AD2437 A^{2}B Transceiver Technical Reference

Revision 0.2, July 2023

Part Number
82-100151-01

,

Notices

Copyright Information

Disclaimer

Trademark and Service Mark Notice

Contents

1 Preface

Thank you for purchasing and developing systems using an A^{2}B® Transceiver from Analog Devices.

Purpose of This Manual

The *AD2437 A^{2}B Transceiver Technical Reference* provides information about the transceivers, including register and bit descriptions. For timing, electrical, and package specifications, see the *AD2437* A^2*B Transceiver Data Sheet*.

Intended Audience

This manual is intended for system designers and programmers who want to develop systems using the A^{2}B transceiver.

What's New in This Manual

This revision (0.2) is the second preliminary version of the document.

Register Documentation Conventions

The register sections and diagrams use the following conventions:

- Registers are presented in address order.
- The reset value appears in binary in the individual bits and in hexadecimal to the left of the register.
- Shaded bits are reserved.

NOTE: To ensure upward compatibility with future implementations, write back the value that is read for reserved bits in a register, unless otherwise specified.

Register description tables use the following conventions:

- Each bit's or bit field's access type appears beneath the bit number in the table in the form (read-access/write-access). The access types include:
  - R= read, RC= read clear, RS= read set, R0= read zero, R1= read one, Rx= read undefined

  - W= write, NW= no write, W1C= write one to clear, W1S= write one to set, W0C= write zero to clear, W0S= write zero to set, WS= write to set, WC = write to clear, W1A= write one action, XCVRA/B= transceiver (A-port /B-port)
- Many bit and bit field descriptions include enumerations, identifying bit values and related functionality. Unless otherwise indicated (with a prefix), these enumerations are decimal values.

2 A^{2}B Overview

The A^{2}B bus is a high bandwidth, bidirectional, digital audio bus. It is capable of transporting I^{2}S/TDM/PDM/SPI data and I^{2}C/SPI control information, along with clock and power signals using a single pair cable, CAT cables, or XLR cables. With a configurable 44.1 kHz or 48 kHz frame rate and up to a 50 Mbps bandwidth, the A^{2}B bus is ideal for transporting digital audio. It delivers superior audio quality relative to analog connections, with a deterministic and low latency.

The A^{2}B bus is based on an I^{2}S/TDM (audio multichannel) connection, which is normally only used for connections between components on the same circuit board. The A^{2}B bus makes this multi-channel connection using a cable that is several meters long. Only two A^{2}B wires are required for the TDM signals (BCLK, SYNC, SIO0-SIO4) through which clock and data are transferred in both directions.

An A^{2}B system connects multiple nodes, where each node can consume data, provide data, or both. The transceivers support a direct interface to general-purpose digital signal processors (DSPs), field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), microphones, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and codecs through a multichannel I^{2}S/TDM interface. PDM input is also natively supported, with the transceiver decimating the input to PCM format before placing it on the A^{2}B bus. The transceiver can provide power to remote nodes using the cables with different power scheme. This feature eliminates the need for a local power supply on each bus-powered subordinate node and thereby reduces the overall system BOM costs.

The A^{2}B system does not require a software stack to processes incoming and outgoing data for the main or subordinate transceiver. Implementing the bus requires a host controller to configure the main and subordinate nodes via I^{2}C/SPI. The subordinate nodes are configured using the bus and require no further intelligence (unless the fault occurred during run time).

The transceivers have the following features:

- Line topology
    - Single main, multiple subordinate nodes connected using a daisy chain
        - Synchronous, phase-aligned clock in all nodes

  - CAT cable, XLR/DMX cable or Single pair wires link between nodes (cable length is specified in the product data sheet)
- Bus communication over superframes at the SYNC rate (44.1 kHz or 48 KHz) of the main transceiver
  - 49.152 Mbps bus bit rate/bandwidth at the 48 KHz SYNC rate
  - Two-way half duplex operation in terms of downstream (from the main transceiver towards the last-in-line subordinate transceiver) and upstream (from the last-in-line subordinate transceiver to the main transceiver)
  - Up to 32 upstream and 32 downstream A^{2}B bus slots (includes audio data slots and SPI data tunnel slots)
- Communication over distance
  - 8-bit to 32-bit multichannel I^{2}S/TDM interface at a programmable data rate
    - Full duplex I^{2}S/TDM port supporting TDM2/TDM4/TDM8/TDM16/TDM32 modes with up to five transmit and receive lines
    - Subordinate transceivers supporting a normal data rate (1×SYNC), an increased data rate of (2×SYNC or 4×SYNC) 96 KHz and 192 KHz, and a reduced data rate (such as 24 kHz, 12 kHz, 6 kHz, 4 kHz, 3 kHz up to 375 Hz) that is configured for each subordinate transceiver
    - Flexible mapping of TX/RX TDM channel data to an A^{2}B bus slot using the I^{2}S/TDM crossbar switch
    - Low latency subordinate node-to-subordinate node communication
  - PDM inputs supporting up to four high dynamic range microphones
  - High speed SPI (up to 10 Mbps) over distance
  - I^{2}C to I^{2}C control and status information
    - Support for I^{2}C fast mode plus (1 MHz)
  - GPIO over distance
    - Up to eight GPIO with configurable pin mapping
- Line fault diagnostic block to detect and localize short-to-VBUS and short-to-GND, short together, open wire or reverse wire faults
- Configurable as A^{2}B bus main or subordinate node
  - Configurable with SigmaStudio® graphical development tool
- Bus power or local power subordinate nodes
  - Up to 50 W A^{2}B bus power
- Support for Low Voltage Input (LVI) mode for 3.3V VIN

- Increased voltage regulator capacity (up to 100 mA each)
- ADC monitoring of supply voltages
- Dedicated hardware reset pin
- Unique ID register for each transceiver
- Support for crossover or straight-through cabling
- Support for LED control using four PWM outputs
- Parallel bus operation to collaborate between two A^{2}B chains
- Two mailboxes in each subordinate node to customize a handshake between the host and a local processor on a subordinate node transceiver.

A^{2}B Terminology

NOTE: Analog Devices is in the process of updating documentation to provide terminology and language that is culturally appropriate. This is a process with a wide scope and will be phased in as quickly as possible. Thank you for your patience.

To make the best use of the A^{2}B system, it is helpful to understand the following terms.

A-Side or A-Port

A^{2}B transceiver interface that faces toward the main (toward the immediately upstream node).

B-Side or B-Port

A^{2}B transceiver interface that faces toward the last-in-line subordinate (toward the immediately downstream next-in-line subordinate)

Bus Link

The A^{2}B bus can consist of multiple daisy-chained subordinate nodes connected to a single main node. The physical connection between a main and subordinate node 0, as well as all physical A^{2}B connections between subordinates, are called bus links.

Bus Monitor

Bus monitor mode enables the transceiver to act as a passive audio bus monitor, also referred to as a *sniffer*.

Bus Powered Subordinate node (BPS)

Subordinate transceivers can tap into the bias voltage on the A^2B bus link and use it as the only power supply. These A^2B subordinate transceivers are considered to be *bus-powered*.

Data Channel

A data channel carries the synchronous I^2S/TDM data for a single sensor/actuator (for example, an ADC, a microphone, or a speaker). The I^2S/TDM interface uses equally sized data channels, where the width of the data word is often smaller than the width of the I^2S/TDM data channel.

Data Slot

A synchronous data word of a single sensor/actuator (for example, an ADC, a microphone, or a speaker), as mapped onto the A^2B bus.

Discovery

The process of discovering all A^2B transceivers one-by-one starting from the main node transceiver. Once power is properly established, each transceiver in the system must be discovered and configured by the host.

Downstream

Communication flow from the main node transceiver toward the subordinate nodes, terminating at the last-in-line subordinate transceiver.

GPIO

General Purpose Input Output

Host

The processor that programs the main transceiver. The host is also the source for the synchronous clock on the A^2B bus. The clock signal (BCLK) is part of the I^2S/TDM interface between the host and main.

I^2C

Inter IC communication bus. When referencing the I^2C protocol, the terms *master* and *slave* have changed to *controller* and *target*.

I ^{2} S/TDM

The inter IC sound (I^{2}S) bus carries pulse code modulated (PCM) information between audio chips on a PCB. The I^{2}S/TDM interface extends the I^{2}S stereo (2-channel) content to multiple channels using time-division multiplexing (TDM).

Line Fault Diagnostics

A mechanism to detect and locate any line fault present in the system

Local Powered Subordinate node (LPS)

Subordinate node transceivers use local power that is sourced by extra wires.

Local Processor

The processor that programs the subordinate transceiver. It is connected to the subordinate node.

LVDS

Low voltage differential signaling

LVI

Low voltage input that supports 3.3 VIN

Mailbox

Virtual mailboxes available on A^{2}B transceivers that allow for interprocessor communication between the host and a control processor of the subordinate transceiver

Main Node

Originator of the clock (derived from the I^{2}S input), downstream data, network control, and power. The main node is comprised of the host processor and an A^{2}B main transceiver, which receives payloads from the host and sends payloads to the host. The term *master node* has changed to *main node* in future revisions of this document. Additionally terms like master transceiver have been changed as well.

PDM

Pulse Density Modulation (PDM) is used in sigma delta converters. PDM format represents an over-sampled 1-bit sigma delta ADC signal before decimation. It is often used as the output format in digital microphones.

PRBS

Pseudo random binary sequence

Preamble

Synchronization bits that signal the start of a control or response frame. The downstream control frame preamble is sent by the main transceiver for every superframe. Subordinate transceivers synchronize to the downstream control preamble and generate a local, phase-aligned main clock from it.

PWM

Pulse Width Modulation

Remote Peripheral

Receiver (processor or peripheral) of the I^{2}C transaction from the locally connected A^{2}B transceiver.

Response Time

Specifies the time a last-in-line transceiver waits after the start of a superframe before it responds with the Synchronization Response Frame (SRF). Response time is programmed in the main transceiver and all next-in-line subordinates so that these transceivers know when to expect the direction to switch from downstream to upstream.

Subordinate Node

Addressable network connection point. Subordinate node transceivers can be the source and/or destination of both downstream and upstream data slots. Every A^{2}B subordinate node has an A^{2}B subordinate transceiver.

The term *slave node* has been changed to *subordinate node* (abbreviated as sub node) in this document. Additionally terms like slave transceiver have been changed as well.

Standby Mode

A low power mode in which only a minimal (19-bit) SCF exists to keep all the subordinate node transceivers synchronized

Synchronization Control and Response Frames (SCF/SRF)

Control frame for nodes (control header) and response frame from nodes (response header). Headers include a preamble for synchronization and enable read and write access to all nodes.

Synchronous Data

Data streamed continuously (for example, audio signals) with a fixed time interval (selectable between 44.1 kHz or 48 kHz) between two successive transmissions to and from the same node.

