**iRail**

**PHYsical (PHY) Interface**

**Requirements and Design**

**Specification**

**Revision 0.4**

**November 12, 2017**

Table of Contents

[1. PHY Requirements 5](#_Toc498274268)

[1.1. PHY Transmit Function Requirements 5](#_Toc498274269)

[1.2. PHY Receive Function Requirements 5](#_Toc498274270)

[1.3. PHY Collision Detection Function Requirements 5](#_Toc498274271)

[1.4. PHY Jabber Functional Requirements 6](#_Toc498274272)

[1.5. MAC / PHY Interface Requirements 6](#_Toc498274273)

[1.5.1. MII Interface Definition 6](#_Toc498274274)

[1.5.2. Management Interface Definition 7](#_Toc498274275)

[1.6. Functionality Not Supported by the iRail PHY 7](#_Toc498274276)

[2. PHY Design Implementation 8](#_Toc498274277)

[2.1. PHY Features 8](#_Toc498274278)

[2.2. PHY Design Implementation 8](#_Toc498274279)

[2.2.1. IEEE 802.3 Ethernet Frame Support 9](#_Toc498274280)

[2.2.1. Transmitter 10](#_Toc498274281)

[2.2.1. PHY Receiver 12](#_Toc498274282)

[2.2.1. Microprocessor Function 14](#_Toc498274283)

[2.2.1. MDIO 15](#_Toc498274284)

[2.2.1. PHY Clock Generation 15](#_Toc498274285)

[3. PHY Simulation Test Bench 17](#_Toc498274286)

[4. IEEE-defined Registers 18](#_Toc498274287)

[4.1. Register 0 - Basic Mode Control Register 18](#_Toc498274288)

[4.2. Register 1 - Basic Mode Status Register 20](#_Toc498274289)

[4.3. Register 2 - PHY Identifier Register 1 21](#_Toc498274290)

[4.4. Register 3 - PHY Identifier Register 2 21](#_Toc498274291)

[4.5. Register 4 - Auto-Negotiation Advertisement Register (ANAR) 21](#_Toc498274292)

[4.6. Register 5 - Auto-Negotiation Link Partner Ability Register (ANLPAR) 22](#_Toc498274293)

[4.7. Register 6 - Auto-Negotiation Expansion Register (ANER) 22](#_Toc498274294)

[4.8. Register 7 - Auto-Negotiation Next Page Transmit Register (ANNPTR) 22](#_Toc498274295)

[4.9. Register 8 - Auto-Negotiation Next Page Receive Register (ANNPRR) 22](#_Toc498274296)

[4.10. Register 9 - 1000Base-T Control Register (GBCRR) 22](#_Toc498274297)

[4.11. Register A - 1000Base-T Control Register (GBSR) 23](#_Toc498274298)

[4.12. Register B - ?? (????R) 23](#_Toc498274299)

[4.13. Register C - ?? (????R) 23](#_Toc498274300)

[4.14. Register D - MMD Access Control Register (MACR) 23](#_Toc498274301)

[4.15. Register E - MMD Access Address Data Register (MAADR) 23](#_Toc498274302)

[4.16. Register F - 1000Base-T Extended Status Register (GBESR) 23](#_Toc498274303)

[5. MII FPGA Software Interface 24](#_Toc498274304)

[5.1. Memory Map 24](#_Toc498274305)

[5.2. Interrupts 24](#_Toc498274309)

[5.3. Register Definitions 25](#_Toc498274311)

[5.3.1. Control Register 25](#_Toc498274312)

[5.3.2. Interrupt Register 26](#_Toc498274313)

[5.3.3. Interrupt Mask Register 27](#_Toc498274314)

[5.3.4. Status Register 28](#_Toc498274315)

[6. Media Access Control (MAC) Functionality 29](#_Toc498274316)

[6.1. General MAC Functions 29](#_Toc498274319)

[6.2. Transmitter MAC Functions 29](#_Toc498274320)

[6.3. Receiver MAC Functions 29](#_Toc498274322)

Table of Figures

[Figure 1 MII Interface Block Diagram 6](#_Toc498274323)

[Figure 2 PHY High Level Block Diagram 9](#_Toc498274324)

[Figure 3 802.3Q Frame Format 9](#_Toc498274325)

[Figure 4 Transmit State Machine 11](#_Toc498274326)

[Figure 5 Jabber Flow Diagram 12](#_Toc498274327)

[Figure 6 Receive State Machine Flow Diagram 14](#_Toc498274328)

[Figure 7 PHY Clock Generation 15](#_Toc498274329)

[Figure 8 PHY SoC Clock Domains 16](#_Toc498274330)

1. PHY Requirements

The PHY shall provide the following functions.

