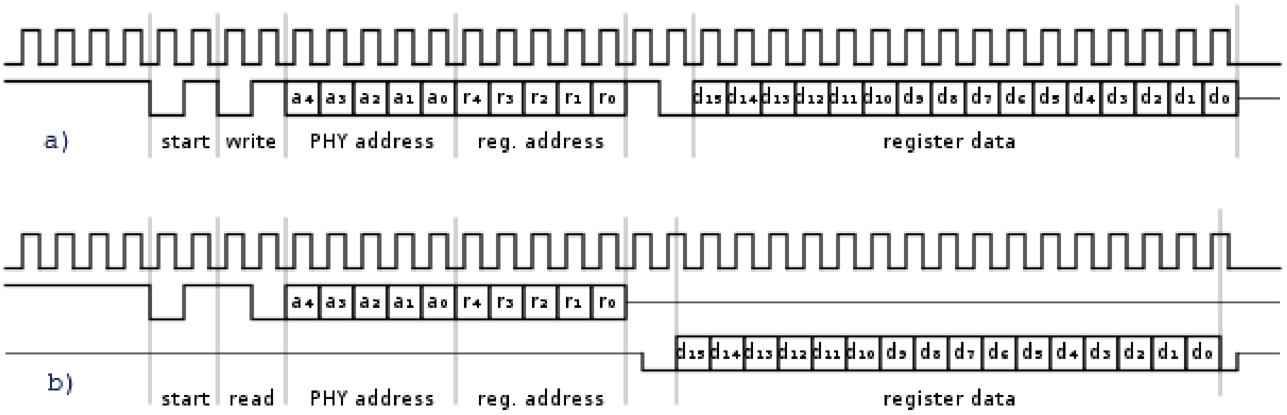


Figure 1. MDIO frame format, in accordance with IEEE 802.3 Clause 22. a) – write a register, b) – read a register.

