**Powered iRail**

**System-On-Chip (SOC)**

**Design Specification**

**Revision 1.7**

**October 29, 2016**

Table of Contents

[1. References 5](#_Toc465781796)

[2. Introduction 6](#_Toc465781797)

[2.1. ARM Cortex M3 Processor 7](#_Toc465781798)

[2.1. PPI Proprietary FPGA Fabric 8](#_Toc465781799)

[3. PPI Proprietary FPGA Description 9](#_Toc465781800)

[3.1. Processor Interface 9](#_Toc465781801)

[3.1. FIFOs 10](#_Toc465781802)

[3.1. Transmitter 10](#_Toc465781803)

[3.1.1. Transmit Packet Processor 10](#_Toc465781804)

[3.1.2. Transmit AFE Interface 11](#_Toc465781805)

[3.1. Receiver 12](#_Toc465781806)

[3.1.1. Receive AFE Interface 12](#_Toc465781807)

[3.1.2. Receive Packet Processor 13](#_Toc465781808)

[3.2. Remote Programmability 14](#_Toc465781809)

[3.3. Packet Definition – Physical Layer 15](#_Toc465781810)

[3.4. Packet Definition – Header 15](#_Toc465781811)

[4. FPGA Software Interface 16](#_Toc465781812)

[4.1. Memory Map 16](#_Toc465781813)

[4.2. Initialization 17](#_Toc465781814)

[4.3. Interrupts 17](#_Toc465781815)

[4.4. Register Definitions 18](#_Toc465781816)

[4.4.1. Control Register 18](#_Toc465781817)

[4.4.2. Interrupt Register 19](#_Toc465781818)

[4.4.3. Interrupt Mask Register 20](#_Toc465781819)

[4.4.4. Status Register 21](#_Toc465781820)

[4.4.5. Address Register High 22](#_Toc465781821)

[4.4.6. Address Register Low 22](#_Toc465781822)

[4.5. MSS GPIO Register Definitions 23](#_Toc465781824)

[4.6. Software Operation 25](#_Toc465781825)

[4.6.1. Initialization Procedure 25](#_Toc465781826)

[4.6.2. Packet Transmit Procedure 25](#_Toc465781827)

[4.6.3. Packet Receive Procedure 25](#_Toc465781828)

[5. FPGA Signal Definitions 27](#_Toc465781829)

[6. FPGA Utilization 30](#_Toc465781830)

Figures

[Figure 1 Powered IRail Topology 6](#_Toc465781623)

[Figure 2 PPI Comms Board Electronics 7](#_Toc465781624)

[Figure 3 Arm Cortex M3 Subsystem 8](#_Toc465781625)

[Figure 4 FPGA Fabric Architecture Block Diagram 9](#_Toc465781626)

[Figure 5 FPGA Processor Interface 10](#_Toc465781627)

[Figure 6 Packet Definition - Physical Layer Format 15](#_Toc465781628)

[Figure 7 Packet Definition - Header Format 15](#_Toc465781629)

Tables

[Table 1 FPGA Memory Map 16](#_Toc465781630)

[Table 2 Control Register Definitions 18](#_Toc465781631)

[Table 3 Interrupt Register Definitions 19](#_Toc465781632)

[Table 4 Interrupt Mask Register Definitions 20](#_Toc465781633)

[Table 5 Status Register Definitions 21](#_Toc465781634)

[Table 7 Address Register High Definitions 22](#_Toc465781635)

[Table 8 Address Register High Definitions 22](#_Toc465781636)

[Table 6 MSS GPIO Register Definitions 24](#_Toc465781637)

[Table 12 FPGA Signal Requirements 30](#_Toc465781638)

[Table 13 CPLD Utilization 30](#_Toc465781639)

Revisions Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Description** | **Date** | **Author** |
| 0.1 | Initial Release | 2/17/15 | SHW |
| 0.2 | Update from Design and Simulation | 3/19/15 | SHW |
| 0.3 | Updated from Doug’s comments | 3/21/15 | SHW |
| 0.4 | Added Comms Module Address Register | 3/23/15 | SHW |
| 0.5 | Updated Section 4.2 | 3/26/15 | SHW |
| 0.6 | Added Packet Length operation | 4/7/15 | SHW |
| 0.7 | Updated Address Register Definitions | 4/19/15 | SHW |
| 0.8 | - Added CRC Error to registers.  - Added Force CRC Error. | 4/27/15 | SHW |
| 0.A | Multiple update to include more complete architectural description and detailed software interface. | 4/11/16 | SHW |
| 1.0 |  | 4/18/16 | SHW |
| 1.1 | Removed Application Control Register and Application Status Register and changed these to MSS GPIO registers. | 8/7/16 | SHW |
| 1.2 | Added Support for Eval Board GPIO | 8/10/16 | SHW |
| 1.3 | - Changed Revision Register to 0x21 in Revision Register  - Added use of FabricIrq0\_IRQn into MSS  - Added External Loopback bit in Control Register  - Removed Force CRC Error and Force Collision in Control Register. | 9/7/16 | SHW |
| 1.4 | - Updated Pinouts in Table 9  - Update section 3.4 to correct field sizes. | 9/12/16 | SHW |
| 1.5 | - Updated Figure 7  - Updated Section 4.4.2 | 10/4/16 | SHW |
| 1.6 | - Updated Figure 7 | 10/25/16 | SHW |
| 1.7 | - Added MSS GPIO Register Definitions | 10/30/16 | SHW |

1. References

Below are the references for the iRail FPGA/SOC.

1. “*IGLOO2 FPGA and SmartFusion2 SoC FPGA Datasheet*”, Revision 4, November 2014, Microsemiconductor Corporation.
2. “SmartFusion2 Microcontroller Subsystem User Guide”, Revision 8, December 2014, Microsemiconductor Corporation.
3. Introduction

PPI has a Powered iRail system that supports up to 8 devices on a single iRail as in Figure 1. This iRail provides both power and communication paths to the devices residing on the iRail. These devices consist of simple flash lights to more complex displays and computers. The communication signal and DC power voltage are merged on to the iRail providing a single conductor for both power and communications. Each device interfaces to the iRail via a “COMMs” Board. The COMMs Board provides the electronics that separates power from signal. In addition, it provides the communication intelligence necessary to interface one device to other devices residing on the iRail.