* *Transmit function.* The ability to transmit serial data bit streams on the baseband medium from the local station to one or more remote stations on the same physical medium or iRail.
* *Receive function.* The ability to receive serial data bit streams over the physical medium.
* *Collision Presence function.* The ability to detect the presence of two or more stations’ concurrent transmissions.
* *Jabber function.* The ability to automatically interrupt the transmit function and inhibit an abnormally long output data stream.

All PHYs are required to implement collision detection per IEEE 802.3. This is typically performed by measuring the presence of energy on the physical medium, in our case, iRail. Refer to

PHY Collision Detection Function Requirements section below.

* 1. PHY Transmit Function Requirements

Below are the PHY transmitter functions.

* Interface to the MII
* Parallel to serial conversion of MII data
* Manchester encoding of serial data
* Physical conversion of TTL Level Manchester encoded serial data to the iRail signal levels.
* Jabber Function
* Collision Presence Function
  1. PHY Receive Function Requirements

Below are the PHY receiver functions.

* Conversion of iRail signals to serial TTL levels
* Manchester decoding of serial TTL level data
* Clock adaptation between transmitting station and receiving station
* Serial to parallel data conversion
* Interface to the MII interface

* 1. PHY Collision Detection Function Requirements

Collision detection is typically performed in the PHY by measuring the energy levels on the signal on the physical medium. Since the iRail products do not have a mechanism to measure these levels, alternative means is required. There are several potential approaches below and are for further discussion.

* Incorporate a circuit to detect collisions directly on the physical medium
* Compare the transmitted data to the received data and differences will be classified as collisions. This does not allow for receivers only (stations not transmitting) to detect collisions. Two approaches to this comparison method could be applied
  + Compare transmitted data to received data after the clock adaptation
  + Compare transmitted data to received data before the clock adaptation. This is feasible, and more simplistic since the transmitter and receiver operate off the same clock frequency.
  1. PHY Jabber Functional Requirements

The PHY shall contain a self-interrupt capability to inhibit transmit data from reaching the medium. Hardware within the PHY (with no external message other than the detection of output data, bits, or leakage, by way of the transmit function) shall provide a nominal window of at least 20 ms to at most 150 ms during which time a normal data link frame may be transmitted. If the frame length exceeds this duration, the jabber function shall inhibit further output data from reaching the medium.

When the transmit function has been positively disabled, the PHY shall then activate the collision presence function as close to the trunk coaxial medium as possible without introducing an extraneous signal on the trunk coaxial medium. A PHY without the monitor function may reset the jabber and collision presence functions on power reset. Alternatively, a MAU without the monitor function may reset these functions after a period of 0.5 s ± 50% if the monitor function has not been implemented. If the monitor function has been implemented then it shall be used to reset the collision presence and jabber functions.

* 1. MAC / PHY Interface Requirements

Two different signal interfaces shall exist between the MAC and PHY. One of the mechanisms is used for data (MII Interface), the other is used for control (MDIO Interface).

* + 1. MII Interface Definition

Below are the signal definitions for the MII Interface.



Figure 1 MII Interface Block Diagram

|  |  |  |
| --- | --- | --- |
| **Signal name** | **Description** | **Direction** |
| TX\_CLK | Transmit clock | PHY to MAC |
| TXD0 | Transmit data bit 0 (transmitted first) | MAC to PHY |
| TXD1 | Transmit data bit 1 | MAC to PHY |
| TXD2 | Transmit data bit 2 | MAC to PHY |
| TXD3 | Transmit data bit 3 | MAC to PHY |
| TX\_EN | Transmit enable | MAC to PHY |
| TX\_ER | Transmit error (optional) | MAC to PHY |

Table 1 MII Transmit Interface Signal Descriptions

|  |  |  |
| --- | --- | --- |
| **Signal name** | **Description** | **Direction** |
| RX\_CLK | Receive clock | PHY to MAC |
| RXD0 | Receive data bit 0 (received first) | PHY to MAC |
| RXD1 | Receive data bit 1 | PHY to MAC |
| RXD2 | Receive data bit 2 | PHY to MAC |
| RXD3 | Receive data bit 3 | PHY to MAC |
| RX\_DV | Receive data valid | PHY to MAC |
| RX\_ER | Receive error | PHY to MAC |
| CRS | Carrier sense | PHY to MAC |
| COL | Collision detect | PHY to MAC |