Superframe

The overall frame structure for A^2B communication. It starts with an SCF, includes optional data slots, and concludes with an SRF. Superframes repeat every 1024 bus clock cycles.

Upstream

Communication flow from the last-in-line subordinate node to the main node

VMTR (Voltage Monitor) ADC

On-chip multichannel successive approximation ADC

A^2B Block Details

The ***A^2B Block Diagram*** show the various blocks used for discovery, bus operation, configuration, and debug on the A^2B transceiver.

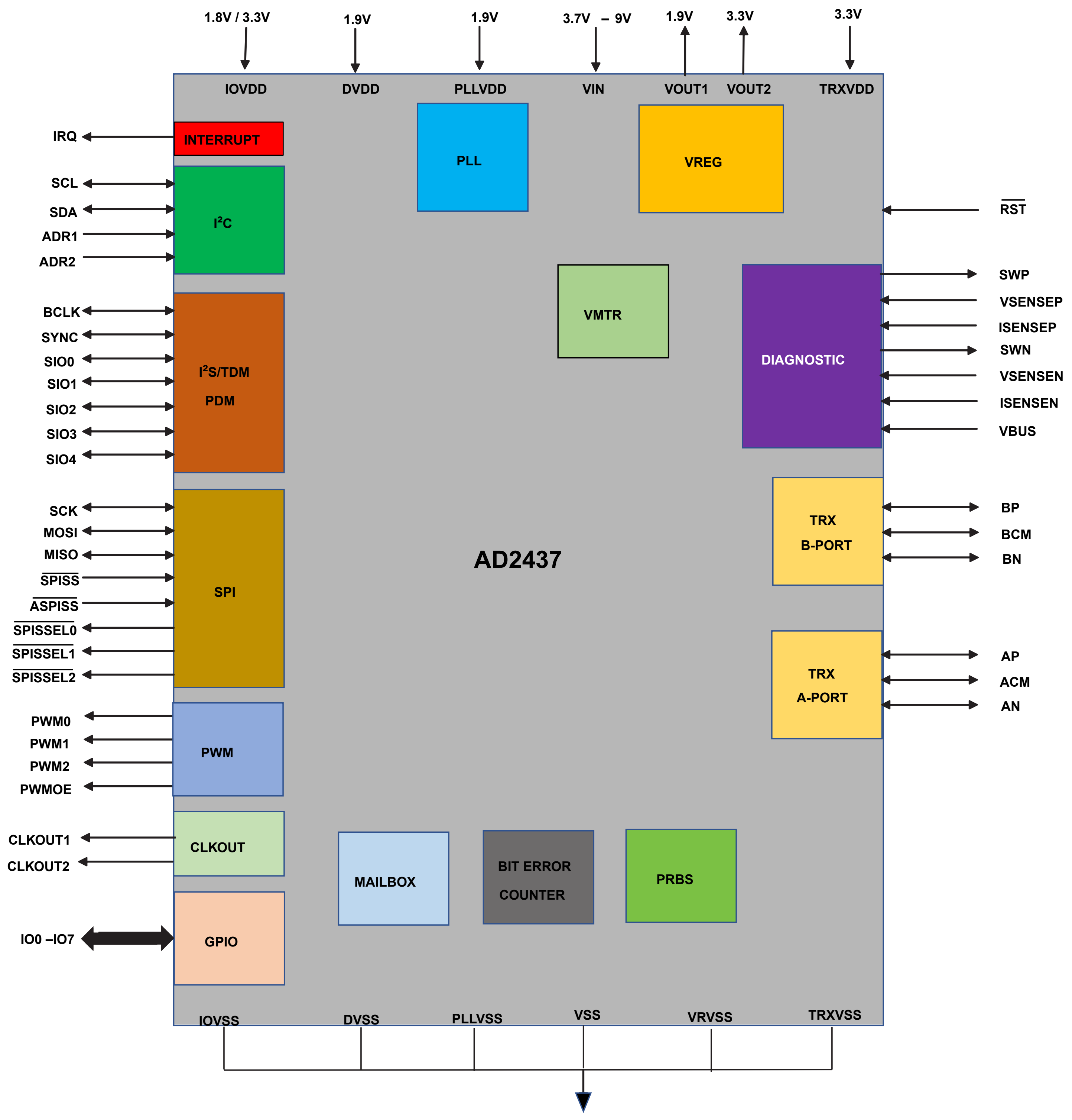


Figure 2-1: A^{2}B Block Diagram

NOTE: All of the pins associated with blocks are shown in the *A^{2}B Block Diagram*, but, some pins may be multiplexed with each other. For example, the ADR1 signal is multiplexed with CLKOUT1, SPI slave select input ($\overline{\text{SPISS}}$), SPI slave select output ($\overline{\text{SPISSEL0}}$), PWM output enable (PWMOE), and GPIO signals. The pins can be configured for one of the functionalities depending on system requirements. For details on pin multiplexing, refer to "Pin Configuration and Function Descriptions" table in the data sheet and General Purpose Input/Output (GPIO) Pins sections.

I^{2}C Interface

The I^{2}C interface is used to access and program the registers of the A^{2}B transceiver. The host processor can configure and control the whole A^{2}B chain using the I^{2}C interface of the main node. The I^{2}C port of the A^{2}B main node is always an I^{2}C target, accepting commands from the host processor. The I^{2}C port of the A^{2}B subordinate node can be an I^{2}C controller when accessing a remote peripheral; or, it can be an I^{2}C target when a local processor needs to access the registers of the local transceiver. The supported I^{2}C bit rates are: 100 kbps, 400 kbps or 1 Mbps. The ***I^{2}C Interface*** diagram shows the I^{2}C interfaces in a typical A^{2}B system.

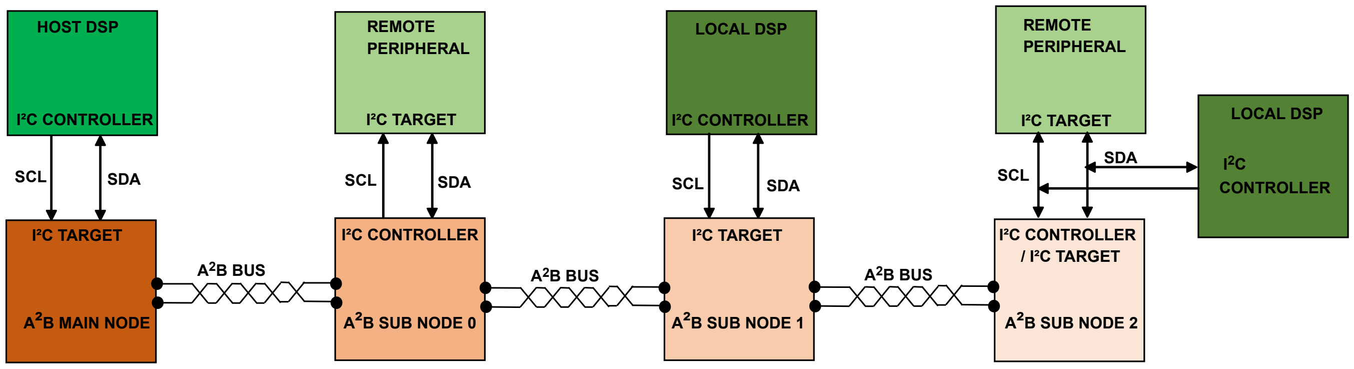


Figure 2-2: A^{2}B System - I^{2}C interface

In this system, the following I^{2}C accesses are possible:

- The host processor accesses the registers of the A^{2}B main transceiver.
- The host processor accesses the registers of any A^{2}B subordinate transceiver over the A^{2}B bus.
- The host processor accesses the registers of remote peripherals connected to the subordinate transceiver. The host initiates I^{2}C access over the A^{2}B bus; the subordinate node transceiver becomes the I^{2}C controller and replicates the access on its I^{2}C port.
- The local processor accesses the registers of the local subordinate node. This access is a local I^{2}C access between the local processor and a connected subordinate node. The subordinate node A^{2}B transceiver supports a multi-controller I^{2}C environment.

The host processor can fully configure and maintain the operation of the A^{2}B system including the main node, subordinate nodes, and remote peripherals connected to the subordinate node. The A^{2}B transceiver supports single word and burst type accesses.

The ADR1 and ADR2 pins determine the I^{2}C address of the transceiver at power-up. For more information, refer to the I^{2}C Interface section.

I^{2}S/TDM/PDM Interface

The I^{2}S/TDM/PDM interface is used to communicate data (typically audio) with the local I^{2}S/TDM/PDM device. The interface uses the A^{2}B bus to exchanges data over distance with other transceiver nodes. The I^{2}S/TDM serial port operates in full-duplex mode.

The TDM interface typically consists of the bit clock (BCLK), a frame synchronization signal (SYNC), data transmit lines (DTX), and data receive lines (DRX). At the main transceiver, the TDM interface is a target; it accepts the BCLK and SYNC signals from the audio host or SoC. At the subordinate transceiver, the TDM interface is the controller; it provides BCLK and SYNC signals to connected I^{2}S/TDM devices.

NOTE: To support parallel A^{2}B buses, one of the A^{2}B subordinate transceivers can be programmed as the TDM target.

Several TDM modes are supported including: TDM2, TDM4, TDM8, TDM16, and, TDM32. The transceiver supports channel sizes of 16 bits and 32 bits. The frame sync type can be pulse mode/50% duty cycle, active high/active low, early sync/left justified. The driving edges of DTX lines can be independently configured with a DRX sampling edge configuration.

The PDM interface consists of the bit clock to the PDM microphones and PDM data output stream from the microphones. The transceiver decimates the input PDM stream and converts it to PCM format before placing it on the A^{2}B bus. Therefore, no extra processing is needed to process the PDM stream. The transceiver supports two PDM input pins. Up to four PDM microphones can be interfaced to the transceiver with two PDM microphones connected to each PDM pin (one driving on the rising edge and the other driving on the falling edge).

The I^{2}S interfaces in a typical A^{2}B system are shown in the ***I^{2}S/TDM/PDM Interface*** figure. It is possible to have both a TDM and a PDM interface operating in a transceiver.