Figure Powered IRail Topology

To understand the concept presented, it is first important to understand the high level implementation of the electronics as in Figure 1. The electrical interface to the iRail is the Analog Front End (AFE) followed by the System On a Chip (SOC).



Figure PPI Comms Board Electronics

The AFE supports both transmission and reception of packets. Transmitted packets are converted from digital data bits to Manchester encoded bits. Received packets are converted from Manchester encoded bits to digital bits.

The SOC (refer to [REF1]) consists of both an Arm Cortex M3 processor and PPI proprietary communication FPGA fabric logic (referred to as FPGA throughout this document).

* 1. ARM Cortex M3 Processor

The ARM Cortex-M3 (referred to as “processor” throughout this document) resides in the SoC and is a low power consumption processor that features low gate count, low interrupt latency, and low-cost debug. It is intended for deeply embedded applications that require optimal interrupt response features. This processor implements the ARM v7-M architecture and is shown in Figure 3. The SmartFusion2 SoC device uses the R2P1 version of the Cortex-M3 core. For more details on the internals like programming model, exception model, instruction set, the Cortex-M3 specific peripherals such as SysTick timer, memory protection unit and others, refer to [REF2].



Figure Arm Cortex M3 Subsystem

The processor sends the commands and receives responses and as a result, executes the necessary functions required for the iRail communications. Additionally, the processor supports any peripherals such as BlueTooth, USB etc. Refer to Figure 3 for the microprocessor subsystem.

* 1. PPI Proprietary FPGA Fabric

The PPI Proprietary FPGA Fabric supports both the transmission and reception of packets and resides between the AFE and processor. Refer to Figure 2 and Figure 4. Transmit packets are first loaded into the Transmit FPGA FIFO. The packet then is converted from bytes to bits, Manchester Encoded, and transmitted over the iRail. When the packet is successfully transmitted, the processor is notified via an interrupt. Received packets are Manchester Decoded, converted from bits to bytes and loaded into the FPGA Receive FIFO. The processor is then notified of a packet reception via an interrupt. Refer to Section 3 for more details on the functions and implementation of the FPGA fabric.

1. PPI Proprietary FPGA Description

The PPI Proprietary FPGA supports both the transmission and reception of packets and resides between the AFE and processor subsystem. A block diagram illustrating the main components of the FPGA is shown below. These functions consist of the Processor Interface, FIFOs, Receive Packet Processor, Receive AFE Interface, Transmit AFE Interface, Transmit Packet Processor and Processor Interface. Refer to Figure 4 below. Details of each block are discussed in the following sections.



Figure FPGA Fabric Architecture Block Diagram

* 1. Processor Interface

The Processor Interface provides the means for the processor to communicate with the FPGA logic. This is 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. Refer to Figure 5. Furthermore, the processor interface supports address decoding, data bus interface, status/control registers and interrupt control.



Figure FPGA Processor Interface

* 1. FIFOs

The FIFOs consist of both a Transmit FIFO and Receive FIFO. The Transmit FIFO provides an 8-bit interface to the AMBA write data interface and an 8-bit interface to the Transmit Packet Processor. The Receive FIFO provides an 8-bit interface to the AMBA read data interface and an 8-bit interface to the Receive Packet Processor. Both the Transmit FIFO and Receive FIFO are 2048 bytes in depth.

* 1. Transmitter

The Transmitter is responsible for accepting packets from the processor (via the Transmit FIFO) and transmitting them on to the iRail. It consists of two primary functions, 1) Transmit Packet Processor and 2) Transmit AFE Interface. Each is described below.

* + 1. Transmit Packet Processor

The Transmit Packet Processor provides the following functions:

* Transmit CRC Generation
* Length Counter initialization
* Transmit State Machine
* Timing Generation
  + - 1. Transmit CRC Generator

The Transmit CRC Generator computes the 16-bit CRC and loads this into the Transmit FIFO at the completion of the packet write from the processor. This is transparent to the software so it doesn’t have to utilize valuable cycles to generate the CRC. The CRC equation is below:

CRC(15:0)=1+x^2+x^15+x^16

The initialization is to all zeros. All the packet bytes are included in the CRC generation including the consumer address, TBD byte, length and data. Refer to Figure 7 for packet definition.

* + - 1. Length Counter

The Length Counter is loaded automatically during a packet write from the AMBA interface. It uses this as a counter to indicate the appropriate time to insert the CRC and terminate the packet transmission.

* + - 1. Transmit State Machine

The Transmit State Machine is responsible for moving data to the TX AFE Interface during the appropriate transmission window. In addition, it handles error conditions and provides interrupts to the processor. Transmit data is loaded into the Transmit FIFO by the processor. The data loaded by the processor includes the Header and Data but not the CRC. The Transmit State Machine provides control to the following functions and discussed in subsequent sections.

* Preamble Generation
* Data from Transmit FIFO
* CRC Generation
* Postamble Generation
  + - 1. Transmit Timing Generation

Transmit timing generation is based off the local 5.00 MHz clock. This clock is generated by the Processor subsystem.

* + 1. Transmit AFE Interface

The Transmit AFE Interface consists of data movement and control required to accept packets from the Transmit FIFO and transmit them on to the iRail. Basic functions are:

* Transmit Enable
* Parallel to Serial Conversion
* Preamble and Postamble Generation
* Manchester Encoding
* Collision Detection

Each is described below.

* + - 1. Transmit Enable

Before the data is transmitted on to the iRail, the Transmit Enable is asserted 15 usec before the data is sent. This is to allow the transmitter AFE to properly charge. Once charged the data can properly interpreted by the receiver.