Table 2 MII Receive Interface Signal Descriptions

* + 1. Management Interface Definition

Below are the signal definitions for the Management Interface.

|  |  |  |
| --- | --- | --- |
| **SIGNAL NAME** | **DESCRIPTION** | **Direction** |
| MDC | **MANAGEMENT DATA CLOCK:** Clock signal for the management data input/output (MDIO) interface. The maximum MDC rate is 25 MHz; there is no minimum MDC rate. MDC is **not** required to be synchronous to the TX\_CLK or the RX\_CLK. | **MAC to PHY** |
| MDIO | **MANAGEMENT DATA I/O:** Bidirectional command / data signal synchronized to MDC. Either the PHY or the MAC may drive the MDIO signal. | **Bidirectional** |

Table 3 Management Interface Signal Definition

* 1. Functionality Not Supported by the iRail PHY

The below functions are not supported by the iRail PHY.

* **Link Integrity Test:** Provides the ability to protect the network from the consequences of failure of the simplex link attached to the RD circuit.
* **Auto-Negotiation:** Optionally provides the capability for a device at one end of a link segment to advertise its abilities to the device at the other end.
* **Collision Detect:** Since the PHY SoC can’t measure the energy levels on the signal on the physical medium, alternative methods of collision detection is necessary.

1. PHY Design Implementation

The following sections describe the PHY design implementation.

* 1. PHY Features

Below are the features provided by the PHY SoC.

* IEEE 802.3Q Ethernet Frame Compatible
* Transmit Function
* Receive Function
* Manchester Encoding/Decoding
* Receive Clock Adaptation
* Collision Detection
* Jabber Function
* MII Transmit and Receive Interface
  1. PHY Design Implementation

Figure 2 PHY High Level Block Diagram provides a high level description of the MII PHY SoC. The functionality can be observed below and are described in more detail later in this specification.

* Transmit function with the following sub-functions
  + Transmit FIFO
  + Transmit State Machine
  + Manchester Encoder
* Receive function with the following sub-functions
  + Manchester Decoder
  + Clock Recovery
  + Receive FIFO
  + Receive State Machine
* Microprocessor Interface
* MDIO
* Clock Generation

Figure 2 PHY High Level Block Diagram also provides the physical interfaces which can be observed below and are described in more detail later in this specification.

* MII Interface
* AFE Interface
* APB Interface

The functionality and interfaces are discussed in more detail within this specification.



Figure 2 PHY High Level Block Diagram

* + 1. IEEE 802.3 Ethernet Frame Support

The SoC PHY supports IEEE 802.3Q frame formats. Refer to Figure 3 802.3Q Frame Format below.



Figure 3 802.3Q Frame Format

The IEEE 802.3Q frame is transparent to the SoC PHY. The MAC is responsible for assembling the entire packet, from the pre-amble through the CRC/FCS bytes. The MAC is also responsible for enforcing the Inter-Frame Gap. The SoC PHY may opt to pick off the Ethernet Size/Type field to monitor and enforce the packet length. [Is there another way to determine the end of packet without use of the Ethernet Size/Type field?]

* + 1. Transmitter

The SoC PHY is responsible for the reliable data transfer from the MII interface to the iRail. To perform this function, several sub-functions are implemented.

* + - 1. TX FIFO

The TX FIFOs interfaces with the Transmit MII interfaces. The TX FIFO is 2048 bytes in depth and 8-bits in width. The TX FIFO stores one packet at a time. It is loaded by the Transmit MII Interface. At the start of a transmit packet, the FIFO will fill up to 48 bytes before starting the transmission. Note that the minimum 802.3Q frame size is 64 bytes. This allows the TX FIFO to protect against under-runs. Under-runs occur when the TX MII interface stalls in the middle of a packet transmission simultaneously with bytes are being pulled from the FIFO and transmitted onto the iRail. The transmitter is informed of an End of Frame when the TX FIFO empty flag asserts. [Is this means to determine the end of packet acceptable?]

* + - 1. TX State Machine

The TX State Machine is responsible for moving data to the TX AFE Interface during the appropriate transmission window. Refer to Figure 4 Transmit State Machine.

First, the MII TX interface begins loading the Ethernet Frame, including the pre-amble pattern, into the TX FIFO. When the TX FIFO level reaches 48 bytes the TX State Machine will start transmitting. To perform the transmission, the TX State Machine controls the logic to perform the below functions.