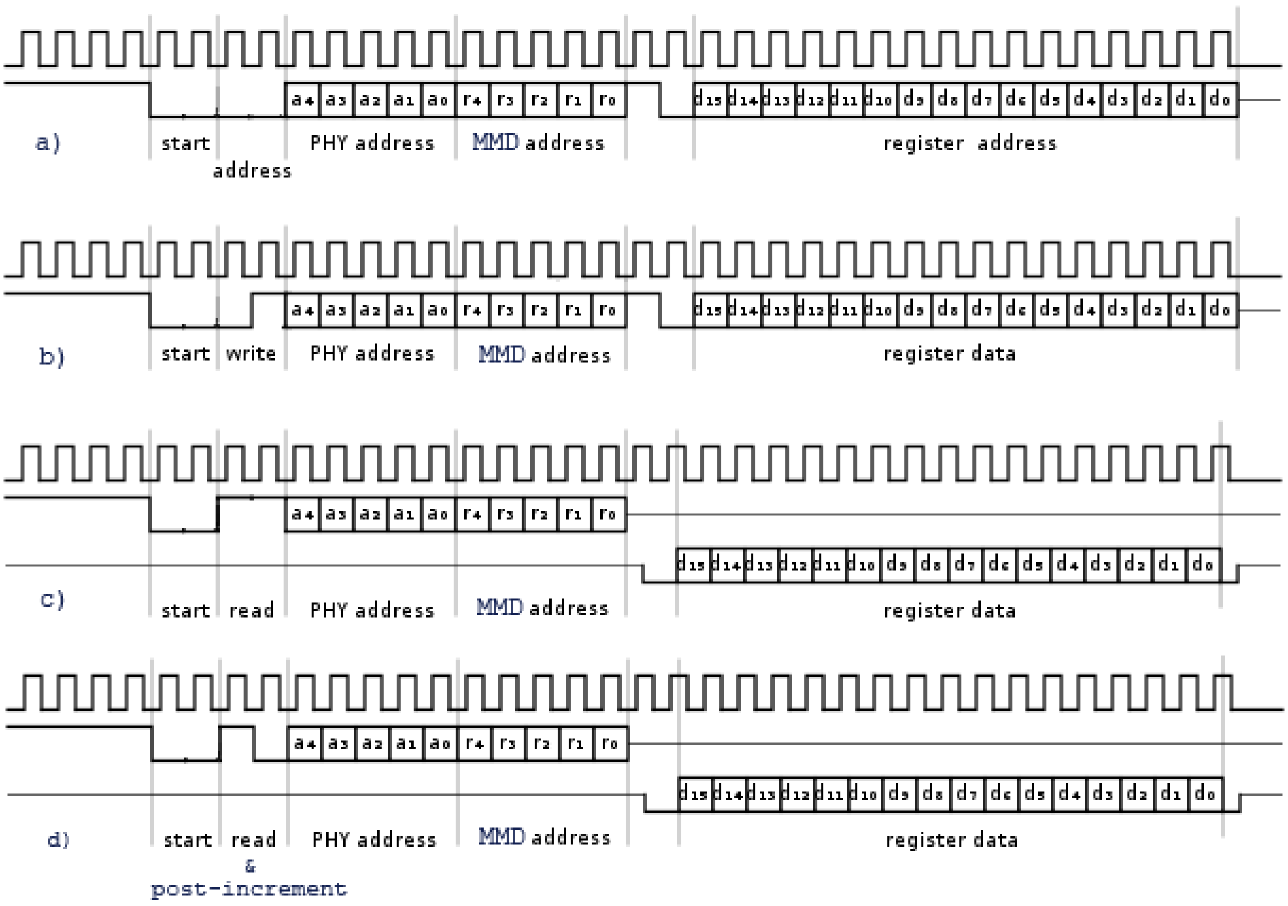
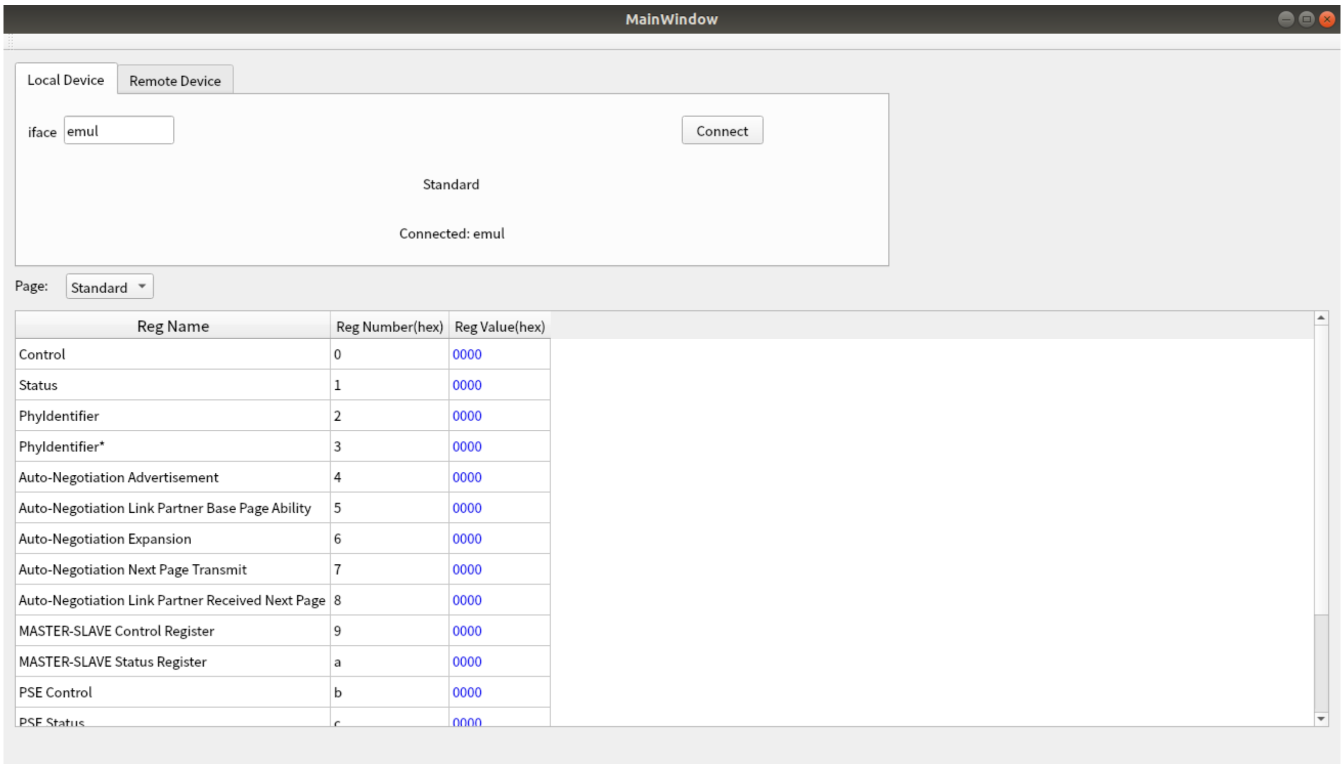