* + - 1. Transmit Parallel to Serial Conversion

The Transmit Parallel to Serial Conversion converts the parallel data from the Transmit FIFO to serial data. The transmit data is pushed to this logic by the Transmit State Machine.

* + - 1. Preamble and Postamble Generation

The Preamble and Postamble Generation is provided by forcing 1’s onto the iRail. Both indications (Pre and Post Amble) are generated by control of the Transmit State Machine. The Preamble is used to start the clock generation and state machines in preparation of the data arrival at the receiver. It is presently set to 3 bytes with one start byte. The Postamble allows the clock generation and state machines to complete the packet reception. It is presently set to 1 byte in length.

Both the Preamble and Postamble are “generics” in the vhdl code and can therefore be change easily and are defined as the number of “bytes”, not bits.

* + - 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. Collision Detection

The Collision Detection logic monitors the line for collisions. Once detected, the logic notifies the Transmit State Machine. When a collision is detected during a transmission, the Transmit State Machine will terminate the transmission, reset the Transmit FIFO and interrupt the processor. The processor will then be required to queue up the packet again.

* 1. Receiver

The Receiver is responsible for accepting packets from the iRail and pushing them on to the Receive FIFO and then interrupting the processor. It consists of two high level functions, 1) Receive AFE Interface and 2) Receive Packet Processor. Each is described below.

* + 1. Receive AFE Interface

The Receive AFE Interface consists of data manipulation and associated control required to accept packets from the iRail and push them to the Receive Packet Processor. Basic functional blocks are below and described in subsequent sections.

1. Receive Timing Recovery
2. Manchester Decoding
3. Serial to Parallel Conversion
4. Preamble Detection
5. AFE State Machine
6. Collision Detection
   * + 1. Receive 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 81.25 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 81.25 MHz clocks, thus realigning the data and sampling point.

* + - 1. Manchester Decoding

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. Receive Serial to Parallel Conversion

The Receive Serial to Parallel Conversion converts the serial data from the Manchester Decoder to parallel data and then pushed to the preamble detection logic.

* + - 1. Preamble Detection

The Preamble Detection logic discovers the preamble pattern on the iRail. When the preamble is detected it provides this indication to the Receive State Machine. This allows the Receive State Machine to find the beginning of packet and thus the alignment of bits into bytes.

* + - 1. AFE State Machine

The AFE State Machine coordinates the reception of data from the AFE to the RX Packet Processor. It starts when an edge is detected on the iRail. Then looks for the start byte and when detected, coordinates the byte construction before passing off to the RX State Machine.

* + - 1. Receive Collision Detection

When data is being received, the transceiver is monitored for violations of the Manchester coding/pulse width. If violations are detected they may be due to either collisions (multiple transmitters accessing the bus) or error conditions. Since error conditions are unlikely (low noise environment) other than discrete shock events (ESD, mechanical disconnect due to firing), any violation detected will be treated as a collision with the remainder of a packet being disregarded, the Receive FIFO flushed and an interrupt sent to the processor.

* + 1. Receive Packet Processor

The Receive Packet Processor consists of data and control required to accept packets from the Receive AFE Interface and push them to the Receive FIFO. Basic functional blocks are below and described in subsequent sections.

1. Receive State Machine
2. Length Counter
3. Address Detection
4. CRC Checker
5. Collision Detection
   * + 1. Receive State Machine

The Receive State Machine is responsible for moving received data from the Receive AFE Interface logic to the Receive FIFO. In addition, it handles error condition and provide interrupts to the processor. The Receive State Machine detects the sync byte, aligns the bits into bytes, and moves data to the Receive FIFO based on the length bytes received. The address, length and CRC are included in the Receive FIFO.

* + - 1. Length Counter

The Length Counter is loaded into a register when a valid packet address is detected. This length counter can then indicate when the address is available, length is available and CRC is available.

* + - 1. Address Detection

The Receive State Machine uses the Address Detection logic to determine if the packet being received is destined for this device. The software must initialize the expected address into the Address Registers. Refer to Section 4.4.5 Address Register High and Section 4.4.6 Address Register Low.

* + - 1. CRC Checker

The CRC Checker compares the CRC provided by the packet received against the CRC generated locally by this logic. If the CRCs do not match, the reception continues and an interrupt is sent to the processor.

* 1. Remote Programmability

The ability to change the FPGA program while operating in circuit is predominantly a software task. Basic requirements include:

1. There must be sufficient Flash memory to store 2 complete FPGA loads. This can be either in Flash attached to the FPGA or in Flash attached to the processor if it is programming the FPGA directly.
2. The processor must be configured to be able to program either the flash attached/embedded in the FPGA or to program the FPGA directly from its own Flash.

The complete image must be downloaded and verified by the processor. File download is a function of the end to end protocols supported by the processor. The file fragments will be passed over the iRail using the same process as for any other packets.

Once the complete image has been downloaded and verified for correctness, the FPGA can be programmed from that file. Note that a golden version should be maintained in case of interruption of the update process or other event requiring a known good build to be loaded.

* 1. Packet Definition – Physical Layer

Below is the physical layer definition of the packet. It is comprised of a Preamble, start bit, data and postamble.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Preamble = 1s  N \* 8 bits | Start  Byte | Data  M \* 8 bits | CRC  2 Bytes | Postamble = 1s  K \* 8 bits |

Figure Packet Definition - Physical Layer Format

Where:

N = VHDL Generic (PREAMBLE\_LENGTH)

M = 1 to 1024

K = VHDL Generic (POSTAMBLE\_LENGTH)

**THE PACKET SIZE MUST BE A MINIMUM OF 8 BYTES (WHICH INCLUDES THE HEADER and LENGTH BYTES of quantity 4).**  This includes the Consumer Address (2 bytes), TBD+Packet Length (2 bytes) and CRC (2 bytes) consequently, the data is a minimum of 2 bytes.

