

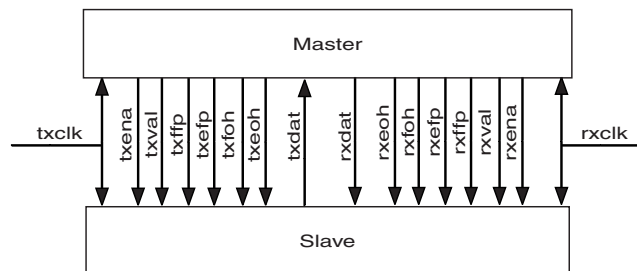
## Features

- High-capacity synchronous bus for carrying isochronous traffic
- Over 2.4 Gigabits per second (Gbps) full-duplex capacity
- Fully synchronous operation
- Byte-level transfer
- Scatter/gather across multiple byte lanes
- Master-controlled dynamic bus sizing, and data rate
- Multiplexing of multiple slave data streams
- Supports multiplex framing formats, including overhead
- Independent transmit and receive clocks
- Single edge clocking

## Topology

Figure 1 shows the general arrangement of the Midbus. It has a single master, and one or more slaves. The master drives the control lines, and txdat signals. The slave drives the rxdat signals.

**Figure 1. General Arrangement**



This document does not specify the exact names of the Midbus signals. A specific instance of a Midbus interface would typically prefix its names with an identifier such as "m", to form "mtxdat", "mrxena", etc.

### Functional Description

A typical application of the Midbus interface is connecting a cell, or packet processor to a Digital Signal level 3 (DS3), or Synchronous Optical Network (SONET) framer/deframer. The Midbus data bus is divided into one, two, or four byte lanes; all are eight bits wide. It supports only byte-oriented protocols, such as: SONET, Packet over SONET (POS), and Asynchronous Transfer Mode (ATM), etc.

The Midbus allows multiple slaves. The master broadcasts all control signals (except for per-slave enable signals), as well as the receive data lines to all slaves. The slaves drive zeros on their transmit data buses during the bus cycles they are not enabled. The master's transmit data bus input is the bitwise-OR of all the slaves' data bus outputs.

The Midbus transfers streams of bytes under the control of a Midbus master. Enable signals allow the master to start and stop the data flow dynamically, and regulate the data throughput. In a two- and four-lane Midbus, the master can also transfer data on a subset of the receive byte lanes (scatter), and transmit byte lanes (gather). The enable and scatter/gather features allow the master to introduce gaps in the payload data stream to accommodate low data rate data streams, framing, and overhead bytes.

The Midbus also supports framed Time Division Multiplexed (TDM) data, including: frame positions, and non-payload data. Examples of this type of traffic include SONET frames. This is intended for applications where the master and slave both process all or part of the TDM frame over and above the payload. The extra framing information is carried in signals indicating fixed, and embedded frame positions; and overhead.

The master and slave must agree on the frame structure beforehand. The various marker signals are used to establish initial synchronization, and verify that synchronization is maintained. The slave should have frame counters, which indicate where the frame position(s), overhead bytes, payload, etc., are within a frame. As the master requests various types of data the payload should check its counters, and adjust them if necessary.

## Receive Direction

The master is the source and the slave is the sink for `rxdat`. All transfers are controlled by `rxval`. If `rxval` is asserted, `rxdat` contains valid frame data.



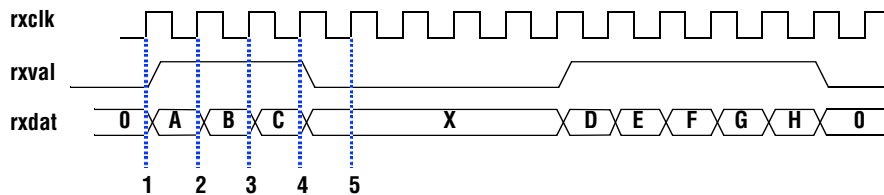
This frame data may be any of payload, overhead, out-of-band signalling, framing patterns, etc. If the data is payload, `rxena` is also asserted.