1. Reading a byte from the TX FIFO
2. Perform parallel to serial conversion
3. Manchester Encode
4. Transmission out the AFE

During this transmission process, the TX State Machine monitors for either a collision or jabber detection. When a collision is detected while transmitting, the SoC PHY will provide a collision present indication via the MII TX Interface. The TX State Machine is not directly affected by collisions when transmitting. When a jabber condition is detected while transmitting, the TX State Machine will react as specified in the Jabber Function section of this document. If the jabber condition is detected, the jabber function inhibits further output data from reaching the AFE and issues an interrupt to the processor. When the transmitter has been disabled, the PHY activates the collision presence function via the MII RX Interface.

If an Ethernet Frame is transmitted without faults and the TX FIFO empties, the TX State Machine terminates transmission on to the iRail.



Figure 4 Transmit State Machine

* + - 1. TX Parallel to Serial Conversion

The TX Parallel to Serial Conversion converts the parallel data from the TX FIFO to serial data.

* + - 1. Manchester Encoding

The serial data is converted to Manchester Encoded data before being transmitted to the AFE. Manchester encoding is creating by XOR-ing the data with a clock twice the data rate, or 10.00 MHz.

* + - 1. Jabber Function

The Jabber Function provides a self-interrupt capability to inhibit transmit data from reaching the iRail. The Jabber Function provides a nominal window of 20 ms during which time a normal data frame may be transmitted. If the frame length exceeds this duration, the jabber function inhibits further output data from reaching the AFE and issues an interrupt to the processor. When the transmitter has been disabled, the PHY activates the collision presence function to the RX MII Interface. Refer to Figure 5 Jabber Flow Diagram below for details of the jabber function operation.



Figure 5 Jabber Flow Diagram

* + 1. PHY Receiver

The PHY SoC is responsible for the reliable data transfer from the iRail to the MII interface. To perform this function, several sub-functions are implemented.

* RX Timing Recovery
* Manchester Decoding
* Serial to Parallel Conversion
* RX FIFO

Each is discussed in more detail in the proceeding sections within this specification.

* + - 1. RX Timing Recovery

The Receive Timing Recovery determines the timing boundaries of the received data. This is accomplished by oversampling the data 16 times. Once the first edge is detected, the logic knows where to sample data. Furthermore, since there is no common clock distributed along the iRail, it is possible to have data errors due to these clock differences between devices. This situation is overcome by oversampling the data by a little more the 16 times clock, or 163.333 MHz and determining when the receive data is about to slip past the sampling point. When this occurs, the receiver is told to “skip” one of the 163.333 MHz clocks, thus realigning the data and sampling point.

* + - 1. Manchester Decode

After the receiver timing is recovered, the serial data is converted from Manchester to digital NRZ data. This is accomplished by XOR-ing the data with a clock twice the data rate, or 10.00 MHz. Once the data is decoded, it is sent over to the Serial to Parallel Conversion logic.

* + - 1. Serial to Parallel Conversion

The Serial to Parallel Conversion converts the received serial data from the Manchester Decoded NRZ data to parallel data. The parallel data is then loaded into the RX FIFO.

* + - 1. RX FIFO

The RX FIFOs consist of four FIFOs each is 2048 depth by 9-bits. One side of the RX FIFO interfaces to the RX Serial to Parallel Conversion logic. The other side of the RX FIFO interfaces to the RX MII interface.

* + - 1. RX State Machine

The RX State Machine coordinates the reception of data from the AFE to the RX FIFOs. It starts when an edge is detected on the iRail. Once a byte has been Manchester Decoded and converted from serial to parallel data, the RX State Machine is notified that a byte is available and will move it into the RX FIFO. This process continues until the iRail goes IDLE thus shutting down the RX State Machine or the RX FIFO overflows. An RX FIFO overflow condition will stop the reception, interrupt the processor, and then wait for the next frame. See Figure 6 Receive State Machine Flow Diagram for details on the RX State Machine’s operation.



Figure 6 Receive State Machine Flow Diagram

* + 1. Microprocessor Function

The Processor Function provides the means for the local processor to communicate with the FPGA logic. This communication path, or bus, utilizes a standard interface referred to as the ARM Advanced Microcontroller Bus Architecture **(AMBA**). AMBA is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in (SoC) designs. It facilitates development of multi-processor designs with large numbers of controllers and peripherals. This interface consists of an address bus, control signals and an 8-bit data bus. The AMBA bus is commonly referred to as “**APB**” in the MicroSemiconductor documentation.