* 1. Packet Definition – Header

Below is the header definition of the packet. The header is embedded within the Data as shown in Figure 6 above.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2 Bytes** | | **2 Bytes** | | **(M-6) Bytes** | **2 Bytes** |
| TBD  5-Bits | Consumer Address 11-Bits | TBD  5-Bits | Packet Length  11-Bits | Data | CRC |

Figure Packet Definition - Header Format

1. FPGA Software Interface

This section describes the FPGA software interface for the Powered IRail Comms Module. Included are the Memory Map and Register Definitions.

* 1. Memory Map

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



Table FPGA Memory Map

* 1. Initialization

When initially powered on the iRail, a CPLD will be immediately ready for operation. Signals allowing communication on the iRail must be initially disabled until the processor completes its boot cycle and indicates it is ready to operate.

For the FPGA, when initially powered it must be programmed either from an attached Flash device. In either case, all active signals must be disabled before the device is fully programmed and the processor is ready to operate

* 1. Interrupts

The Interrupt into the MSS is level sensitive, active high. The 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
* Transmit FIFO Underrun
* Receive FIFO OverFlow
* Receive FIFO Underrun
* Transmit Complete
* Receiver Packet Available
* Receiver CRC Error
* Collision 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 | TX\_FIFO\_XMT | Transmit FIFO Transmit. Setting this bit to ‘1’ starts the Transmit FIFO transmitting. This bit will clear when the entire packet is sent. |
| 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 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 | *TX PKT CMPLT* | Transmit Packet Complete Interrupt. When a logic ‘1’, indicates that the transmit packet has data to be sent over the iRail. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. This interrupt will not be asserted if a transmit error condition occurs (TX FIFO Underrun, TX FIFO Overflow, Collision Detect) |
| 6 | RX PKT AVAIL | Receive Packet Available Interrupt. When a logic ‘1’, indicates that a receive packet is available for consumption. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. This interrupt will not be asserted if a transmit error condition occurs (RX FIFO Underrun, RX FIFO Overflow, RX CRC Error) |
| 5 | TX FIFO UNDRUN | Transmit FIFO Under run Interrupt. When a logic ‘1’, indicates that the transmit FIFO has underrun. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit. |
| 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 | RX FIFO UNDRUN | Receive FIFO Under-run Interrupt. When a logic ‘1’, indicates that the receive FIFO has underrun. An interrupt is also generated. Cleared when a logic ‘1’ is written to this bit.. |
| 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 | RX CRC ERROR | Receive CRC Error Interrupt. When a logic ‘1’, indicates that a Receive CRC Error has occurred. 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 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 | *TX PKT CMPLT MASK* | Transmit Packet Complete Interrupt Mask. When set to logic ‘1’, the transmit packet data complete interrupt is masked. When set to logic ‘0’, the transmit packet data complete interrupt is enabled. This bit is cleared on a read from this register. |
| 6 | RX PKT AVAIL *MASK* | Receive Packet Available Interrupt Mask. When set to logic ‘1’, the receive packet interrupt is masked. When set to logic ‘0’, the receive packet interrupt is enabled. This bit is cleared on a read from this register. |
| 5 | TX FIFO UNDRUN *MASK* | Transmit FIFO Under run Interrupt Mask. When set to logic ‘1’, the transmit FIFO has underrun interrupt is masked. When set to logic ‘0’, the transmit FIFO has underrun interrupt is enabled. This bit is cleared on a read from this register. |
| 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 | RX FIFO UNDRUN *MASK* | Receive FIFO Under-run Interrupt Mask. When set to logic ‘1’, the receive FIFO underrun interrupt is masked. When set to logic ‘0’, the receive FIFO underrun interrupt is enabled. This bit is cleared on a read from this register. |
| 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 | RX CRC ERROR MASK | Receive CRC Error Interrupt Mask. When set to logic ‘1’, the Receive CRC Error Interrupt is masked. When set to logic ‘0’, the Receive CRC Error 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 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 | UNUSED | This bit is unused and read as logic 0. |
| 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 Status Register Definitions

* + 1. Address Register High

The FPGA address register is houses the high address (consumer) for the Comms Module. It is used to compare receive packets. If there is a match between the receive packet and this address register, the packet is destined for this Comms Module. Note there are 4 of these registers, which allows for 4 high address (consumer) for the Comms Module

**THE PACKET SIZE MUST BE A MINIMUM OF 8 BYTES (WHICH INCLUDES THE HEADER and LENGTH BYTES of quantity 4).**  This includes the Consumer Address (2 bytes), TBD+Packet Length (2 bytes) and CRC (2 bytes) consequently, the data is a minimum of 2 bytes.

Address: 0x30, 0x38, 0x40, 0x48

Access: Read/Write

Reset Value: 0x00000000

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-2 | UNUSED |  |
| 1 | COMMS\_ADDR(9:8) | This register encodes the local Comms module address. It is used to compare with the address field in the receive packets. It there is a match, then the packet is destined for this Comms module. |
| 0 |

Table Address Register High Definitions

* + 1. Address Register Low

The FPGA address register is houses the Low address (consumer) for the Comms Module. It is used to compare receive packets. If there is a match between the receive packet and this address register, the packet is destined for this Comms Module. Note there are 4 of these registers, which allows for 4 low address (consumer) for the Comms Module

**THE PACKET SIZE MUST BE A MINIMUM OF 8 BYTES (WHICH INCLUDES THE HEADER and LENGTH BYTES of quantity 4).**  This includes the Consumer Address (2 bytes), TBD+Packet Length (2 bytes) and CRC (2 bytes) consequently, the data is a minimum of 2 bytes.

Address: 0x34, 0x3C, 0x44, 0x4C

Access: Read/Write

Reset Value: 0x00000000

Contents:

|  |  |  |
| --- | --- | --- |
| ***Bit*** | ***Name*** | **Description** |
| 31-8 | UNUSED |  | |
| 7 | COMMS\_ADDR(7:0) | This register encodes the local Comms module address. It is used to compare with the address field in the receive packets. It there is a match, then the packet is destined for this Comms module. |
| 6 |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |
| 0 |

Table Address Register High Definitions

* 1. MSS GPIO Register Definitions

The Application Specific Registers utilize the General Purpose IO (GPIO) build in to the Microprocessor Sub-System (MSS) of the SmartFusion2 SoC. The MSS provides one GPIO hard peripheral supporting 32 General Purpose I/Os. If a GPIO is to be utilized it must be enabled on the within the MSS.