Figure 2. MDIO frame format, in accordance with IEEE 802.3 Clause 45. a) – set register address, b) – write a register, c) – read a register, d) – read a register and post-increment the address.



iface emul

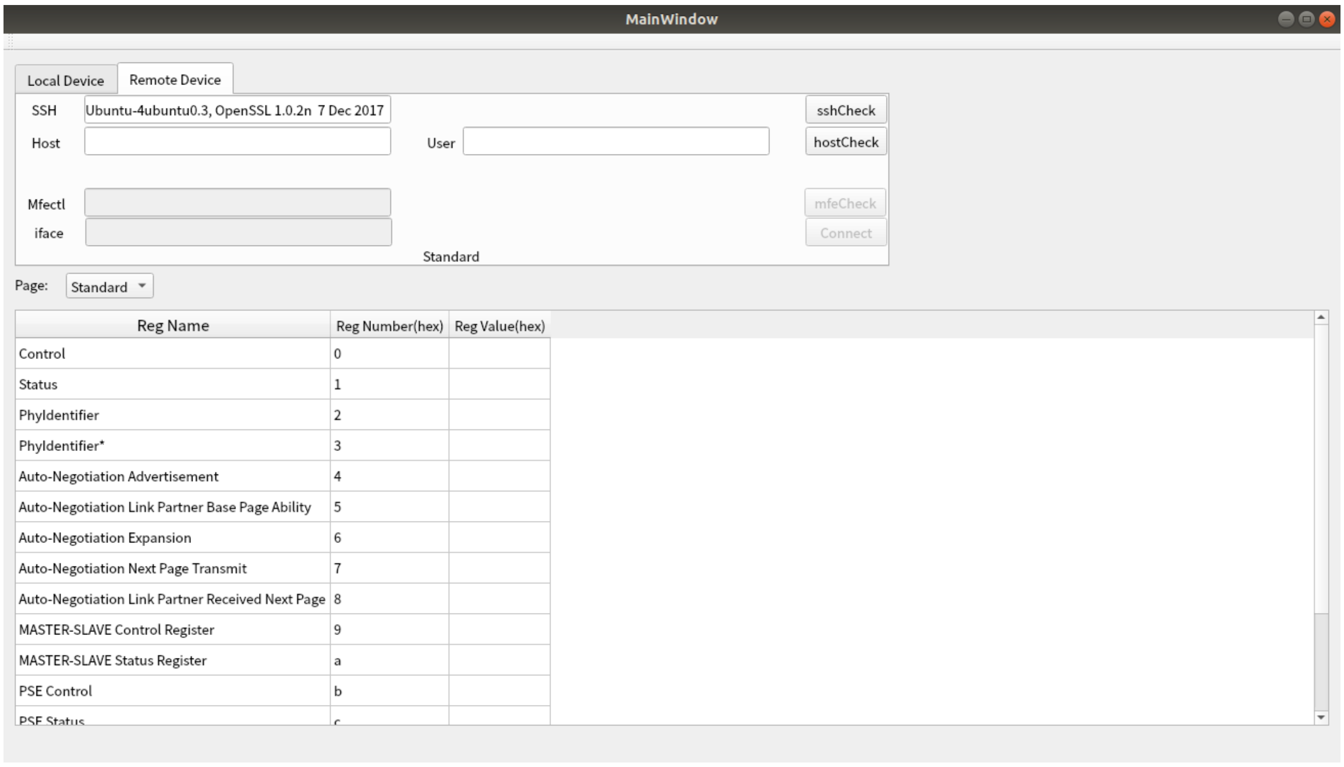
Standard

Connected: emul

Page: Standard

Reg Name	Reg Number(hex)	Reg Value(hex)
Control	0	0000
Status	1	0000
PhyIdentifier	2	0000
PhyIdentifier*	3	0000
Auto-Negotiation Advertisement	4	0000
Auto-Negotiation Link Partner Base Page Ability	5	0000
Auto-Negotiation Expansion	6	0000
Auto-Negotiation Next Page Transmit	7	0000
Auto-Negotiation Link Partner Received Next Page	8	0000
MASTER-SLAVE Control Register	9	0000
MASTER-SLAVE Status Register	a	0000
PSE Control	b	0000
PSE Status	c	0000

Figure 3. The program GUI.