Furthermore, the processor interface supports address decoding, data bus interface, status/control registers and interrupt control.

* + 1. MDIO

Management data input/output (MDIO) is a serial bus defined for the Ethernet family of IEEE 802.3 standards for the media independent interface (MII). The MII connects the media access control (MAC) devices with the Ethernet physical layer (PHY) circuits. The MDIO reads and writes the control and status registers of the PHY, configures each PHY before operation, and monitors link status during operation.

MDIO was originally defined in Clause 22 of IEEE RFC802.3. In the original specification, a single MDIO interface is able to access up to 32 registers in 32 different PHY devices. These registers provide status and control information such as: link status, speed ability and selection, power down for low power consumption, duplex mode (full or half), auto-negotiation, fault signaling, and loopback.

Note that the MDIO Intellectual Property has limitations. Since it has no IO Ports, some of the control/status register bits are not in the MDIO logic but instead, in the processor interface registers.

* + 1. PHY Clock Generation

The PHY FPGA uses and external 20MHz crystal oscillator to generated all necessary clocks. Refer to Figure 7 PHY Clock Generation below. There are 4 clocks generated, each is discussed in the following sections.



Figure 7 PHY Clock Generation

* + - 1. 71.000 MHz Clock

The 71.000 MHz Clock is used as the AMBA bus clock (or APB bus clock). It must be run at the same or half the frequency of the on-board Cortex-M3 processor. In this case, 71.000 MHz is half the frequency of the on-board Cortex-M3 processor.

* + - 1. 2.500 MHz Clock

The 2.500 MHz Clock is used as the clock for the transmit and receive MII interface.

* + - 1. 20.000 MHz Clock

The 20.000 MHz Clock is twice the iRail bit rate clock (10 MHz). It is used within the transmitter to generate the bit rate and Manchester Encoding.

* + - 1. 163.333 MHz Clock

The 163.333 MHz Clock is 2% faster than sixteen times (160 MHz) the bit rate clock (10 MHz) and used to recover the receive data from the iRail. Refer to RX Timing Recovery section for more details. The 163.333 MHz Clock is used in the receiver to process and store data in the RX FIFO.

* + - 1. Clock Domains

As mentioned previously, there are 4 clocks domains within the PHY SoC; 2.5 MHz, 71.000 MHz, 20.000 MHz and 163.333 MHz. Figure 7 PHY Clock Generation below maps each clock domain on top of the PHY block diagram.



Figure 8 PHY SoC Clock Domains

1. PHY Simulation Test Bench

**TBD**

1. IEEE-defined Registers

The IEEE defined registers 0x00-0x0F shall behave as per the tables, below.  Keep in mind that many behaviors are simplified for iRail, per the content above.

* 1. Register 0 - Basic Mode Control Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.15 | | Reset | 1 = software reset. Bit is self-clearing | RW/SC | Mandatory, per 802.3 |
| 0.14 | | Loop-Back | 1 = loop-back mode 0 = normal operation | RW | 0 default  1 Optional |
| 0.13 | | Speed Select (LSB) | 1 = 100Mbps 0 = 10Mbps | RW | **0 only.**  Writing 1 does nothing. |
| 0.12 | | Auto-Negotiation Enable | 1 = enable auto-negotiation process (override 0.13 and 0.8) 0 = disable auto-negotiation process | RW | **0 only.**  Writing 1 does nothing. |
| 0.11 | | Power Down | 1 = power-down mode 0 = normal operation | RW | Mandatory, per 802.3 |
| 0.10 | | Isolate | 1 = electrical isolation of PHY from MII and TX+/TX- 0 = normal operation | RW | 0 default  1 Optional |
| 0.9 | | Restart Auto-Negotiation | 1 = restart auto-negotiation process 0 = normal operation. Bit is self-clearing | RW | **0 only.**  Writing 1 does nothing. |
| 0.8 | | Duplex Mode | 1 = full-duplex 0 = half-duplex | RW | **0 only.**  Writing 1 does nothing. |
| 0.7 | | Collision Test | 1 = enable COL test 0 = disable COL test | RW | 0 default  1 Optional |
| 0.6 | | Reserved |  | RO | **Returns 0** |
| 0.5 | | Reserved |  | RO | **Returns 0** |
| 0.4 | | Reserved |  | RO | **Returns 0** |
| 0.3 | | Reserved |  | RO | **Returns 0** |
| 0.2 | | Reserved |  | RO | **Returns 0** |
| 0.1 | | Reserved |  | RO | **Returns 0** |
| 0.0 | | Disable Transmitter | 0 = enable transmitter 1 = disable transmitter | RW | Mandatory, per 802.3 |
|  |  |  |  |  |  |