.

Following are the features of the MSS GPIO block:

* 32 individually configurable GPIOs
* Each GPIO is dynamically programmable as an input, output, or bi-directional I/O within the MSS. The software must be aware of this configuration. See below as an example.
* Each GPIO can be configured as an interrupt source to the ARM® Cortex® -M3 processor in Input mode
* The reset state of the GPIOs is configurable
* The GPIOs can be selectively reset by either the hard reset (power-on reset, user reset from the fabric) or the soft reset from the SYSREG block

Each board will have its own application specific configuration of the GPIO. The generic (non-board specific) register definitions are below. For details on configuring the GPIO from both a hardware and software standpoint refer to [SmartFusion2 MSS GPIO Configuration Guide](http://www.microsemi.com/document-portal/doc_download/135151-smartfusion2-mss-gpio-configuration-guide) and [UG0331: SmartFusion2 Microcontroller Subsystem User Guide](http://www.microsemi.com/document-portal/doc_download/130918-ug0331-smartfusion2-microcontroller-subsystem-user-guide) section 15.

Address: 0x40013000

Access: Read/Write

Reset Value: 0x00000000

| ***GPIO #*** | ***Schematic Name*** | ***Type*** | **Description** |
| --- | --- | --- | --- |
| 0 | GP0B | BIDI | Board Specific. |
| 1 | GP1Bf | OUT | Board Specific. |
| 2 | ID-RES2 | IN | This bit along with the other three ID-RES bits specify the board type. Refer to the definition in Confluence [page](https://protoprodinc.atlassian.net/wiki/display/IRAIL/Node+ID+Strapping+Definitions). |
| 3 | SPI-FLASH-IO2 | BIDI | Write protect for SPI Flash. Clearing to logic 0 write protects. Setting to logic 1 allow writes to the SPI Flash. |
| 4 | SPI-FLASH-IO3 | BIDI | When set to logic 0, Hold (pause) serial transfer in single bit or Dual data commands. When set to logic 1, continue transfers. |
| 5 | GP5Af | OUT | Board Specific. |
| 6 | GP6Af | OUT | Board Specific. |
| 7 | GP7Af | OUT | When set to logic 0, Hold (pause) serial transfer in single bit or Dual data commands. When set to logic 1, continue transfers. |
| 8 | GP8Af | OUT | Board Specific. |
| 9 | ID-RES0 | IN | This bit along with the other three ID-RES bits specify the board type. Refer to the definition in Confluence [page](https://protoprodinc.atlassian.net/wiki/display/IRAIL/Node+ID+Strapping+Definitions). |
| 10 | ID-RES3 | IN | This bit along with the other three ID-RES bits specify the board type. Refer to the definition in Confluence [page](https://protoprodinc.atlassian.net/wiki/display/IRAIL/Node+ID+Strapping+Definitions). |
| 11 | GP11Af | IN | Board Specific. |
| 12 | GP12B | BIDI | Board Specific. |
| 13 | GP13Af | OUT | Board Specific. |
| 14 | GREENn | BIDI | Set to logic 1 to turn off the green LED. Clear to logic 0 to turn on green LED. |
| 15 | REDn | BIDI | Set to logic 1 to turn off the red LED. Clear to logic 0 to turn on red LED. |
| 16 | BLUEn | BIDI | Set to logic 1 to turn off the blue LED. Clear to logic 0 to turn on blue LED. |
| 17 | GP17A | BIDI | Board Specific. |
| 18 | GP18A | BIDI | Board Specific. |
| 19 | ID-RES1 | IN | This bit along with the other three ID-RES bits specify the board type. Refer to the definition in Confluence [page](https://protoprodinc.atlassian.net/wiki/display/IRAIL/Node+ID+Strapping+Definitions). |
| 20 | SPI-FLASH-RSTn | OUT | When set to logic 0, SPI Flash is **out** of reset. When set to logic 1 SPI Flash is **in** reset. |
| 21 | GP21Bf | OUT | Board Specific. |
| 22 | GP22Bf | OUT | Board Specific. |
| 23 |  |  | **Not Used** |
| 24 | GP24Bf | OUT | Board Specific. |
| 25 | GP25A | BIDI | Board Specific. |
| 26 | GP26A | BIDI | Board Specific. |
| 27 |  |  | **Not Used** |
| 28 | FPGA Reset | OUT | Setting this bit to ‘1’ resets the FPGA Logic. Clearing this bit takes the FPGA logic out of reset. |
| 29 |  |  | **Not Used** |
| 30 |  |  | **Not Used** |
| 31 | GP31A | BIDI | Board Specific. |

Table 8 MSS GPIO Register Definitions











* 1. Software Operation

Below the software operation for packet transmit and receive sequences are described below. Included are the potential error conditions.

* + 1. Initialization Procedure

The FPGA must first be initialized before packet communications can begin.

1. Control Register – Ensure that the transmitter and receiver are out of reset
2. Interrupt Register – Ensure that all interrupts are cleared by writing a logic 1 to any active interrupts.
3. Interrupt Mask Register – Ensure that the chosen interrupts are enabled.
4. **Error! Reference source not found.** – This register will be device specific and will be defined on a per device basis.
5. Address Register High – Configure the high order two bits of the address.
6. Address Register Low – Configure the lower byte of the address.
   * 1. Packet Transmit Procedure

To transmit a packet the following steps must be completed in the order described below.

1. Fill the Transmit FIFO with the desired data.
2. Ensure to put the length count as specified in section 3.4.
3. Set the TX\_FIFO\_XMT in Section 4.4.1 Control Register.
4. The TX\_FIFO\_XMT bit is automatically cleared when the all the data has been transmitted.
5. A Transmit Complete interrupt is generated when the transfer is complete if not masked. Refer to the Interrupt Register in Section 4.4.2, bit “*TX PKT CMPLT*”.
6. At this point, the Transmit FIFO can be loaded with more data.