The `rxclk` may be at a higher frequency than the data rate. If `rxval` is negated, the slave ignores the data on `rxbyte`, and the data value is undefined.

Figure 2 shows the receive bus protocol. A transaction proceeds as follows:

1. Master asserts `rxval` (one or more bits asserted), plus `rxena`, `rxffp`, `rxefp`, `rxfoh`, and `rxeh` as appropriate, and drives data onto the `rxdat` bus into lanes as indicated by `rxval`.
2. On the following clock rising edge the slave observes that `rxval` is asserted and captures `rxdat` plus `rxena`, `rxffp`, etc.
3. The slave observes that `rxval` is asserted and captures `rxdat`.
4. The slave observes that `rxval` is asserted and captures `rxdat`. The master stops data flow by deasserting `rxval` and driving unspecified data on `rxdat`, meeting setup and hold requirements.
5. The slave observes that `rxval` is deasserted and ignores `rxdat` and the other signals.

Figure 2. Receive Bus Protocol



The master negates `rxval` if `rxdat` is not valid, i.e. if the master has no new data because of, for example:

- Loss Of Frame (LOF) synchronization
- Loss Of Signal (LOS)
- Other major malfunction
- Midbus capacity exceeds requirements (See “Low Data Rate” on page 5)

## Transmit Direction

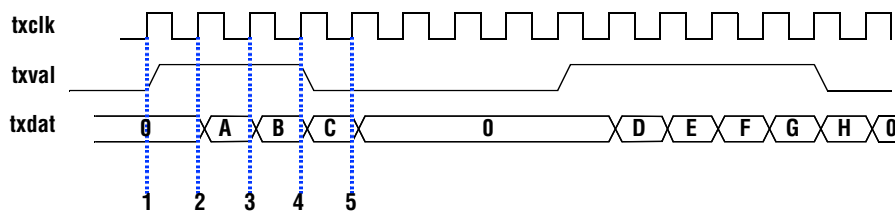
All transfers are controlled by `txval`. If `txval` is asserted, the slave must drive one or more bytes of frame data one `txclk` cycle later. Note that this frame data may be any of payload, overhead, out-of-band signalling, framing patterns, etc. If the data required is payload, `txena` is asserted simultaneously.

The `txclk` may be at a higher frequency than the data rate. In this case, the `txval` signal is asserted for a fraction of the clock cycles. The `txval` is asserted for one clock cycle for each byte of the TDM frame. If `txval` is negated, the slave drives zeros on `txval` one clock cycle later. This allows the master to multiplex transmit streams by ORing their `txdat` lines.

The master is the sink, and the slave is the source of the `txdat`. [Figure 3](#) shows the transmit bus protocol.

1. The master asserts `txval`, plus `txena`, `txffp`, `txefp`, `txfoh`, and `txeoh` as appropriate.
2. The slave observes that `txval` and other signals are asserted and drives data onto the `txdat` bus into lanes as directed by `txval`.
3. The master captures `txdat`. The slave starts driving the next data onto the `txdat` bus.
4. The master captures `txdat` and deasserts `txval`.
5. The master captures `txdat`. The slave observes that `txval` is negated and drives zeros on `txdat`, meeting setup and hold requirements.

**Figure 3. Transmit Bus Protocol**

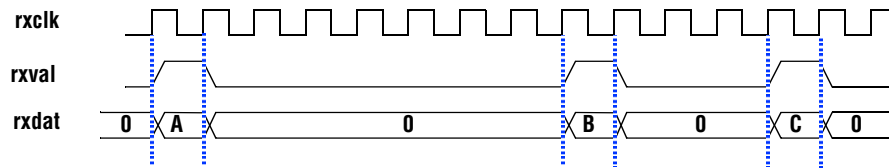


There is a one clock cycle latency from when the master asserts `txval`, and when the master samples `txdat`. However, the bus is pipelined, so data can be transferred on every clock cycle.