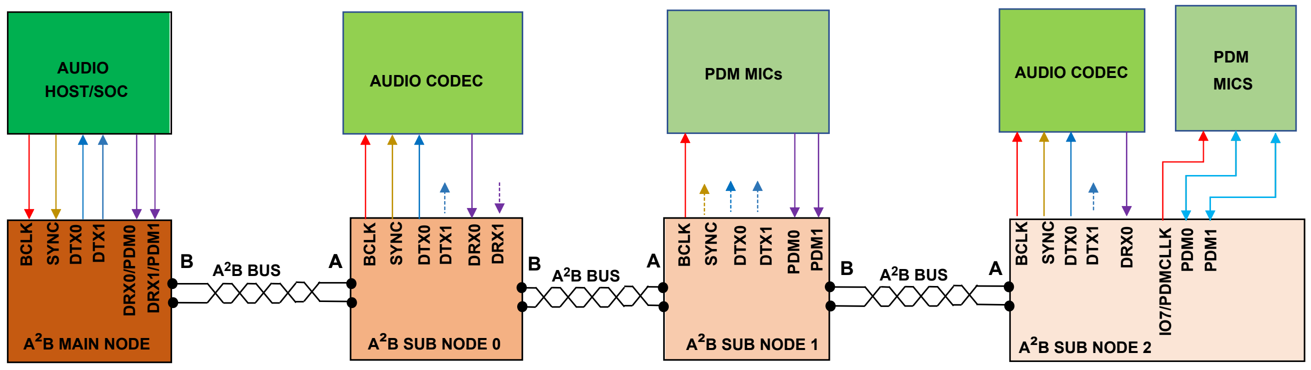


Figure 2-3: I^{2}S/TDM/PDM Interface

Once configured, the transceivers can communicate with each other without intervention from the main transceiver or host processor. Subordinate node-to-subordinate node communication is possible without passing the data to the main transceiver.

The I^{2}S/TDM/PDM port includes five programmable data pins (SIO0-SIO4) which can be configured to any combination of up to two PDM streams and up to five I^{2}S/TDM streams, with a maximum of four I^{2}S/TDM streams in the same direction (DTX/DRX). For more information, refer to I^{2}S/TDM Port Programming Concepts.

SPI Interface

The transceiver supports an SPI interface that is used for control or data transfer. When used for control, it can access local node registers, subordinate node registers and remote peripherals, similar to I^{2}C operations. It can be used for asynchronous data exchange similar to synchronous data exchange over the I^{2}S/TDM port. The SPI port consists of a serial clock (SCK), Master Out Slave In (MOSI), and Master In Slave Out (MISO) signals. When in SPI slave mode, the slave select lines ($\overline{\text{SPISS}}$ or $\overline{\text{ASPISS}}$) are used to select the device. When in SPI master mode, the transceiver has three SPI slave select lines ($\overline{\text{SPISSEL0}}$/1/2).

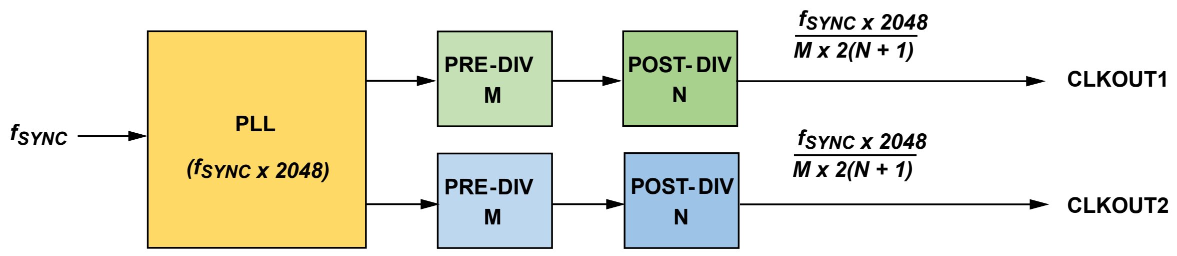
CAUTION: In SPI slave mode, ADR1 is the primary slave select line. It is possible to use ADR2 or SIO2 as an alternate slave select line, but this is available only for one type of SPI datatunnel transaction, the register based full-duplex transaction.

Pulse Width Modulation (PWM)

The transceiver supports a PWM block for LED control for ambient lighting. It uses three PWM outputs and one main node dimmer output (OE). The PWM outputs sends the clock at a fixed frequency (192 kHz, 96 kHz, 48 kHz, 24 kHz, 12 kHz, 3 kHz, 1.5 kHz, 750 Hz, 375 Hz, or 187.5 Hz) and modulates the duty cycle (high/low time) for brighter or dimmer lights. The PWM block can operate using a frequency hopping scheme in which the PWM frequency hopper randomly selects frequencies from 187.5 to 3 kHz to spread the PWM emissions over a range of frequencies.

Clock Outputs (CLKOUT)

The transceiver can generate two clock outputs as general-purpose clock sources: CLKOUT1 and CLKOUT2. The clock signals are generated from an internal PLL after applying a predivisor (Pre-DIV) of either 2 or 32 and then a postdivisor (Post-DIV) with a value between 0 and 15.



f_{SYNC}
PLL
(f_{SYNC} x 2048)
PRE-DIV M
POST-DIV N
PRE-DIV M
POST-DIV N
$\frac{f_{SYNC} \times 2048}{M \times 2(N+1)}$
CLKOUT1
$\frac{f_{SYNC} \times 2048}{M \times 2(N+1)}$
CLKOUT2

Figure 2-4: CLKOUT PLL

The clock signal can be used to provide MCLK to connected audio codecs or an audio DSP on a subordinate node, so that whole audio chain can be synchronous.

NOTE: CLKOUT signals are synchronous with other signals like the bit clock (BCLK) or PDMCLK, but are not phase-aligned with these signals.

General Purpose IOs (GPIOs)

The transceiver supports up to eight GPIO pins, which can be configured as either GP inputs or GP outputs for general purpose operations such as mute control, handshake, and LED control. When in GP input mode, the transceiver can generate an interrupt on a pin toggle. Therefore, it is not required to poll the status. The host processor can control these IOs using the GPIO registers of the transceiver.

The transceiver also support the GPIO over distance feature in which the GP input of one transceiver can be mapped to the GP output of another transceiver. Therefore, GPIO communication between multiple transceivers can happen without host intervention after initial programming. The GP output toggles automatically according to the signal on the GP input. See General Purpose Input/Output (GPIO) Pins.

NOTE: Most of the GPIO signals are multiplexed with peripheral pins such as I^{2}S/TDM, SPI, and I^{2}C. Verify the availability of the GPIO pins needed for a specific application.

Interrupt (IRQ)

The transceiver features a dedicated interrupt pin (IRQ) to signal various interrupts to the host processor. The signal indicates:

- Bit errors on the bus
- A line fault on the bus
- Status interrupts like main transceiver PLL locked and subordinate transceiver discovery done
- SPI status/error, I^{2}C error
- An interrupt from subordinate node GP input pins
- Mailbox interrupts

- VMTR interrupts

The interrupts reported by any transceiver are raised on the IRQ pin of main transceiver. Therefore, the host can manage the whole A^{2}B system without requiring intelligence or a stack on the subordinate transceivers. When the IRQ pin is asserted, the host controller checks the type and source of the interrupt and takes the required action. The subordinate transceiver can locally generate interrupts only for mailbox and SPI operations.

Bit Error Counter

Each transceiver supports a bit error counter that can count bit errors up to a configured threshold before interrupting the host. This feature avoids individual reporting of bit errors and reduces host attention. The threshold can be set based on acceptable noise and robustness over a period.

PRBS

Pseudo random binary (PRBS) sequence is used for error checking between nodes. When PRBS node-to-node check is enabled, each transceiver checks the incoming data bits and transmits the expected data to the next-in-line transceiver. This feature permits better determination of where bus errors occur.

Mailbox

Each A^{2}B subordinate transceiver contains two mailboxes (MBOX0 and MBOX1) that allow interprocessor communication between the host processor at the main node and the local processor at the A^{2}B subordinate node. The direction of mailbox can be configured to transmit or receive.

If a local processor at a A^{2}B subordinate node must send a message to the host processor at the A^{2}B main node, the mailbox can be configured in transmit mode. In this case, the local processor writes the message (up to 4 bytes) to the mailbox data registers of the local A^{2}B subordinate transceiver. The A^{2}B main transceiver informs host processor about the new message by an interrupt on the IRQ pin. The host can read out the message by accessing the mailbox data registers of the A^{2}B subordinate transceiver.

If the host processor must send a message to the local processor of the A^{2}B subordinate node, the mailbox can be configured in receive mode. In this case, the host processor writes the message (up to 4 bytes) to the mailbox data registers of the A^{2}B subordinate transceiver. The A^{2}B subordinate transceiver informs the local processor about the new message by an interrupt on IRQ pin. The local processor can read the message by accessing the mailbox data registers of the connected A^{2}B subordinate transceiver.

PLL

The PLL block is an important block of the transceiver. At the main transceiver, the SYNC signal of the TDM interface (typically 44.1 KHz or 48 KHz) is used as the input to the PLL; at the subordinate transceiver, the Synchronization Control frames (SCF) coming over the A^{2}B bus are used as input to the PLL.

The main transceiver generates the SCFs and transmits them over the A^{2}B bus to all of the subordinate transceivers. The SCFs are at same rate as the SYNC signal received by the main transceiver. Therefore, all transceivers are synchronous with the frame rate (from the audio host processor to the main transceiver). This synchronization eliminates need for a local oscillator for PLL operation.

The *PLL Input Path* figure shows the SYNC signal in a typical system.

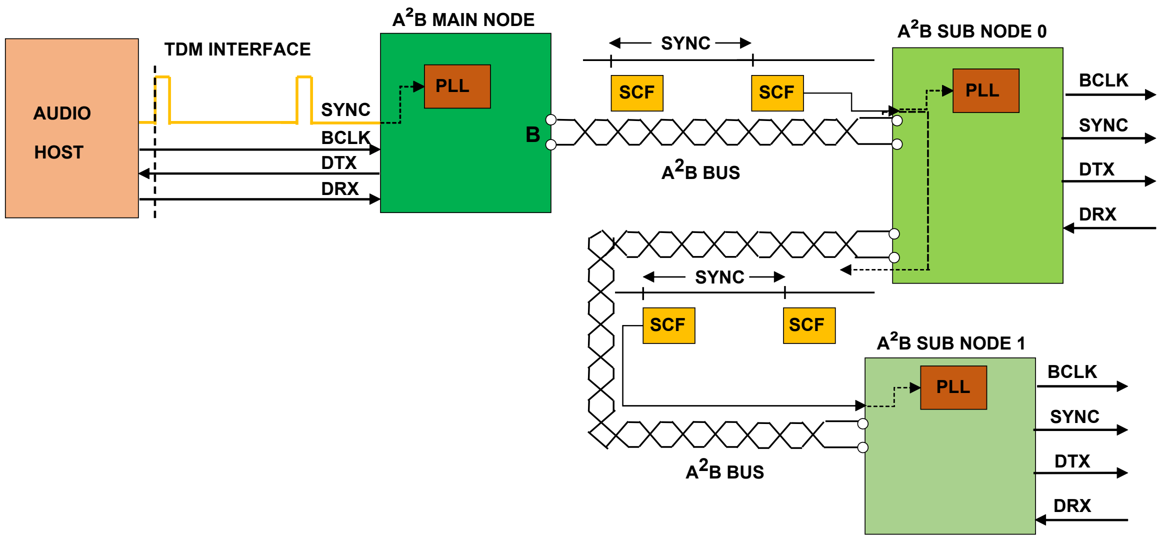


Figure 2-5: PLL Input Path

The PLL multiplies the input signal frequency (f_{SYNC} or SCF) by 2048 for internal operation. For example, if the SYNC rate is 48 kHz, the PLL output is 98.304 MHz; if the SYNC rate is 44.1 KHz, the PLL output is 90.317 MHz.