Several error conditions can result from a packet transmission. They include the following:

* Transmit FIFO Not Empty – NOT YET IMPLEMENTED. If the processor attempts to start a transmission when the Transmit FIFO is not empty, the transmission will wait until the FIFO is empty.
* Collision Detected - If a collision is detected, the Transmission terminates, the Transmit FIFO is flushed, and an interrupt provided to the processor. Refer to the Interrupt Register in Section 4.4.2.
  + 1. Packet Receive Procedure

To receive a packet the following steps must be completed in the order described below.

1. A Receive Packet Available interrupt will be generated if not masked. Refer to Section 4.4.2 Interrupt Register and Section 4.4.3 Interrupt Mask Register.
2. Read the length indication from the packet in the Receive FIFO, refer to 4.1 Memory Map for address location.
3. Read the remaining bytes from the Receive FIFO.

Several error conditions can result from a packet reception. They include the following:

* Collision Detected – NOT YET IMPLEMENTED.
* Address Mismatch – If the address of the packet does not match the receivers address, then the reception is terminated. There is no need to notify the processor of this event since it is intended to filter out packets bound for other devices.
* CRC Error – If a CRC error is detected, the processor is interrupted and must either reset the Receive FIFO or pull out the data for inspection. Refer to the Interrupt Register in Section 4.4.2.