* 1. Register 1 - Basic Mode Status Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.15 | 100BASE-T4 | 1 = T4 capable 0 = not T4 capable | RO | **0 only.**  Writing 1 does nothing. | |
| 1.14 | 100BASE-TX Full-Duplex | 1 = capable of 100BASE-X full-duplex 0 = not capable of 100BASE-X full-duplex | RO | **0 only.**  Writing 1 does nothing. | |
| 1.13 | 100BASE-TX Half-Duplex | 1 = capable of 100BASE-X half-duplex 0 = not capable of 100BASE-X half-duplex | RO | **0 only.**  Writing 1 does nothing. | |
| 1.12 | 10BASE-T Full-Duplex | 1 = 10Mbps with full-duplex 0 = no 10Mbps with full-duplex capability | RO | **0 only.**  Writing 1 does nothing. | |
| 1.11 | 10BASE-T Half-Duplex | 1 = 10Mbps with half-duplex 0 = no 10Mbps with half-duplex capability | RO | **1 only.**  Writing 0 does nothing. | |
| 1.10 | Reserved |  | RO | **Returns 0** | |
| 1.9 | Reserved |  | RO | **Returns 0** | |
| 1.8 | Reserved |  | RO | **Returns 0** | |
| 1.7 | Reserved |  | RO | **Returns 0** | |
| 1.6 | No (MII) Preamble | 1 = preamble suppression 0 = normal preamble | RO | 1 Mandatory.  0 Optional. | |
| 1.5 | Auto-Negotiation Complete | 1 = auto-negotiation process completed 0 = auto-negotiation process not completed | RO | **1 only.** | |
| 1.4 | Remote Fault | 1 = remote fault 0 = no remote fault | RO/LH | **0 only.** | |
| 1.3 | Auto-Negotiation Ability | 1 = capable to perform auto-negotiation 0 = unable to perform auto-negotiation | RO | **0 only.** | |
| 1.2 | Link Status | 1 = link is up 0 = link is down | RO/LL | **1 only.** | |
| 1.1 | Jabber Detect | 1 = jabber detected 0 = jabber not detected. Default is low | RO/LH | **0 only.** | |
| 1.0 | Extended Capability | 1 = supports extended capabilities registers | RO | **0 only.** | |
|  |  |  |  |  |  | |

* 1. Register 2 - PHY Identifier Register 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2.15-0 | OUI | Composed of the 6th to 21st bits of the Organizationally Unique Identifier (OUI), respectively. | RW - before IEEE registration  RO - After IEEE registration | Until this OUI is registered with IEE, the value may be set by the host processor as TBD.  (After registration, the value will be that assigned by the IEEE.) | |
|  |  |  |  |  |  | |

OUI, or nowadays "MAC Address Block Large (MA-L)" is assigned to companies for a fee.  [Reference](https://standards.ieee.org/develop/regauth/oui/index.html).

Since iRail uses ATMEL AT24MAC402 to provide MAC addresses, it may be OK to use the first 24 bits of that number - 0xFCC23D, which is assigned to Atmel.  (I'm not 100% sure about this!) It may be better to use 0xFCC23D as a default, and let register 0x10 and 0x11 be used to overwrite, thus keeping this IEEE-defined register as effectively Read Only.

* 1. Register 3 - PHY Identifier Register 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.15-10 | OUI\_LSB | Assigned to the 0 through 5th bits of the OUI. | RO | TBD - See Register 2 text. |
| 3.9-4 | Model Number | Model Number | RO | TBD (iRail PHY module #) |
| 3.3-0 | Revision Number | Revision Number | RO | TBD (iRail PHY module rev#) |

* 1. Register 4 - Auto-Negotiation Advertisement Register (ANAR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4.15-0 | N/A | Ignored | RO | **0 only.** |

Auto-Negotiation is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register 5 - Auto-Negotiation Link Partner Ability Register (ANLPAR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5.15-6 | N/A | Ignored | RO | **0 only.** |
| 5.5 | 10Base-T | 1: 10Base-T is supported by link partner 0: 10Base-T not supported by link partner | RO | **1 only.** |
| 5.4-0 | Selector Field | Link Partner’s Binary Encoded Node Selector. Only CSMA/CD 00001 is specified. | RO | **00001 only** |

Auto-Negotiation is not supported, so all bits except bits 0-5 act as 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register 6 - Auto-Negotiation Expansion Register (ANER)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6.15-0 | N/A | Ignored | RO | **0 only.** |