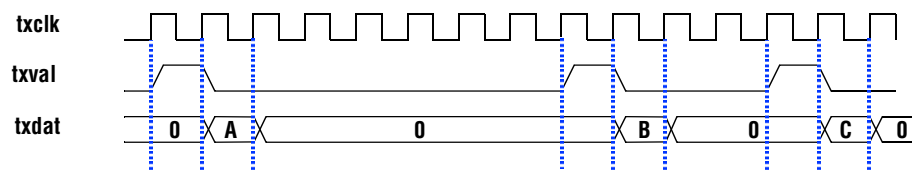
## Low Data Rate

The Midbus is suitable for rate-adapting between a low data rate slave and a high data rate master. To do this, the master asserts the byte enables and valid signals with the appropriate duty cycle. Figures 4 and 5 illustrate the low data rate receive and transmit.

**Figure 4. Low Data Rate Receive**



**Figure 5. Low Data Rate Transmit**



The aggregate data rate will not exceed the facility data rate. Otherwise, no guarantee is made on the duty cycle of val. For example, when carrying an STS-1 on a 78 MHz Midbus, val will be asserted on average one clock cycle in twelve. However, no guarantee is made on the distribution of active cycles (i.e. val asserted). For example, ena may be asserted irregularly or for multiple contiguous cycles.

## Usage of Framing Signals

### Bus Signals

- txdat, txena, txffp, txefp, txfoh, and txeh are synchronous to txclk
- rxdat, rxena, rxffp, rxefp, rxfoh, and rxeh are synchronous to rxclk
- rxclk and txclk are unrelated
- All signals are active high
- If the Midbus is carrying data for a bit serial interface (e.g. T1 or T3), the high-order bits are transmitted and received first (i.e. left-to-right)

Table 1 shows the Midbus signals for one-, two-, and four-lane data paths.

Signal Name	Optional	Description	Width (32-bit bus)	Width (16-bit bus)	Width (8-bit bus)	Direction
txclk	No	Data clock	1	1	1	Input to master and slave
txena	No	Payload byte request(s) for txdat	[3:0]	[1:0]	1	Master to slave
txval	No	Frame (payload+overhead) byte enables for txdat	[3:0]	[1:0]	1	Master to slave
txdat	No	Data from slave(s)	[31:0]	[15:0]	[7:0]	Slave to master
txffp	Yes	Transmit fixed frame position	[3:0]	[1:0]	1	Master to slave
txefp	Yes	Transmit embedded frame position	[3:0]	[1:0]	1	Master to slave
txfoh	Yes	Transmit fixed overhead	[3:0]	[1:0]	1	Master to slave
txeoh	Yes	Transmit embedded overhead	[3:0]	[1:0]	1	Master to slave
rxclk	No	Data clock	1	1	1	Input to master and slave
rxena	No	Payload byte enable(s) for rxdat	[3:0]	[1:0]	1	Master to slave
rxval	No	Frame (payload+overhead) byte enables for rxdat	[3:0]	[1:0]	1	Master to slave
rxdat	No	Data to slave(s)	[31:0]	[15:0]	[7:0]	Master to slave
rxffp	Yes	Receive fixed frame position	[3:0]	[1:0]	1	Master to slave
rxefp	Yes	Receive embedded frame position	[3:0]	[1:0]	1	Master to slave
rxfoh	Yes	Receive fixed overhead	[3:0]	[1:0]	1	Master to slave
rxehoh	Yes	Receive embedded overhead	[3:0]	[1:0]	1	Master to slave

## Multiple Byte Lanes

In two- and four-lane Midbuses, the txdat and rxdat are divided into two and four byte lanes, respectively. In a four-lane Midbus, byte lane 3 contains the first byte transmitted or received. This means data is transmitted from the first bit to the last bit in decreasing order of byte lane number, and decreasing order of bit number.

The byte enables and control signals, txena, rxena, txval, rxval, txffp, rxffp, txefp, rxefp, txfoh, rxfoh, txehoh, and rxehoh correspond to the byte lanes. Therefore, txena [N] / rxena [N] controls byte lane N. For example: rxena [2] controls rxdat [23 : 16] (byte lane 2) of a 32-bit bus.