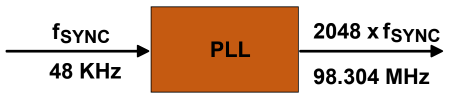


Figure 2-6: PLL Formula

Power Supplies

The A²B transceiver has following power supplies:

- VIN – the main power supply of the transceiver. It feeds the internal voltage regulators and the line diagnostics block.
- PLLVDD – typically, a 1.9 V supply that provides power to the PLL of the transceiver
- DVDD– typically, a 1.9 V supply that provides power to the digital blocks of the transceiver
- TRXVDD– typically, a 3.3 V supply that provides power to the A-port and B-port LVDS transceivers
- IOVDD – the supply powers the digital IOs of the A²B transceiver. This supply can be 1.8 V or 3.3 V depending on IO levels of the interfaced device (audio host/codec/microphones).

Voltage Regulators

The transceiver can generate two voltage supplies from the VIN voltage: VOUT1 (typically, 1.9 V) and VOUT2 (typically, 3.3 V). The on-chip regulators must be used to supply different power domains such as PLLVDD, DVDD, and TRXVDD. However, IOVDD can be powered through either the on-chip regulators or by an external regulator. VOUT1 and VOUT2 can also power external peripherals like PDM microphones and audio codecs. Each regulator (VOUT1/2) can supply a current up to 100 mA maximum, with a combined current (VOUT1 + VOUT2) of up 130 mA. Refer to the product data sheet for more details.

In normal mode, the power supplies of the A^{2}B transceiver must be connected to VOUT1 and VOUT2 as shown in the ***Voltage Regulator - Normal Mode*** figure. IOVDD can be connected to either VOUT1 or VOUT2 depending on the 1.8 V or 3.3 V IO requirement.

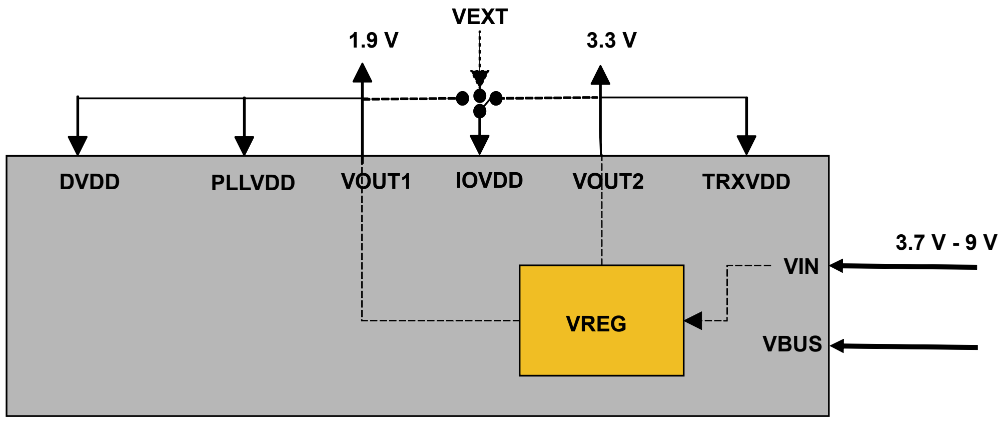


Figure 2-7: Voltage Regulator - Normal Mode

The transceiver supports a Low Voltage Input (LVI) mode in which VIN can be 3.3 V. In this mode, only VOUT1 is available; VOUT2 must be connected to the VIN pin. There is no restriction on VBUS; any voltage within the respective specified range can be used. The `A2B_SWSTAT2.LVI_MODE` bit indicates that the transceiver is in LVI mode. The ***Voltage Regulator - LVI Mode*** figure show the power supplies used in LVI mode.

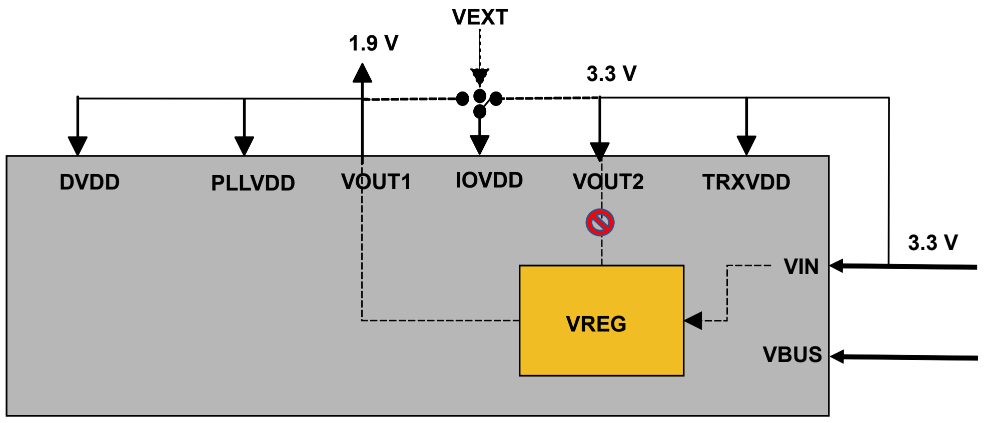


Figure 2-8: Voltage Regulator - LVI Mode

NOTE: The decoupling capacitors on different power domains are not shown in the figure.

Voltage Monitor ADC (VMTR)

The voltage monitor ADC allows the host processor to monitor the health of key voltages on A^{2}B bus nodes. It can selectively monitor VIN, VBUS, DVDD, TRXVDD, and IOVDD supply voltages and high/low side downstream currents. The VMTR can generate interrupts based on configured maximum and minimum thresholds of selected supplies.

Reset

The transceiver remains in a reset state until all of the supplies (VIN, IOVDD, DVDD, TRXVDD) are stable.

The transceiver features a dedicated active-low hardware reset pin to reset the device. The reset pin can be deasserted after all the power domains are stable, thus eliminating the need for power-up sequencing.

Once the transceiver powers up, the transceiver transitions through the main transceiver state machine or subordinate transceiver state machine depending on host configurations. Refer to the Operating States for details.

LVDS Transceiver Ports

The A^{2}B transceiver consists of two bidirectional, differential A^{2}B line driver and receiver XCVR (transceiver) ports – known as A-port and B-port. The A-port transceiver interface is toward the A^{2}B main transceiver; the B-port transceiver interface is toward the next-in-line A^{2}B subordinate transceiver.

The B-port of a transceiver is connected to the A-port of the next-in-line transceiver using the A^{2}B cable. The A^{2}B cable connection starts from the B-port of the A^{2}B main node to the A-port of the last-in-line A^{2}B subordinate transceiver through the middle A^{2}B subordinate nodes. The *A^{2}B System Ports* figure shows the main and subordinate nodes connected in daisy chain.

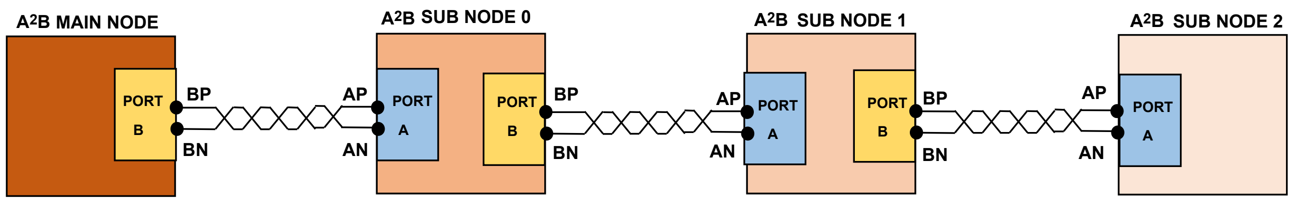


Figure 2-9: A^{2}B System Ports

NOTE: The Bus Interface Network (BIN) on the A-port and B-port are not shown in the figure.

For the main node, only the B-port is used; it is connected to the A-port of the first-in-line subordinate transceiver. For the last-in-line subordinate transceiver, only the A-port is used; it is connected to the B-port of the next-in-line (upstream) transceiver.

Line Fault Diagnostics

The line diagnostic block of the transceiver can detect and localize cable line faults that occur on the A^2B bus. Different types of line faults are detected in different power schemes which is explained in the Line Fault Diagnostics chapter.

These line faults are detected during and after discovery in the system run time. When a fault is detected during discovery, the switches that enable the bias current to the next-in-line node are disconnected automatically. The main transceiver indicates the fault condition to the host main using the interrupt (IRQ) pin. Refer the A^2B System Debug section for details.

Power Configurations

The AD2437 transceiver supports three power configurations:

- XLR/DMX cable based. This configuration uses a XLR or DMX cable with an external NMOS on the high side.
- CAT cable based. This configuration uses a CAT cable and RJ45 connector with an external NMOS on the high side.
- Single Wire Pair. This configuration has an external NMOS on both the high and low side.

Transceiver Identification

Every A^2B transceiver has a vendor ID register (`A2B_VENDOR`), a product ID register (`A2B_PRODUCT`), and a version ID (`A2B_VERSION`) register to indicate to a host which A^2B transceivers are present in a system.

Every A^2B transceiver vendor is assigned a unique vendor ID (Analog Devices A^2B transceivers use 0xAD as the vendor ID). The `A2B_PRODUCT` and `A2B_VERSION` registers are assigned by the chip vendor to uniquely identify the chips. The transceiver models use the 0x37 (AD2437) product ID.

Every A^2B transceiver contains a 48-bit unique ID. Read the `A2B_CHIPID0` through `A2B_CHIPID5` registers to obtain the unique ID.

A^2B Bus Details

The A^2B bus provides a multichannel I^2S/TDM, I^2C, and SPI link over distances between nodes. It embeds clock, data, synchronization signals, and power onto the differential cable using different power scheme. The data contains bidirectional synchronous digital audio data from the I^2S/TDM interface and asynchronous data from an I^2C and/or SPI interface.

The A^2B communication system is a single-main, multiple-subordinate node system where the transceiver at the host controller is the main transceiver. The main transceiver generates clock, synchronization, and framing signals for all subordinate nodes. The host generates a periodic synchronization signal on the I^2S/TDM interface at a fixed frequency (typically, 48 kHz) to which all A^2B transceivers synchronize. Communication on the A^2B bus occurs in periodic superframes. The superframe frequency is the same as the synchronization signal frequency; data is transferred at a bit rate that is 1024 times faster (typically, 49.152 MHz). Each superframe is divided into periods of downstream transmission, upstream transmission, and no transmission (where the bus is not driven).

Any node transceiver can place data from the I^{2}S/TDM, SPI, or I^{2}C interface on the A^{2}B bus; any node can consume data from the A^{2}B bus and put it on their I^{2}S/TDM, SPI, or I^{2}C interface.