1. FPGA Signal Definitions

This section provides FPGA signal definitions. See Table 9 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Port** | **Pin** | **Fixed** | **Function** |
| atck | L9 | Fixed | JTAG\_TCK/M3\_TCK |
| atdi | N9 | Fixed | JTAG\_TDI/M3\_TDI |
| atdo | R7 | Fixed | JTAG\_TDO/M3\_TDO/M3\_SWO |
| atms | AA3 | Fixed | JTAG\_TMS/M3\_TMS/M3\_SWDIO |
| atrstb | Y3 | Fixed | JTAG\_TRSTB/M3\_TRSTB |
| BLUEn | E20 | No | MSIO130NB9 |
| Data\_FAIL | L2 | No | MSIO24NB4/SPI\_1\_SS3/GPIO\_16\_A |
| DEVRST\_N | M9 | Fixed | DEVRST\_N |
| DRVR\_EN | W2 | Yes | MSIO5PB4 |
| GPIO\_0\_BI | J2 | Yes | MSIO33NB3/GPIO\_0\_B |
| GPIO\_1\_M2F | J1 | Yes | MSIO38PB3/GPIO\_9\_B |
| GPIO\_2\_IN | V1 | Yes | MSIO7NB4/CAN\_TX/GPIO\_2\_A/USB\_DATA0\_A |
| GPIO\_3\_BI | T5 | Yes | MSIO8PB4/CAN\_RX/GPIO\_3\_A/USB\_DATA1\_A |
| GPIO\_4\_BI | U5 | Yes | MSIO8NB4/CAN\_TX\_EN\_N/GPIO\_4\_A/USB\_DATA2\_A |
| GPIO\_5\_M2F | H2 | Yes | MSIO38NB3/GPIO\_10\_B |
| GPIO\_6\_M2F | H5 | Yes | MSIO35PB3/GPIO\_3\_B |
| GPIO\_7\_M2F | H4 | Yes | MSIO35NB3/GPIO\_4\_B |
| GPIO\_8\_M2F | J4 | Yes | MSIO30PB3 |
| GPIO\_9\_IN | N1 | Yes | MSIO22NB4/SPI\_0\_SS2/GPIO\_9\_A/USB\_DATA6\_A |
| GPIO\_10\_IN | M4 | Yes | MSIO23PB4/SPI\_0\_SS3/GPIO\_10\_A/USB\_DATA7\_A |
| GPIO\_11\_M2F | K2 | Yes | MSIO30NB3 |
| GPIO\_12\_BI | G5 | Yes | MSIO39NB2/MMUART\_1\_DTR/GPIO\_12\_B |
| GPIO\_13\_M2F | L7 | Yes | MSIO27PB3 |
| GPIO\_14\_BI | G1 | Yes | MSIO40NB2/MMUART\_1\_DSR/GPIO\_14\_B |
| GPIO\_15\_BI | F2 | Yes | MSIO41PB2/MMUART\_1\_RI/GPIO\_15\_B/CCC\_NE1\_CLKI1 |
| GPIO\_16\_BI | F3 | Yes | MSIO41NB2/MMUART\_1\_DCD/GPIO\_16\_B |
| GPIO\_17\_BI | P2 | Yes | MSIO18PB4/SPI\_1\_SS4/GPIO\_17\_A |
| GPIO\_18\_BI | R2 | Yes | MSIO18NB4/SPI\_1\_SS5/GPIO\_18\_A |
| GPIO\_19\_IN | T2 | Yes | MSIO14PB4/SPI\_0\_SS4/GPIO\_19\_A |
| GPIO\_20\_OUT | T1 | Yes | MSIO14NB4/SPI\_0\_SS5/GPIO\_20\_A |
| GPIO\_21\_M2F | K7 | Yes | MSIO27NB3 |
| GPIO\_22\_M2F | K5 | Yes | MSIO26PB3 |
| GPIO\_24\_M2F | K4 | Yes | MSIO26NB3 |
| GPIO\_25\_BI | P1 | Yes | MSIO20PB4/GPIO\_25\_A/GB9/VCCC\_SE0\_CLKI |
| GPIO\_26\_BI | N2 | Yes | MSIO20NB4/GPIO\_26\_A/GB13/VCCC\_SE1\_CLKI |
| GPIO\_31\_BI | K1 | Yes | MSIO33PB3/GPIO\_31\_A |
| GREENn | D21 | No | MSIO120NB9 |
| I2C\_1\_SCL | R5 | Yes | MSIO11NB4/I2C\_1\_SCL/GPIO\_1\_A/USB\_DATA4\_A/CCC\_NE1\_CLKI0 |
| I2C\_1\_SDA | R4 | Yes | MSIO11PB4/I2C\_1\_SDA/GPIO\_0\_A/USB\_DATA3\_A/CCC\_NE0\_CLKI0 |
| MAC\_MII\_COL | L19 | Yes | MSIO123PB9/GB6/CCC\_NW1\_CLKI1 |
| MAC\_MII\_CRS | M20 | Yes | MSIO124NB9 |
| MAC\_MII\_MDC | F18 | Yes | MSIO118PB9 |
| MAC\_MII\_MDIO | G17 | Yes | MSIO118NB9 |
| MAC\_MII\_RX\_CLK | M18 | Yes | MSIO123NB9 |
| MAC\_MII\_RX\_DV | L20 | Yes | MSIO124PB9/GB2/CCC\_NW0\_CLKI1 |
| MAC\_MII\_RX\_ER | E17 | Yes | MSIO128NB9 |
| MAC\_MII\_RXD[0] | L18 | Yes | MSIO125NB9 |
| MAC\_MII\_RXD[1] | L17 | Yes | MSIO125PB9/CCC\_NW1\_CLKI0 |
| MAC\_MII\_RXD[2] | K18 | Yes | MSIO126NB9 |
| MAC\_MII\_RXD[3] | K17 | Yes | MSIO126PB9/CCC\_NW0\_CLKI0 |
| MAC\_MII\_TX\_CLK | J18 | Yes | MSIO135NB9 |
| MAC\_MII\_TX\_EN | F20 | Yes | MSIO131PB9 |
| MAC\_MII\_TX\_ER | F19 | Yes | MSIO131NB9 |
| MAC\_MII\_TXD[0] | G18 | Yes | MSIO132NB9 |
| MAC\_MII\_TXD[1] | H17 | Yes | MSIO132PB9 |
| MAC\_MII\_TXD[2] | G20 | Yes | MSIO134NB9 |
| MAC\_MII\_TXD[3] | G21 | Yes | MSIO134PB9 |
| MANCH\_OUT\_N | Y1 | Yes | MSIO2NB4 |
| MANCH\_OUT\_P | AA2 | Yes | MSIO2PB4 |
| MANCHESTER\_IN | Y2 | Yes | MSIO5NB4 |
| MDDR\_ADDR[0] | A4 | Yes | DDRIO58NB1/MDDR\_ADDR0 |
| MDDR\_ADDR[1] | E7 | Yes | DDRIO57PB1/MDDR\_ADDR1 |
| MDDR\_ADDR[2] | E6 | Yes | DDRIO57NB1/MDDR\_ADDR2 |
| MDDR\_ADDR[3] | A5 | Yes | DDRIO56PB1/MDDR\_ADDR3 |
| MDDR\_ADDR[4] | B5 | Yes | DDRIO56NB1/MDDR\_ADDR4 |
| MDDR\_ADDR[5] | D5 | Yes | DDRIO55PB1/MDDR\_ADDR5 |
| MDDR\_ADDR[6] | D6 | Yes | DDRIO55NB1/MDDR\_ADDR6 |
| MDDR\_ADDR[7] | A2 | Yes | DDRIO54NB1/MDDR\_ADDR7 |
| MDDR\_ADDR[8] | F5 | Yes | DDRIO53PB1/MDDR\_ADDR8 |
| MDDR\_ADDR[9] | E5 | Yes | DDRIO53NB1/MDDR\_ADDR9 |
| MDDR\_ADDR[10] | B2 | Yes | DDRIO52PB1/MDDR\_ADDR10 |
| MDDR\_ADDR[11] | B1 | Yes | DDRIO52NB1/MDDR\_ADDR11 |