This allows non-contiguous data on the bus, i.e. if the rxena [3 : 0] pattern is 'b1011 this indicates the first byte is on rxdat [31 : 24], the second byte on rxdat [15 : 8], and the last byte on rxdat [7 : 0].

Table 2 shows the byte enable to byte lane correspondence.

<b>Table 2. Byte Enable to Byte Lane Correspondence</b>				
rxena, txena Bit	Byte Lane	rxdat, txdat Bits		
		32-bit bus	16-bit bus	8-bit bus
3	3	[31:24]	N/A	N/A
2	2	[23:16]	N/A	N/A
1	1	[15:8]	[15:8]	N/A
0	0	[7:0]	[7:0]	[7:0]

Table 3 shows examples of data on the rxdat bus and the rxena signals for a four-lane bus. One- and two-lane buses behave similarly, but with the high number byte lanes and enables not used.

<b>Table 3. Byte Enable Patterns</b>							
rxena3	rxena2	rxena1	rxena0	lane 3 rxdat [31:24]	lane 2 rxdat [23:16]	lane 1 rxdat [15:8]	lane 0 rxdat [7:0]
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	Byte n
0	0	1	0	0	0	Byte n	0
1	0	0	0	Byte n	0	0	0
1	1	1	1	Byte n	Byte n+1	Byte n+2	Byte n+3
1	0	0	1	Byte n	0	0	Byte n+1
1	0	1	0	Byte n	0	Byte n+1	0
1	0	1	1	Byte n	0	Byte n+1	Byte n+2



The Midbus does not support fractional byte data transfer, non byte aligned data, multiple masters, flow control by slave, or dynamic bus sizing slave

## Signal Usage

The `val` signal is asserted for each byte of the frame (including payload and non-payload bytes such as framing and overhead). `ena` is asserted for each payload byte. The usage of the `ffp`, `efp`, `foh`, and `eoh` signals depends on the application. In SONET, the bytes are used as follows:

**Table 4. SONET STS-1 Signal Usage**

SONET byte	<code>val</code>	<code>ena</code>	<code>ffp</code>	<code>efp</code>	<code>foh</code>	<code>eoh</code>
A1, A2, other TOH bytes	1	0	0	0	1	0
J0	1	0	1	0	1	0
J1	1	0	0	1	0	1
B3, other POH bytes	1	0	0	0	0	1
Fixed stuff	1	0	0	0	0	0
Payload	1	1	0	0	0	0

The `ffp` associated with J0 indicates that the following byte is the first payload byte of the STS-1 frame. The `efp` associated with J1 indicates that the following byte is the first payload byte of the SPE. Note that J0 was formerly known as C1.

Figures 6 and 7 illustrate the multi-slave receive and transmit general arrangement.