The ***A^{2}B Superframe*** figure shows the basic frame structure, known as a superframe on the A^{2}B bus, through which all nodes communicate with each other.

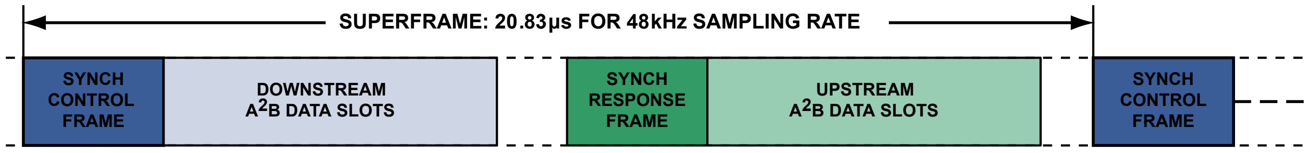


Figure 2-10: A^{2}B Superframe

The main node starts the superframe by putting the Synchronization Control Frame (SCF) on the A^{2}B bus followed by the downstream data slots that it wants to send it to subordinate nodes. All subordinate nodes receive the SCF field and the downstream slots from previous node connected to the A-port and pass those to the next-in-line node connected to the B-port, after consuming and/or contributing some slots as per configuration. During this downstream part of the superframe, the bus direction is from the main node towards last-in-line subordinate node.

The next half of superframe is upstream. The last-in-line subordinate node initiates the transaction by putting the Synchronization Response Frame (SRF) on the A^{2}B bus. It follows with the upstream data slots intended for the main node and middle subordinate nodes. All of the middle subordinate nodes receive the SRF field and the upstream slots from the next-in-line node connected to the B-port. It passes the slots to the upstream node connected to the A-port after consuming and/or contributing slots. The upstream transaction ends with the main node receiving the SRF and the upstream slots. During this upstream part of the superframe, the bus direction is from last-in-line subordinate node towards the main node.

The A^{2}B transceiver supports a direct point-to-point connection and allows multiple, daisy-chained nodes at different locations to contribute and/or consume time division multiplexed channel content.

NOTE: All superframe bits on the A^{2}B bus are differential and in Manchester encoded format. The differential swing is specified in the product data sheet.

Clock on the Bus

The entire A^{2}B chain runs synchronously with the SYNC signal fed by the host controller to the main node. The main transceiver locks the PLL from this signal and provides the synchronization frames over the A^{2}B bus to all the subordinate transceivers at this rate. The SCF at the start of the superframe includes a preamble field that provides the synchronization signal to all of the subordinate nodes. The subordinate node transceivers lock their PLL based on the preamble field. The subordinate transceivers can provide the main clock (MCLK) to local peripherals like audio codecs and microphones. Therefore, the whole audio chain can be synchronous without the need of a local oscillator or ASRC.

Data on the Bus

A superframe consists of an initial period of downstream transmission and a later period of upstream transmission. The A^{2}B bus can contain I^{2}C data, I^{2}S/TDM data, SPI data, GPIO information, interrupt information, and an internal handshake/data exchange between nodes.

The I^{2}C data is exchanged between nodes using the SCF and SRF. When the host processor must access a subordinate node register or a remote peripheral, the main transceiver sends access details like the register address and register values over the SCF field to the respective subordinate transceiver. In the case of a read access, the subordinate transceiver passes the register value over the SRF field to the main transceiver. The I^{2}C access does not occupy the downstream or upstream synchronous data fields. When the SPI interface is used for control purposes, the transceiver uses the SCF and SRF bandwidth similar to an I^{2}C access.

The I^{2}S/TDM data is exchanged between nodes over downstream and upstream synchronous data fields. The downstream starts after the SCF. The exchange starts from the main transceiver towards the last-in-line subordinate transceiver. It consists of I^{2}S/TDM channels that the particular transceiver must send to the transceivers that are connected towards its B-port. All nodes can contribute their channels to the downstream part of superframe. Any subordinate transceivers can consume the downstream slots received on its A-port and pass the remaining slots on the B-port to the downstream nodes.

After the SRF, the upstream data starts from the last-in-line subordinate transceiver towards the main transceiver. It consists of I^{2}S/TDM channels that the particular transceiver must send to the transceivers that are connected towards its A-port. All subordinate transceivers can contribute their channels to the upstream part of the superframe. The subordinate transceivers can also consume the upstream slots received on its B-port and pass the remaining slots on the A-port to the upstream transceivers. The main transceiver receives the upstream data slots at the end.

When an SPI interface is used as a data tunnel, it uses downstream and upstream bandwidth similar to the I^{2}S/TDM interface.

The ***A^{2}B Data Flow*** figure shows the flow of data on the bus.

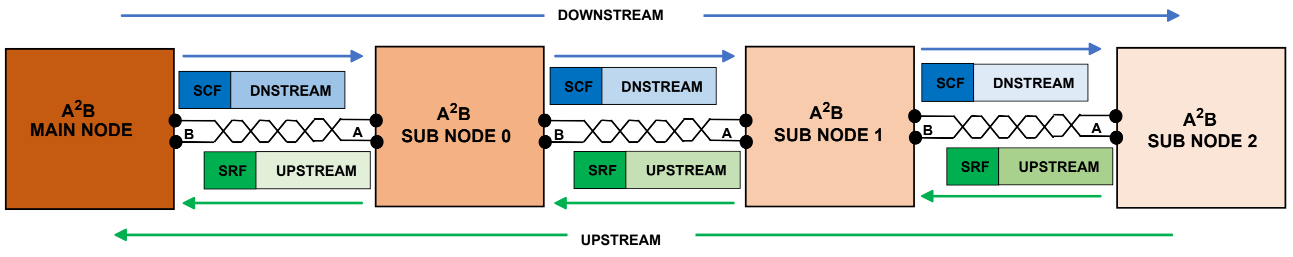


Figure 2-11: A^{2}B Data Flow

All nodes in an A^{2}B system are sampled synchronously in the same A^{2}B superframe. Downstream data from the transceivers arrives at all downstream subordinate nodes in the same A^{2}B superframe; upstream data from every node arrives synchronously in the same I^{2}S/TDM frame at any upstream node. The communication over distance does not require involvement of the main transceiver; it can be performed directly between subordinate transceivers.

The nodes also communicate interrupt and IO information for GPIO over distance. These details are exchanged over the SCF and SRF fields.

Power on the Bus

The A^2B transceiver can deliver power to the subordinate nodes using different power schemes. In case of the single pair wire, the bus bias comes from a regulator via the external MOSFET and is added to the differential data signals. When using the RJ45 power scheme, the bus bias is added to 3 pairs of the CAT cables and data is added to remaining pair. Similarly in case of the XLR/DMX power scheme, power is added to 2 lines of the cable and the 3rd line is used as the return path. During downstream transactions, the B-port of the transceiver outputs the SCF and downstream slots in a differential format.

In the single wire pair, the high-pass filter on the B-port blocks the DC bus bias and the low-pass filter on the bus bias path blocks the differential data signals. Therefore, a differential signal rides on the DC bus bias and travels to the next-in-line transceiver. At the next-in-line transceiver (on its A-port), the differential data signal is passed through the high-pass filter and received. The DC bus bias is blocked by the high-pass filter. The differential data signal is blocked by the low-pass filter. But, the DC bus bias is passed through and supplies power to the transceiver and the external load (such as a codec/MIC or other regulators). In this way, the subordinate transceiver can be powered using bus bias, thereby, avoiding power cables running to the subordinate node board.

Similarly, during upstream transactions, the A-port of the downstream node outputs the SRF and upstream slots in differential format. The low-pass filter on the bus bias path blocks the differential data signals. The differential signal rides on the DC bus bias and travels to the upstream node. At the upstream node transceiver on its B-port, the differential data signal is passed through the high-pass filter and received. The differential data signal is blocked by the low-pass filter on the bus bias path.

When RJ45 CAT cables are used, the 24V DC bus bias is sent to the next-in-line transceiver using three pairs of the CAT cables and the differential data is transferred using the remaining pair. A 5V DC is added on the remaining pair using a low pass filter and blocks the differential data signals and a high pass filter is used to block the DC from getting into the data pins of the IC. The differential data along with the 5V DC is sent to port A of the transceiver on the fourth pair of the CAT cable. Here the differential data signal is passed through the high-pass filter and received, and the 5V DC bus bias is blocked by the high-pass filter. This 5V is passed using the low pass filter to power the IC and the EEPROM on the node. The 24V DC bus bias is sent to the sub node after the next-in-line transceiver node is verified as an A^2B node.

When XLR/DMX cables are used, the 24V DC bus bias is sent to the next-in-line transceiver using the two wires on the cable and the third wire is used as the ground return path. The high-pass filter on the B-port blocks the DC bus bias from going into the data pins and the low-pass filter on the bus bias path blocks the differential data signals. Therefore, a differential signal rides on the DC bus bias and travels to the next-in-line transceiver. At the next-in-line transceiver (on its A-port), the differential data signal is passed through the high-pass filter and received. The third wire on the XLR/DMX cable is connected to the board ground on both the ports.

3 Node Initialization

The A^{2}B bus system consists of a single main node and multiple subordinate nodes connected using a daisy chain. The host processor that is connected to the main node brings up all the A^{2}B nodes in normal discovery mode.

Operating States

The A^{2}B transceiver has four states:

- RESET
- POWER-UP
- PLL LOCKED
- SUSTAIN

The operation of main and subordinate nodes is similar in these states (based on register configurations). However, the way the transceiver enters and exits the states can vary.

The ***Transceiver State Diagram*** figure shows transceiver state information when bringing up and running a complete A^{2}B system.

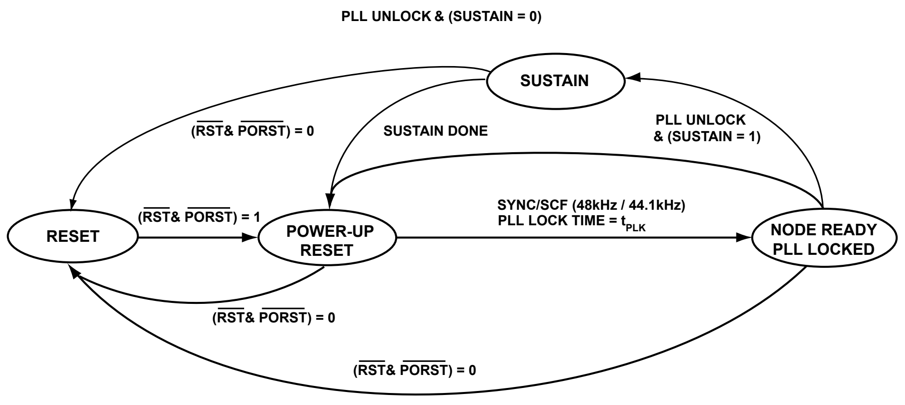


Figure 3-1: Transceiver State Diagram

RESET State

Initially, all nodes are in the RESET state. The RESET state is a function of the power applied to the transceiver and the state of the hardware reset pin ($\overline{\text{RST}}$). The internal power-on reset circuit monitors the state of the VIN, DVDD, IOVDD, and TRXVDD power supply pins. It asserts an internal power-on reset signal ($\overline{\text{PORST}}$) until the chip reset deassertion voltage threshold (V_{RSTN}) is met for each of the power supply pins. The $\overline{\text{RST}}$ pin also controls the reset state of part. Once all of the power supplies are stable and the $\overline{\text{RST}}$ pin is deasserted, the part transitions to the POWER-UP state. When in the RESET state, all A^{2}B system blocks and registers are held in reset; no registers can be programmed until the transceiver advances to the POWER-UP state.