Auto-Negotiation is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register 7 - Auto-Negotiation Next Page Transmit Register (ANNPTR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7.15-0 | N/A | Ignored | RO | **0 only.** |

Auto-Negotiation is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register 8 - Auto-Negotiation Next Page Receive Register (ANNPRR)

Auto-Negotiation is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

|  | | | | |
| --- | --- | --- | --- | --- |
| 8.15-0 | N/A | Ignored | RO | **0 only.** |

* 1. Register 9 - 1000Base-T Control Register (GBCRR)

|  | | | | |
| --- | --- | --- | --- | --- |
| 9.15-0 | N/A | Ignored | RO | **0 only.** |

1000Base-T is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register A - 1000Base-T Control Register (GBSR)

|  | | | | |
| --- | --- | --- | --- | --- |
| 10.15-0 | N/A | Ignored | RO | **0 only.** |

1000Base-T is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register B - ?? (????R)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11.15-0 | N/A | Ignored | RO | **0 only.** |

The register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register C - ?? (????R)

|  | | | | |
| --- | --- | --- | --- | --- |
| 12.15-0 | N/A | Ignored | RO | **0 only.** |

The register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register D - MMD Access Control Register (MACR)

|  | | | | |
| --- | --- | --- | --- | --- |
| 13.15-0 | N/A | Ignored | RO | **0 only.** |

The indirect addressing of Clause 42 is not supported in iRail; the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register E - MMD Access Address Data Register (MAADR)

|  | | | | |
| --- | --- | --- | --- | --- |
| 14.15-0 | N/A | Ignored | RO | **0 only.** |

The indirect addressing of Clause 42 is not supported in iRail; the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

* 1. Register F - 1000Base-T Extended Status Register (GBESR)

|  | | | | |
| --- | --- | --- | --- | --- |
| 10.15-0 | N/A | Ignored | RO | **0 only.** |

1000Base-T is not supported, so the register acts as an 'undefined/unsupported' registers, returning 0, and ignoring all writes.

1. MII FPGA Software Interface

This section describes the PHY FPGA software interface. Included are the Memory Map and Register Definitions.

* 1. Memory Map

This section provides the Memory Map for the PHY FPGA. There registers are as follows;



Table 4 FPGA Memory Map

* 1. Interrupts

The Interrupt into the MSS is level sensitive, active high. The PHY SoC interrupts are combined into a single interrupt and presented to the MSS core via MSS Interrupt 0 (FabricIrq0\_IRQn). The sources of interrupts are the following:

* Transmit FIFO Overflow
* Receive FIFO Overflow
* Receive FIFO Underrun
* Transmit Complete
* Collision Detected
* Jabber Detected
  1. Register Definitions

Below are the Register Definition’s for the Comms Module FPGA. Each are describes below.

* + 1. Control Register

The FPGA control register is used to manage board level functions.

Address: 0x10

Access: Read/Write

Reset Value: 0x00000000

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-8 | UNUSED |  |
| 7 | TX FIFO RST | Transmit FIFO Reset. Setting this bit to ‘1’ resets the Transmit FIFO. Must cleared to ‘0’ to take the Transmit FIFO out of reset. |
| 6 | RX\_RST | Receive Reset. Setting this bit to ‘1’ resets the Receive Interface. Must cleared to ‘0’ to take the Receive Interface out of reset. |
| 5 |  |  |
| 4 | INTRNL\_LPBACK | When set to 1 the Manchester transmitter is looped back to the Manchester receiver. |
| 3 | UNUSED | This bit is unused. |
| 2 | UNUSED | This bit is unused. |
| 1 | EXTERNAL\_LPBACK | When set to 1 the Manchester transmitter is looped back to the Manchester receiver externally to the SoC. |
| 0 | UNUSED | This bit is unused. |

Table 5 Control Register Definitions

* + 1. Interrupt Register

The FPGA interrupt register is used to indicate interrupts. All interrupts are cleared when a logic ‘1’ is written to its associated bit.