**Figure 6. Receive Multi-slave**

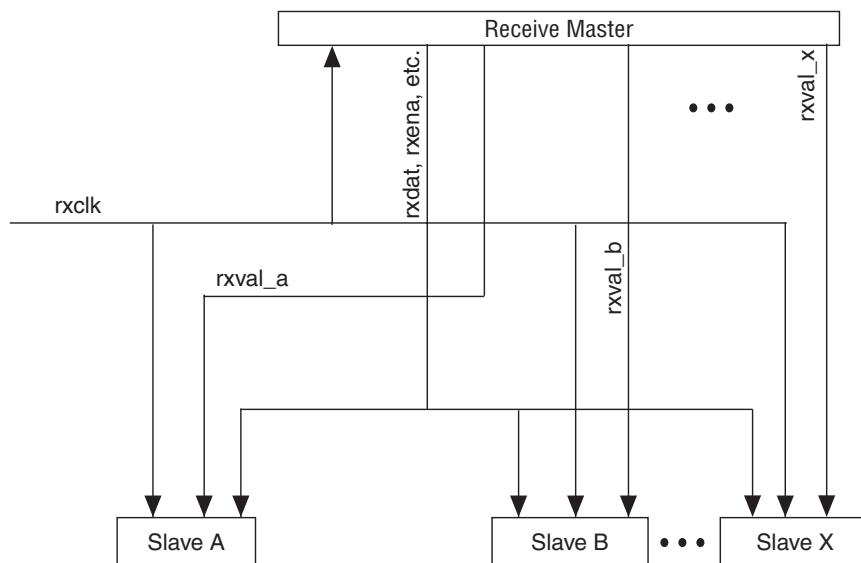


Figure 7. Transmit Multi-slave

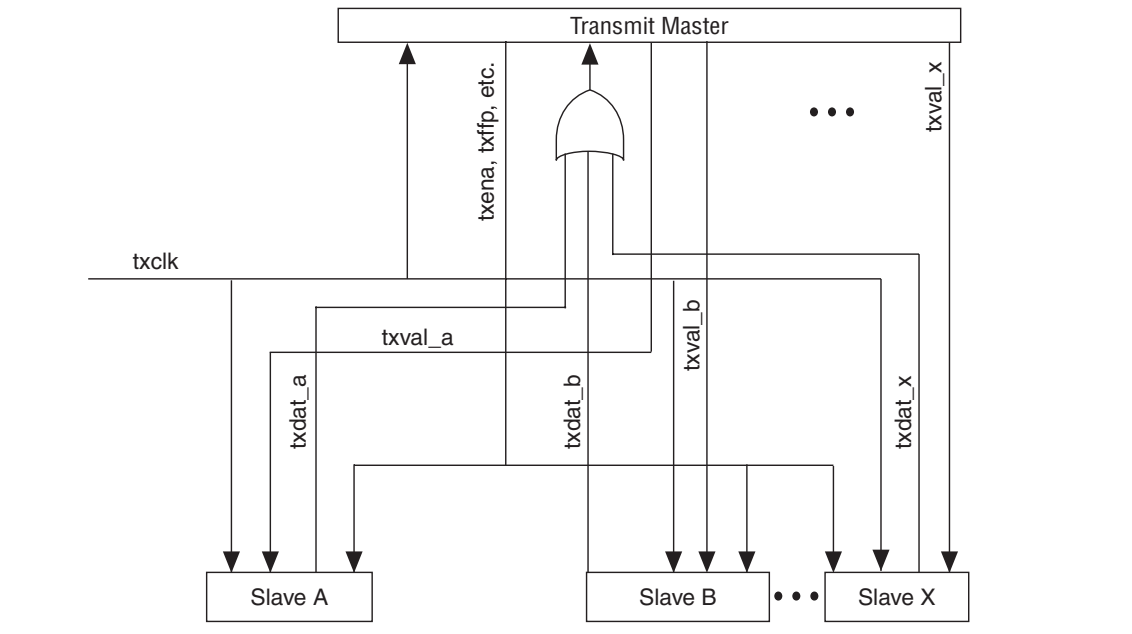
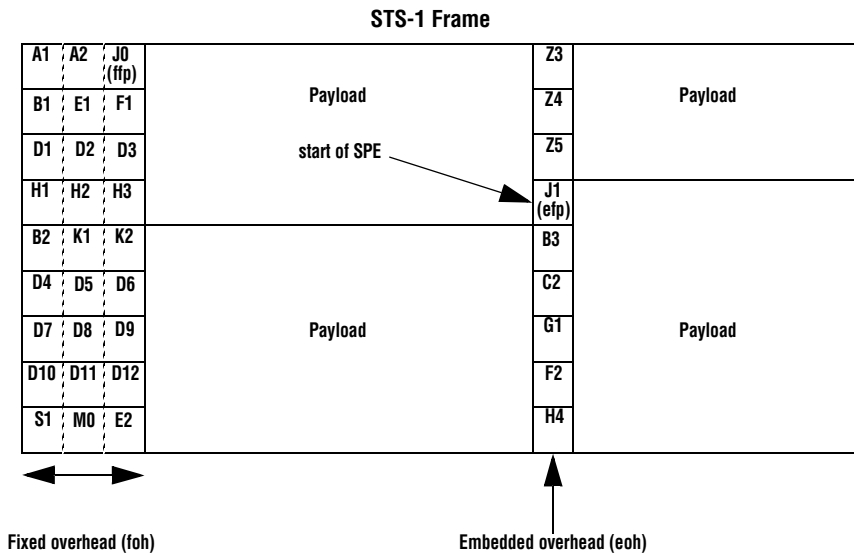