When the transceiver is in another state (for example, POWER-UP, PLL LOCKED, or SUSTAIN), it goes into the RESET state when:

- the $\overline{\text{RST}}$ pin is asserted low, or
- any of the power supply VIN, DVDD, IOVDD, or TRXVDD drop below the V_{RST} voltage threshold

POWER-UP State

The transceiver transitions from the RESET state to the POWER-UP state when:

- the transceiver power supplies are stable, and
- the hardware $\overline{\text{RST}}$ pin is in the deasserted state

In this state, the transceiver is powered on, but its PLL is not locked.

If a valid MCLK signal is not detected after power-on, the transceiver samples the ADR1 and ADR2 pins to determine the BASE_ADDR for locally accessing the node over the I^{2}C interface. By default, all transceivers are subordinate transceivers waiting for discovery frames (SCFs) to be received on the A-port. The host processor brings up the local node as the main node by setting the `A2B_CONTROL.MSTR` bit and then and discovers the remote nodes as subordinate nodes.

NOTE: Once the transceiver comes out of the RESET state and enters the POWER-UP state , the I^{2}C and SPI ports are accessible after t_{ACCESS} and then the node registers can be programmed.

During the POWER-UP state, the main node tries to lock the PLL on the incoming SYNC signal; the subordinate node tries to lock the PLL on the SCFs signals received over the A^{2}B bus on its A-port. Once the node PLL is locked, the transceiver transitions to the PLL LOCKED state. The node can then be configured and made operational.

The transceiver returns to the POWER-UP state from the PLL LOCKED state through an optional SUSTAIN state when:

- a main node PLL unlocks due to a SYNC signal issue (break), or
- a subordinate node PLL unlocks due to SCF signals corruption

NOTE: When the node returns to the POWER-UP state from the operational PLL LOCKED state, all registers return to their reset values except for `A2B_CONTROL`, and `A2B_BMMCFG`. During the POWER-UP state, it is possible to program the registers of the local node before its PLL is locked. Some registers take effect immediately in the POWER-UP state, while other registers take effect after the node PLL locks. If registers are programmed in the POWER-UP state, a failed PLL lock attempt can return the registers to the reset value. This can happen when the node PLL locks for a brief period, and then unlocks due to a SYNC/SCF issue. Therefore, it is recommended to configure the node registers in the PLL LOCKED state. See Subordinate Node Bring-Up for details.

NOTE: In the POWER-UP state, the node is accessed with BASE_ADDR only. Since there are no subordinate nodes discovered on port-B yet, the accesses with BUS_ADDR should not be tried. In POWER-UP state, A^2B transceivers have limited amount of logic active, including remote I^2C management. Therefore the node acknowledges the accesses with BUS_ADDR without any NAK or I2CERR, and the bus read accesses returns 0x00 value. For detailed information see Transceiver I^2C Device Address Selection.

PLL LOCKED State

The transceiver transitions to the PLL LOCKED state when the PLL locks successfully during the POWER-UP state. This is a *ready* state in which the nodes can be configured to exchange data with other nodes on the A^2B bus.

Once the main node successfully locks its PLL to the incoming SYNC signal, it is ready to discover the connected subordinate nodes. The subordinate nodes lock their PLL using the SCFs sent by the main node during the discovery process. The subordinate nodes maintains this lock during the operational state. During the operational state, if the node PLL unlocks, the transceiver returns to the POWER-UP state and most of the node configuration is lost. The node transceiver must be brought up again.

SUSTAIN State

If there is a bus communication loss during the operational state and the node does not receive SCFs over the bus (needed to maintain the PLL lock), the subordinate node PLL unlocks and the node returns to the POWER-UP state. The transceiver has an optional SUSTAIN state, in which audio signals are gracefully muted before the transceiver returns to the POWER-UP state.

Upon entering the clock SUSTAIN state, the transceiver:

- Runs at the current clock frequency for 1024 SYNC periods (for example, ~21.33 ms for a 48 KHz SYNC)
  - I^2S/TDM ports continue running and data is attenuated slowly
  - GP output signals continue to hold the same state. The transceiver can signal the SUSTAIN state on a GPIO pin.
  - PLL relock is not attempted

After completion of the SUSTAIN state, the transceiver:

- Enters the POWER-UP state, thereby resetting all registers. The node can transition to the PLL LOCKED state if stable SCF discovery frames are present.

During the SUSTAIN state, a GPIO pin of transceiver can be optionally driven high to indicate the start and end of the SUSTAIN phase. The sustain mode output can be enabled using the `A2B_SUSCFG.SUSOE` bit; it is driven on a GPIO pin indicated in the `A2B_SUSCFG.SUSSEL` bit field. The selected pin has a higher priority than the GPIO configuration, but a lower priority than the functional configuration for the pin. For example, if IO7 is configured as a GP output in an operational state and sustain mode output is configured on the same pin, then, during the SUSTAIN state, the IO7 pin becomes the sustain mode output. But, if the IO7 pin is configured as PDMCLK in the operational state and the sustain mode output is configured on same pin, then, during the SUSTAIN state, IO7 continues to function as PDMCLK, and the sustain mode output is not driven.

During the SUSTAIN state, decaying data values are produced on the I^2S/TDM transmit data (DTXn) pins. The last known good sample is gradually attenuated to mute the audio signals gracefully. Negative values attenuate to zero, while positive values attenuate to –109 dB (0x00001F00) on the enabled data pins.

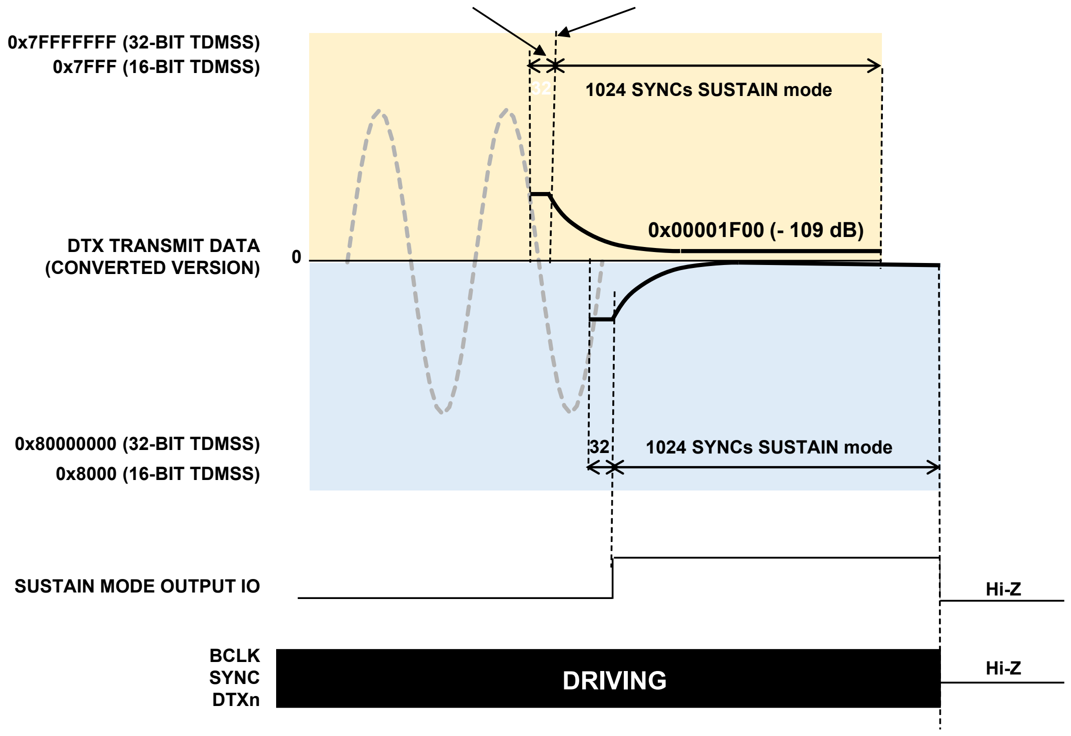


Figure 3-2: Sustain Mode

NOTE: SUSTAIN mode is enabled, by default. It can be disabled by setting the `A2B_SUSCFG.SUSDIS` bit. If rediscovery of the A^2B bus is needed during run time, a running subordinate transceiver can go into SUSTAIN mode. It runs for ~21.3 ms in that state. The host must consider this time when rediscovering the bus.

During run time, if a soft reset is applied to the main node with `A2B_CONTROL.MSTR` = 0, the node is no longer the main node. This condition causes the PLL to unlock. The transceiver, subsequently, enters the SUSTAIN state.

Therefore, it is recommended to apply the soft reset to the node with `A2B_CONTROL.MSTR` = 1. However, if the invert SYNC mode is used (`A2B_I2SGCFG.INV` = 1), follow the recommended sequence given in the Discovery Flow chapter.

The subordinate node enters the SUSTAIN state in the following cases:

1. When the node cannot recover the PLL input from SCFs on the bus. In this case, after 32 missed SCFs, the node PLL unlocks; the transceiver enters into SUSTAIN mode. During the SUSTAIN state, the node becomes the last-in-line node (`A2B_NODE.LAST`). All other registers maintain their value, including the `A2B_SWCTL` register that controls power to the downstream nodes. After the completion of SUSTAIN mode, the node returns to the POWER-UP state, thereby, resetting all registers.
2. When the host processor (over the bus) or a local processor (using the I^2C port) applies a soft reset to the node. In this case, the node registers reset immediately (except the `A2B_CONTROL`, and `A2B_BMMCFG` registers). This reset causes the node switch to open and, thereby, stops power to the downstream nodes. Since the node loses the register configuration, the GPIO pins and DTX pins are three-stated immediately. The node runs for 1024 SYNC in this state and then returns to the POWER-UP state.

Node Bring-Up

The A^2B system consists of a single main node and multiple subordinate nodes connected in a daisy chain. The host processor first brings up the local A^2B node as the main node and then brings up connected subordinate nodes one-by-one. During the discovery process, once a subordinate transceiver locks its PLL, the host processor can communicate with the transceiver using the A^2B bus. The host processor initializes the node to set it into an operational mode. The node starts exchanging data with other nodes. Once the configuration of A^2B bus is complete, host intervention is minimal. Intervention is required only when A^2B nodes generate interrupts. This design avoids the requirement of a heavy stack and reduces host processor MIPS during run time.