SSH Ubuntu-4ubuntu0.3, OpenSSL 1.0.2n 7 Dec 2017

Host

User

Mfectl

iface

Standard

Page: Standard

Reg Name	Reg Number(hex)	Reg Value(hex)
Control	0	
Status	1	
PhyIdentifier	2	
PhyIdentifier*	3	
Auto-Negotiation Advertisement	4	
Auto-Negotiation Link Partner Base Page Ability	5	
Auto-Negotiation Expansion	6	
Auto-Negotiation Next Page Transmit	7	
Auto-Negotiation Link Partner Received Next Page	8	
MASTER-SLAVE Control Register	9	
MASTER-SLAVE Status Register	a	
PSE Control	b	
PSE Status	c	

Figure 4. The program GUI in the remote mode.

III. THE PROGRAM GUI

The program GUI is presented in Figure 3.

The upper part of the program window contains a panel for parameters and actions (buttons) to connect to the PHY.

In the simplest case in Figure 3 there is the only one parameter – a name of a network interface (i.e. eth0, eth1, etc.).

In connected state, the program displays the product name of the PHY, in accordance with its manufacturer description, if the program is able to detect it.

The lower part of the program window contains a table of PHY registers.

Each register is displayed with its name, register number and current value. The values are updated several times per second.

The names of the registers are displayed in accordance with the manufacturer’s documentation, if the program is able to detect the type of the PHY, otherwise register names in accordance with [1] are displayed.

User can change (edit) the value of any register in the table, and when finished (pressing the Enter key), the new value is transferred into the register. The editing state of the register is indicated by rendering its value in red color.

There is a page selector between connection panel and register table, which provides switching between main register page and additional indirect register pages (MMDs). On changing the value of the selector the table is reloaded with the names of the registers of the selected register page and starts display register values from the selected page.

The more complex case, called “remote mode”, is shown in Figure 4. In this case the program provides access to PHY registers on another computer. The SSH protocol and the ssh external program are used to connect to that remote computer. The mfectl program [5] on that remote computer is used as an agent, which provides access to PHY registers.

There are additional parameters to be entered to make the connection: remote computer hostname or IP-address (Host) and username on the remote computer (User).

Due to still-experimental state of the program, there are also additional actions to establish a remote connection step-by-step. At 1st step we are to check the presence of the ssh external program, at 2nd step we are to check an ability to make an SSH connection to a remote computer, and at the 3rd step we are to check the presence of the mfectl program on the remote computer.

After all, the user enters a network interface name in the remote computer and commits the connection. Further work with PHY registers does not differ from the simple case in Figure 3.

IV. THE PROGRAM FEATURES

Here we summarize the main features of the program:

The program displays and provides an ability to change the values of the PHY registers.

The program displays register names in accordance with manufacturer description for known types of PHY.

The program provides an access to the registers in additional register pages (MMDs) as well.

The program provides an access to the registers of the PHYs in remote hosts as well.

V. IMPLEMENTATION DETAILS

The program uses ioctl calls [6] SIOCGMIIREG and SIOCSMIIREG to access PHY registers on local computer.

The program, uses the ssh external program to make a network connection to a remote computer and the mfectl program on the remote computer to access PHY registers in the remote computer.

The program implements the indirect access algorithms in accordance with [1] to access registers in additional register pages.

The program uses the descriptions of the registers of PHY of particular types in the form of “ini”-files, as shown in the following example:

```
[Intel 82574L]
0="Control"
1="Status"
2="PhyIdentifier"
3="PhyIdentifier*"
……
```

The program detects the type of the PHY by analyzing values in the registers 2 and 3 (PHY Identifier).

The GUI of the program was designed and developed on the top of the Qt Library, version 5 [7].

VI. CONCLUSION

Interactive graphical access to Ethernet PHY registers is the actual requirement of processes of Ethernet hardware design, testing and debugging.

The presented program meets this requirement.

The program GUI, features and implementation details were described in the article.

REFERENCES

[1] IEEE Std 802.3 - 2012. IEEE Standard for Ethernet. Clause 22.
[2] IEEE Std 802.3 - 2012. IEEE Standard for Ethernet. Clause 45.
[3] Tobias Waldekranz. Phytool. https://github.com/wkz/phytool. Retrieved: 15/01/2019.
[4] Tim Harvey. conformance testing: mii-reg.c. http://trac.gateworks.com/attachment/wiki/conformance_testing/mii-reg.c. Retrieved: 15/01/2019
[5] Dmitry E. Gouriev. Tools to control an Ethernet transceiver under testing and debugging. International Journal of Open Information Technologies, Vol. 6, No 4, 2018. (In Russian). http://injoit.org/index.php/j1/article/view/556. Retrieved: 30/06/2019.
[6] ioctl (2). Linux manual pages.
[7] QT library. https://www.qt.io/product/framework. Retrieved: 30/01/2019