Address: 0x14

Access: Read Only

Reset Value: 0x00000000

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-8 | UNUSED |  |
| 7 |  |  |
| 6 |  |  |
| 5 |  |  |
| 4 | TX FIFO OVRFLW | Transmit FIFO Overflow Interrupt. When a logic ‘1’, indicates that the transmit FIFO has overflowed. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. |
| 3 |  |  |
| 2 | RX FIFO OVRFLW | Receive FIFO Overflow Interrupt. When a logic ‘1’, indicates that the receive FIFO has overflowed. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. |
| 1 | JABBER DTCT | Jabber Detected Interrupt. When a logic ‘1’, indicates that a jabber was detected. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. |
| 0 | COLLISION DTCT | Collision Detected Interrupt. When a logic ‘1’, indicates that a collision was detected. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. |

Table 6 Interrupt Register Definitions

* + 1. Interrupt Mask Register

The FPGA interrupt mask register is used to manage board level functions.

Address: 0x18

Access: Read/Write

Reset Value: 0x00000000

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-8 | UNUSED |  |
| 7 |  |  |
| 6 |  |  |
| 5 |  |  |
| 4 | TX FIFO OVRFLW *MASK* | Transmit FIFO Overflow Interrupt Mask. When a logic ‘1’, the transmit FIFO has overflow interrupt is masked. When a logic ‘0’, the transmit FIFO has overflow interrupt is enabled. This bit is cleared on a read from this register. |
| 3 |  |  |
| 2 | RX FIFO OVRFLW *MASK* | Receive FIFO Overflow Interrupt Mask. When set to logic ‘1’, the receive FIFO overflow interrupt is masked. When set to logic ‘0’, the receive FIFO overflow interrupt is enabled. This bit is cleared on a read from this register. |
| 1 | JABBER DTCT MASK | Jabber Detected Interrupt Mask. When set to logic ‘1’, the jabber detected interrupt is masked. When set to logic ‘0’, the jabber detected interrupt is enabled. This bit is cleared on a read from this register. |
| 0 | COLLISION DTCT *MASK* | Collision Detected Interrupt Mask. When set to logic ‘1’, the collision detected interrupt is masked. When set to logic ‘0’, the collision detected interrupt is enabled. This bit is cleared on a read from this register. |

Table 7 Interrupt Mask Register Definitions

* + 1. Status Register

The FPGA status register is used to observe board level status.

Address: 0x1C

Access: Read Only

Reset Value: 0x00000005

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-8 | UNUSED |  |
| 7 | UNUSED | This bit is unused and read as logic 0. |
| 6 | UNUSED | This bit is unused and read as logic 0. |
| 5 | UNUSED | This bit is unused and read as logic 0. |
| 4 |  |  |
| 3 | TX FIFO FULL | Logic 1 indicates that the Transmit FIFO is full. Logic 0 indicates that the Transmit FIFO is not full. |
| 2 | TX FIFO EMPTY | Logic 1 indicates that the Transmit FIFO is empty. Logic 0 indicates that the Transmit FIFO is not empty. |
| 1 | RX FIFO FULL | Logic 1 indicates that the Receive FIFO is full. Logic 0 indicates that the Receive FIFO is not full. |
| 0 | RX FIFO EMPTY | Logic 1 indicates that the Receive FIFO is empty. Logic 0 indicates that the Receive FIFO is not empty. |

Table 8 Status Register Definitions

1. Media Access Control (MAC) Functionality

Below are the functions to be performed in the MAC sublayer and therefore not considered for inclusion into the PHY.

* 1. General MAC Functions
* Data encapsulation (transmit and receive)
  + Framing (frame boundary delimitation, frame synchronization)
  + Addressing (handling of source and destination addresses)
  + Error detection (detection of physical medium transmission errors)
  + Frame check sequence generation / checking
* Media Access Management
  + Medium allocation (collision avoidance)
    - deference
    - inter-packet gap
  + Contention resolution (collision handling)
    - collision detection and enforcement
    - collision back-off and retransmission
    - carrier extension
    - packet bursting
  1. Transmitter MAC Functions
* Transmit Data Encapsulation Comprises (4.2.3.1)
* Packet assembly
  + Preamble Generation
  + Start of Frame Delimiter
  + Padding Field
  + Extension Field
* Frame check sequence generation
* Transmit Media Access Management Comprises
* Deference (4.2.3.2.1)
* Inter-packet gap (4.2.3.2.2)
* Collision Detection and Enforcement (4.2.3.2.4)
* Collision Back-off and Retransmission (4.2.3.2.5)
  1. Receiver MAC Functions
* Receive Data Decapsulation Comprises
  + Address Recognition (4.2.4.1.1)
  + Frame Check Sequence Validation (4.2.4.1.2)
  + Frame Disassembly (4.2.4.1.3)
* Receive Media Access Management Comprises
  + Framing (4.2.4.2.1)
  + Collision filtering (4.2.4.2.2)