The following sections describe the main and subordinate node bring-up processes.

Main Node Bring-Up

The A^2B transceiver that is directly connected to the host processor is the A^2B main node. Every A^2B transceiver powers-up as a subordinate node. The host processor brings up the connected node transceiver as the main node using the following process:

1. When the node enters the POWER-UP state from the RESET state, wait t_{ACCESS} for internal operations of transceiver to complete. The device is I^2C/SPI accessible after this time. The node registers can be programmed using I^2C and/or SPI.
2. The host processor configures the node as a main node by setting the `A2B_CONTROL.MSTR` bit.

NOTE: Applying a soft reset along with setting the `A2B_CONTROL.MSTR` bit is optional during a fresh power-up. Applying a soft reset while initiating a full rediscovery is recommended. Use the rediscovery process after the nodes are configured in operational mode. The host processor can keep the uniform settings for a fresh discovery and rediscovery.

3. The host processor or audio host provides a stable SYNC signal at the audio sampling rate of the system (selectable between 48 kHz or 44.1 kHz). The transceiver starts locking its PLL to this signal.

   NOTE: The SYNC signal can be supplied before setting the `A2B_CONTROL.MSTR` bit, wherein the transceiver starts locking the PLL as soon as `A2B_CONTROL.MSTR` bit is set. Or, the SYNC signal can be supplied after setting the `A2B_CONTROL.MSTR` bit, wherein the transceiver waits for the SYNC signal used for locking the PLL.

4. The transceiver locks its PLL to the incoming SYNC signal within the PLL lock time (t_{PLK}) specification. The PLL lock time depends on the SYNC pin input frequency. It is implemented by a 360-sync cycle digital counter. Therefore, PLL lock time is not affected by ageing or temperature. For a 48 KHz SYNC frequency, the PLL lock time is 360 x (1/48K) = 7.5 ms.

   NOTE: The (internal PLL lock logic) node transceiver:

      a. Starts locking the PLL.
      b. Waits for the PLL lock time (360 sync cycle counter).
      c. Checks whether the PLL locked. If the PLL is not locked, it reattempts to lock (repeat process). If the PLL locks, then proceed.

   If the SYNC signal is not stable or not per the specifications given in the data sheet, the PLL lock time can be greater than t_{PLK} specification due to frequent PLL lock and unlock attempts. The host processor should set a timeout that is a multiple of the t_{PLK} so that a non-responsive transceiver can be detected by the software. The timeout also depends on when the SYNC is available for locking the PLL.

5. Once the PLL is locked, the node transitions to the PLL LOCKED state and generates the main node PLL locked (main node running) interrupt on the IRQ pin (INTTYPE = 0xFF). This interrupt indicates that the main transceiver is ready for further programming and can bring-up the connected subordinate nodes using the discovery process.

   NOTE: Once the node PLL is locked, writing 0 or 1 to the `A2B_CONTROL.MSTR` bit has no effect. It does not unlock the PLL or return the transceiver to subordinate mode.

6. After reset, the IRQ pin is in high-impedance until the PLL locks. The default active high polarity of IRQ pin is based on the `A2B_PINCFG.IRQINV` bit such that when an interrupt becomes active, the IRQ pin toggles to a high state.

   To configure an active low state for the IRQ pin:

   a. Apply a soft reset by setting the `A2B_CONTROL.SOFTRST` bit

b. Change the polarity of the IRQ pin by setting the `A2B_PINCFG.IRQINV` bit

c. Set the `A2B_CONTROL.MSTR` bit

NOTE: A pull-up or pull-down resistor may be needed to keep the IRQ pin in a known inactive state when the IRQ three-state option is enabled (`A2B_PINCFG.IRQTS` = 1).

7. If the main transceiver PLL becomes unlocked during the bus operation due to a SYNC signal break, the transceiver goes into the POWER-UP state. The event is indicated to the host processor when the `A2B_CONTROL.MSTR` =1 value is maintained during the operational state. The `A2B_CONTROL` register value is retained when a node transitions to the POWER-UP state; the transceiver attempts to relock the PLL when the SYNC signal resumes. Once the PLL is relocked, the main node generates the PLL locked interrupt (INTTYPE=0xFF). Therefore, if the host receives the PLL locked interrupt during an operational state, it can determine that the main node went through the POWER-UP state and the bus must be reinitialized.

8. During an operational state, if any of the VIN, DVDD, IOVDD, or TRXVDD power supply drops below V_{RST}, or if the $\overline{\text{RST}}$ pin is asserted low, the transceiver goes into the RESET state without generating an interrupt.

NOTE: The voltage monitor ADC can be configured to detect a drop in supply voltages (below an operational threshold) and generate an interrupt.

Subordinate Node Bring-Up

After POWER-UP, the subordinate node transceivers must be discovered before they can communicate with the host processor and other A^2B nodes. The host processor discovers the subordinate nodes as follows.

1. By default, every node is in subordinate mode, unless the local processor sets the `A2B_CONTROL.MSTR` bit to make it a main node. In subordinate mode, the transceiver waits to receive synchronization control frames (SCFs) on its A-port.

2. Close the external MOSFET switch of the upstream node by setting the `A2B_SWCTL.ENSW` bit (for example, when discovering subordinate node 0, the host processor must close the switch on the main node).

   - Closing the switch provides the bias on the A^2B bus. If a node to be discovered is bus-powered, it gets powered from the bus bias.
   - Upon closing the switch, the upstream node starts sending SCFs to the subordinate node to be discovered. The node starts locking its PLL to the preamble field of the SCF frames. The PLL lock time and lock mechanism in a subordinate node is similar to the main node PLL operation except for the PLL input source.

NOTE: If there is a critical fault present on the subordinate node connected to A^2B bus, the upstream node checks the bus before closing the switch and reports a fault interrupt.

3. Write the response cycle of the next-in-line node (to be discovered) into the `A2B_DISCVRY` register of the main node. The main node starts sending discovery frames with embedded response cycle values through the

SCF field. Once discovery is started, the upstream node waits for the SRF response from the next-in-line node transceiver.

4. When the node to-be-discovered completes its PLL lock operation, it extracts the response time value from discovery frames and starts generating SRFs from the subsequent superframes at the response time. When the main node receives the SRF from a newly discovered subordinate node, it raises the subordinate node discovered interrupt (INTTYPE = 0x18). This interrupt indicates to the host processor that the subordinate node is discovered, ready for configuration, and ready to be put into operation.

   NOTE: 1. During discovery, all upstream nodes adjust their response cycle timing as detailed in the Response Cycles section.

   2. After reset, the IRP pin is in high-impedance until the PLL locks.

5. During discovery, the bus-powered subordinate node takes extra time to power-up itself. In this case, the subordinate node discovery time includes:

   - Bring-up time of the VIN supply – generally, the bus bias is directly connected to the VIN pin of the transceiver. Therefore, the VIN power-up time of the transceiver depends on the capacitive load connected to the VIN pin. In some cases, there could be an extra load affecting the bus bias and, subsequently, the VIN power-up time of the transceiver.
   - Bring-up time of the subordinate node – Once the VIN supply is stable, the internal voltage regulators are powered up to generate the VOUT1/2 supplies. The VOUT1/2 supplies feed power domains like DVDD, PLLVDD, TRXVDD, and IOVDD externally. Once all of the power domains are stable, the internal power-on reset ($\overline{\text{PORST}}$) signal is deasserted. The node performs some internal operation before being fully ready to accept the SCFs from the bus and start locking the PLL.

     NOTE: The $\overline{\text{RST}}$ pin of subordinate transceiver must be pulled high in hardware.

   - Node PLL lock time – The typical PLL lock time is constant as specified by t_{PLK} . It depends on an internal 360 count SCF counter. If there is corruption of the SCFs due to a noisy condition, such that the node cannot recover the clock input for the PLL, it may take extra time to lock the PLL. PLL lock reattempts are in terms of t_{PLK}.

   For a local powered subordinate node, it is possible that the transceiver is already powered up when the host processor starts the discovery process, and the node is waiting for SCFs to be received over bus for locking its PLL. In this case, when the host processor initiates the discovery, the node starts locking its PLL immediately. If the subordinate node board uses an optocoupler on the A-port to power-up regulators on the board, bring-up time is similar to that of a bus-powered subordinate node.

   NOTE: It is recommended that the host processor set a timeout in excess of the subordinate node discovery time so that a non-responsive transceiver can be detected by software. This wait time is in the host software. There is no interrupt/indication from the A^{2}B main node for a software timeout. Typically, t^1 ms is the wait time for a subordinate node PLL lock operation that allows multiple relock attempts. The VIN supply bring-up time can be included if it is considerable. The wait time (t^{1}ms) is not a hard recommendation. The host processor can rely on interrupts (successful discovery or discovery failure due to a fault) to avoid waiting for a long time (in general t^1 = 70 ms).

6. Configure the subordinate node using A^2B bus accesses. Once the subordinate node PLL is locked, it is ready for initialization. Local programming of the node is not needed. This adds simplicity to system; an intelligent main or stack is not needed at the subordinate node.

   When necessary, a local processor can program the subordinate node registers through its I^2C port. The registers can be configured during the POWER-UP state of a node (for example, before being discovered by the host processor), or in the PLL LOCKED state (once the node is discovered). The local processor can check the node discovery status by:

   - reading the `A2B_NODE.DISCVD` bit
   - checking the GPIO pin status toggled by host processor upon discovery
   - using a mailbox handshake

   When registers are programmed in the POWER-UP state, the configuration can:

   - take effect immediately (for example, GPIO)
   - take effect after the node PLL locks
   - take effect after audio traffic starts on the bus (for example, slot configuration)
   - cannot be done locally and must be programmed by the host processor only using the A^2B bus.

   CAUTION: If registers are programmed in the POWER-UP state, then failed PLL lock attempt may return the registers to the RESET state. This can happen if the node PLL unlocked due to an SCF corruption issue after locking for a brief period. To avoid this, it is recommended to configure the node registers once its PLL is locked.

   NOTE: The local processor accesses the registers via the I^2C port. The host processor accesses the registers using the A^2B bus. Therefore, there is no contention in register accesses. However, if both processors write to same register at same time, the order in which this register gets written and overwritten cannot be predicted. It is not recommended to write to the same register from both sides; or if required, a handshake between processors can done before attempting the access.

7. If the discovery process results in failure due to a line fault present on the bus, terminate the discovery process by setting the `A2B_CONTROL.ENDDSC` bit.

8. Initiate rediscovery or partial discovery to bring back any dropped nodes into the operational A^2B chain. When a node cannot derive a valid PLL input for 32 consecutive superframes:

   - The PLL unlocks (for example, due to an excessive noise condition that results in corrupted SCF frames on the bus)
   - The node transceiver returns to the POWER-UP state via the optional SUSTAIN phase.