Figure 8 illustrates the STS-1 Frame. The val signals may be used to multiplex data from multiple source slaves or issue data to multiple slave sinks. The master issues separate val signals to each slave. In the case of multiple sources, their outputs are ORed together.

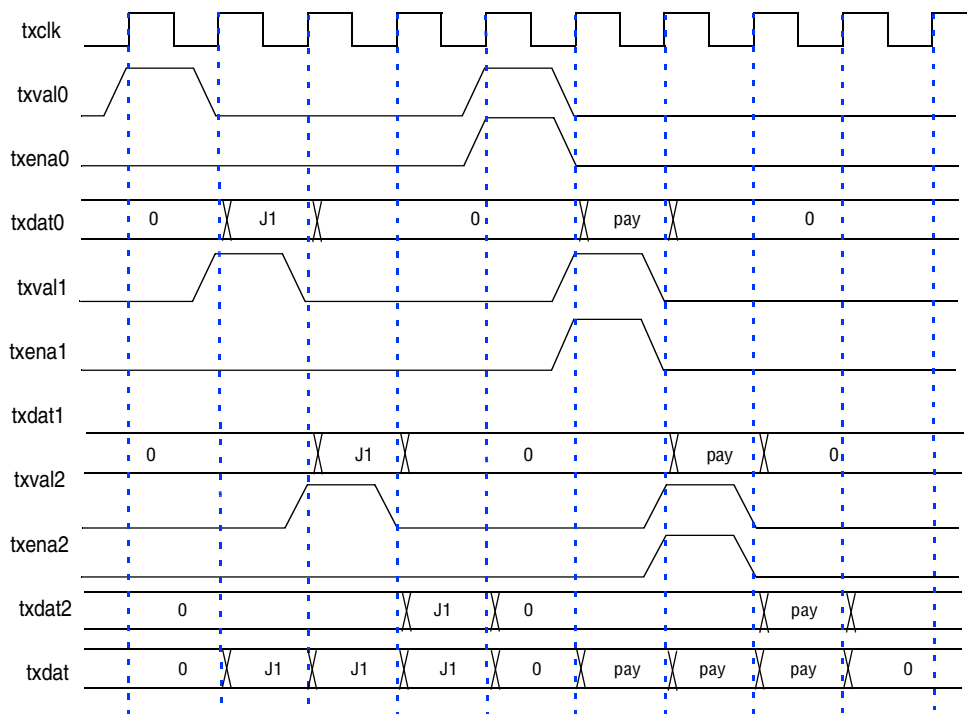
Figure 8. SONET STS-1 Frame



## Midbus Interface Functional Specification 8

Figure 9 shows the multiplexing of three slaves (0, 1, and 2). txdat is the OR of the slaves' txdat0, txdat1, and txdat2. In this example, one timeslot is completely unused. There is a one cycle delay from the val signal to dat valid.

Figure 9. Transmit Multiplexing



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
<http://www.altera.com>  
Applications Hotline:  
(800) 800-EPLD  
Customer Marketing:  
(408) 544-7104  
Literature Services:  
[lit\\_req@altera.com](mailto:lit_req@altera.com)

Altera, APEX, APEX 20K, APEX 20KE, MegaCore, MegaWizard, OpenCore, Quartus, and Quartus II are trademarks and/or service marks of Altera Corporation in the United States and other countries. Altera acknowledges the trademarks of other organizations for their respective products or services mentioned in this document. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Copyright © 2001 Altera Corporation. All rights reserved.