| MDDR\_ADDR[12] | E4 | Yes | DDRIO51PB1/MDDR\_ADDR12 |
| MDDR\_ADDR[13] | D4 | Yes | DDRIO51NB1/MDDR\_ADDR13 |
| MDDR\_ADDR[14] | C1 | Yes | DDRIO50PB1/MDDR\_ADDR14 |
| MDDR\_ADDR[15] | C2 | Yes | DDRIO50NB1/MDDR\_ADDR15 |
| MDDR\_BA[0] | C6 | Yes | DDRIO59PB1/MDDR\_BA0 |
| MDDR\_BA[1] | B6 | Yes | DDRIO59NB1/MDDR\_BA1 |
| MDDR\_BA[2] | B4 | Yes | DDRIO58PB1/MDDR\_BA2 |
| MDDR\_CAS\_N | D8 | Yes | DDRIO61NB1/MDDR\_CAS\_N |
| MDDR\_CKE | G9 | Yes | DDRIO62PB1/MDDR\_CKE |
| MDDR\_CLK | B7 | Yes | DDRIO60PB1/MDDR\_CLK |
| MDDR\_CLK\_N | A7 | Yes | DDRIO60NB1/MDDR\_CLK\_N |
| MDDR\_CS\_N | G10 | Yes | DDRIO62NB1/MDDR\_CS\_N |
| MDDR\_DM\_RDQS[0] | A12 | Yes | DDRIO72PB1/MDDR\_DM\_RDQS0 |
| MDDR\_DM\_RDQS[1] | D11 | Yes | DDRIO66NB1/MDDR\_DM\_RDQS1 |
| MDDR\_DQ[0] | D13 | Yes | DDRIO75PB1/MDDR\_DQ0 |
| MDDR\_DQ[1] | E13 | Yes | DDRIO75NB1/MDDR\_DQ1 |
| MDDR\_DQ[2] | D14 | Yes | DDRIO74PB1/MDDR\_DQ2 |
| MDDR\_DQ[3] | D15 | Yes | DDRIO74NB1/MDDR\_DQ3 |
| MDDR\_DQ[4] | A13 | Yes | DDRIO72NB1/MDDR\_DQ4 |
| MDDR\_DQ[5] | A14 | Yes | DDRIO71PB1/MDDR\_DQ5 |
| MDDR\_DQ[6] | B14 | Yes | DDRIO71NB1/MDDR\_DQ6 |
| MDDR\_DQ[7] | G11 | Yes | DDRIO70PB1/MDDR\_DQ7 |
| MDDR\_DQ[8] | B12 | Yes | DDRIO69PB1/MDDR\_DQ8 |
| MDDR\_DQ[9] | B11 | Yes | DDRIO69NB1/MDDR\_DQ9 |
| MDDR\_DQ[10] | E11 | Yes | DDRIO68PB1/MDDR\_DQ10/CCC\_NE0\_CLKI2 |
| MDDR\_DQ[11] | E12 | Yes | DDRIO68NB1/MDDR\_DQ11 |
| MDDR\_DQ[12] | D9 | Yes | DDRIO65PB1/MDDR\_DQ12/GB12/CCC\_NE1\_CLKI2 |
| MDDR\_DQ[13] | E10 | Yes | DDRIO65NB1/MDDR\_DQ13 |
| MDDR\_DQ[14] | B9 | Yes | DDRIO64PB1/MDDR\_DQ14/CCC\_NE1\_CLKI3 |
| MDDR\_DQ[15] | A10 | Yes | DDRIO64NB1/MDDR\_DQ15 |
| MDDR\_DQS[0] | A15 | Yes | DDRIO73PB1/MDDR\_DQS0 |
| MDDR\_DQS[1] | C11 | Yes | DDRIO67PB1/MDDR\_DQS1/GB8/CCC\_NE0\_CLKI3 |
| MDDR\_DQS\_TMATCH\_0\_IN | D10 | Yes | DDRIO66PB1/MDDR\_TMATCH\_0\_IN |
| MDDR\_DQS\_TMATCH\_0\_OUT | G13 | Yes | DDRIO70NB1/MDDR\_TMATCH\_0\_OUT |
| MDDR\_ODT | A3 | Yes | DDRIO54PB1/MDDR\_ODT |
| MDDR\_RAS\_N | A8 | Yes | DDRIO63PB1/MDDR\_RAS\_N |
| MDDR\_RESET\_N | E8 | Yes | DDRIO61PB1/MDDR\_RESET\_N |
| MDDR\_WE\_N | A9 | Yes | DDRIO63NB1/MDDR\_WE\_N |
| MMUART\_0\_RXD\_F2M | D1 | Yes | MSIO43PB2/MMUART\_1\_CLK/GPIO\_25\_B/GB14/VCCC\_SE1\_CLKI |
| MMUART\_0\_TXD\_M2F | F1 | Yes | MSIO42PB2/GB10/VCCC\_SE0\_CLKI |
| MMUART\_1\_CTS | G2 | Yes | MSIO40PB2/MMUART\_1\_CTS/GPIO\_13\_B/CCC\_NE0\_CLKI1 |
| MMUART\_1\_RTS | G4 | Yes | MSIO39PB2/MMUART\_1\_RTS/GPIO\_11\_B |
| MMUART\_1\_RXD | D2 | Yes | MSIO43NB2/MMUART\_1\_RXD/GPIO\_26\_B |
| MMUART\_1\_TXD | E1 | Yes | MSIO42NB2/MMUART\_1\_TXD/GPIO\_24\_B |
| RCVR\_EN | W1 | Yes | MSIO7PB4 |
| REDn | U4 | No | MSIO4PB4 |
| SPI\_0\_CLK | U2 | Yes | MSIO12PB4/SPI\_0\_CLK/USB\_XCLK\_A |
| SPI\_0\_DI | U1 | Yes | MSIO12NB4/SPI\_0\_SDI/GPIO\_5\_A/USB\_DIR\_A |
| SPI\_0\_DO | P4 | Yes | MSIO13PB4/SPI\_0\_SDO/GPIO\_6\_A/USB\_STP\_A |
| SPI\_0\_SS0 | P5 | Yes | MSIO13NB4/SPI\_0\_SS0/GPIO\_7\_A/USB\_NXT\_A |
| SPI\_0\_SS1\_0 | M1 | Yes | MSIO22PB4/SPI\_0\_SS1/GPIO\_8\_A/USB\_DATA5\_A |
| SPI\_1\_CLK | N4 | Yes | MSIO16PB4/SPI\_1\_CLK |
| SPI\_1\_DI | N7 | Yes | MSIO16NB4/SPI\_1\_SDI/GPIO\_11\_A |
| SPI\_1\_DO | M5 | Yes | MSIO17PB4/SPI\_1\_SDO/GPIO\_12\_A |
| SPI\_1\_SS0 | L5 | Yes | MSIO17NB4/SPI\_1\_SS0/GPIO\_13\_A |
| SPI\_1\_SS1\_0 | L3 | Yes | MSIO23NB4/SPI\_1\_SS1/GPIO\_14\_A |
| TEST\_TX | T3 | No | MSIO4NB4 |
| XTL | AA4 | Fixed | XTLOSC\_MAIN\_XTAL |

Table FPGA Signal Requirements

1. FPGA Utilization

Table 10 below provides the FPGA utilization.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Used** | **Total** | **Percentage** |
| 4LUT | 1004 | 6060 | 16.57 |
| DFF | 753 | 6060 | 12.43 |
| I/O Register | 0 | 477 | 0.00 |
| User I/O | 24 | 159 | 15.09 |
| -- Single-ended I/O | 24 | 159 | 15.09 |
| -- Differential I/O Pairs | 0 | 79 | 0.00 |
| RAM64x18 | 0 | 11 | 0.00 |
| RAM1K18 | 2 | 10 | 20.00 |
| MACC | 0 | 11 | 0.00 |
| Chip Globals | 8 | 8 | 100.00 |
| CCC | 1 | 2 | 50.00 |
| RCOSC\_25\_50MHZ | 0 | 1 | 0.00 |
| RCOSC\_1MHZ | 0 | 1 | 0.00 |
| XTLOSC | 1 | 1 | 100.00 |
| MSS | 1 | 1 | 100.00 |

Table CPLD Utilization