- The node loses all configuration information and most of the registers return to the RESET state (except `A2B_CONTROL`, and `A2B_BMMCFG`)
- The next-in-line upstream node generates an SRFMISSERR error interrupt (INTTYPE=5) for 32 consecutive superframes; the next-in-line upstream node becomes the last-in-line node (sets the `A2B_NODE.LAST` bit) and stops reporting SRFMISSERR. If the upstream node switch is still closed (not opened by host processor upon detecting the node drop), the dropped node may get SCFs using the bus to relock its PLL. Since the node does not see the discovery frames embedded with its response cycle value, it does not generate the SRF response to upstream nodes. The upstream node does not expect the SRF response from the downstream node when the discovery process is not ongoing.

The ***Summarized Discovery Flow*** figure shows the main components of the flow used to bring up the A^{2}B chain. For a detailed discovery flow, see the Discovery Flow chapter.

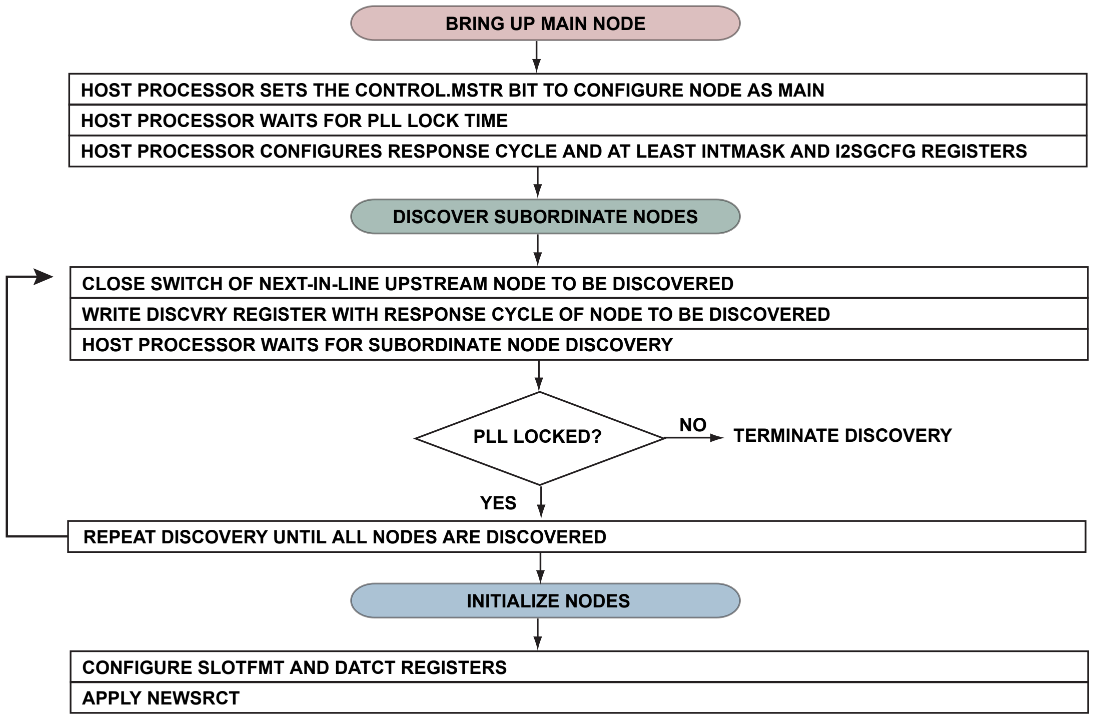

BRING UP MAIN NODE
HOST PROCESSOR SETS THE CONTROL.MSTR BIT TO CONFIGURE NODE AS MAIN
HOST PROCESSOR WAITS FOR PLL LOCK TIME
HOST PROCESSOR CONFIGURES RESPONSE CYCLE AND AT LEAST INTMASK AND I2SGCFG REGISTERS
DISCOVER SUBORDINATE NODES
CLOSE SWITCH OF NEXT-IN-LINE UPSTREAM NODE TO BE DISCOVERED
WRITE DISCVRY REGISTER WITH RESPONSE CYCLE OF NODE TO BE DISCOVERED
HOST PROCESSOR WAITS FOR SUBORDINATE NODE DISCOVERY
PLL LOCKED?
NO
TERMINATE DISCOVERY
YES
REPEAT DISCOVERY UNTIL ALL NODES ARE DISCOVERED
INITIALIZE NODES
CONFIGURE SLOTFMT AND DATCT REGISTERS
APPLY NEWSRCT

Figure 3-3: Summarized Discovery Flow

4 A^{2}B Bus Architecture

The A^{2}B bus is a high-bandwidth bidirectional digital audio bus that is capable of transporting data and control information over distances, along with clock and power usingusing different power schemes. The following sections describes the A^{2}B bus architecture.

Bus Topology

The ***Bus Topology*** figure shows the line topology of the A^{2}B bus. The bus includes a single main node transceiver and multiple subordinate node transceivers connected using a daisy chain. The B-port of a node is connected to the A-port of next node via a cable.

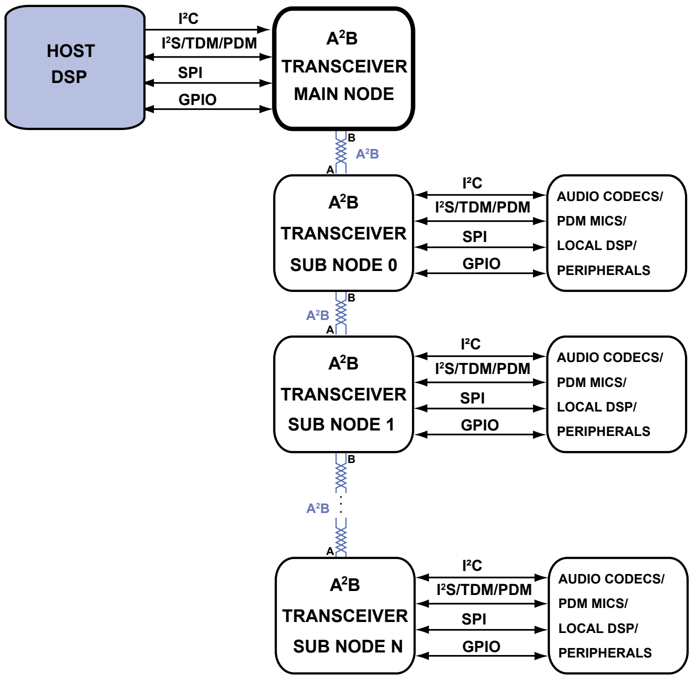

HOST DSP
I²C
I²S/TDM/PDM
SPI
GPIO
A²B TRANSCEIVER MAIN NODE
B
A²B
A
A²B TRANSCEIVER SUB NODE 0
AUDIO CODECS/ PDM MICS/ LOCAL DSP/ PERIPHERALS
A²B TRANSCEIVER SUB NODE 1
A²B TRANSCEIVER SUB NODE N

Figure 4-1: Bus Topology

The A^{2}B system connects I^{2}S/TDM, PDM, I^{2}C, SPI, and GPIO peripherals over distance. Each transceiver has local I^{2}S/TDM, PDM, I^{2}C, SPI, GPIO peripheral data ports. These ports are not directly connected to other transceiver ports. The transceivers are connected using the cable only. The transceiver embeds all of the peripheral data onto the A^{2}B bus in the differential signal format and sends the data over distance using the A^{2}B bus. Other transceivers in the A^{2}B chain consume the required data on the A^{2}B bus and recreate the data on their local I^{2}S/TDM, PDM, I^{2}C, SPI, and GPIO ports. The data is subsequently communicated to the connected peripherals.

- The I^{2}C peripheral is used for control purposes. The host processor that is connected to the main transceiver controls the whole A^{2}B bus including the main transceiver, all subordinate transceivers, and remote peripherals connected to the subordinate transceivers (via the I^{2}C port). The local processor connected to the subordinate transceiver can also access the transceiver registers; but, it cannot access other transceivers over the bus.
- I^{2}S/TDM is the main peripheral used for audio data communication. Each node can receive the data available on its TDM RX pins and send it over the A^{2}B bus to other nodes. The other nodes on the bus can receive this data and put it on its TDM TX pins. Therefore, the I^{2}S/TDM peripherals connected to different transceivers can communicate over long distances and exchange data.

  NOTE: Since the TDM port of the nodes are not directly connected to each other, the TDM settings of different nodes can be independently configured based on the interfaced TDM device.
- The transceiver supports a direct PDM microphone interface. It converts the PDM stream into I^{2}S format data. Therefore, no PDM stream processing is needed by the host or local processor.
- The SPI peripheral can be used for control purposes (same as I^{2}C) or it can be used to exchange data (same as I^{2}S/TDM port) between SPI devices connected over distance.
- The transceiver supports general purpose IO communication between distant nodes. The GPIO over distance feature replicates the IO status on a node to the other nodes automatically (without host intervention).

All of the peripheral data is embedded in the A^{2}B bus and communicated between nodes based on the transceiver configuration. The nodes recreate the data on the local peripheral port with a deterministic low latency.

The A^{2}B bus provides a synchronization signal to all of the subordinate nodes. The nodes extract clock information from the bus and use the clock to lock their PLL. The subordinate nodes can provide the main clock (MCLK) to the connected peripheral devices. Therefore, all nodes and devices in the A^{2}B system run synchronously without requiring a crystal or oscillator on the board. See the PLL section for details.

The A^{2}B bus can deliver power over the bus . The regulator voltage is transmitted over the A^{2}B bus as bus bias using an external MOSFET. The data signals from the A^{2}B transceiver are in a differential format (Manchester encoded) and are driven on top of the bus bias. The bus powered subordinate nodes extract the power from the A^{2}B bus to supply transceiver power domains, and the external load (for example, codec/PDM microphones or other peripheral devices). Refer to Power on the Bus for details.

Bus Packets (Superframes)

Data on the A^{2}B bus is exchanged between nodes in a fixed packet size known as a *superframe*.

- The A^{2}B main transceiver initiates superframes as periodic packets.

- The rate of the superframes is the same as the synchronization signal frequency (f_{SYNCM}) that the host processor supplies to the A^{2}B main node (typically, 44.1 KHz or 48 KHz audio rates)
- The superframe include 1024 bits
  - Bandwidth of the A^{2}B bus = 1024 x f_{SYNCM} = 49.152 Mbps for a 48 KHz audio rate
  - Each bus bit = SYSBCLK = 1 / (1024 x f_{SYNCM})
- Each superframe is divided into periods of downstream transmission, upstream transmission, and no transmission (where the bus is not driven).
  - The first part of superframe is downstream. The A^{2}B main node starts the downstream transmission by putting a 64-bit SCF on the bus, followed by downstream data slots. The downstream transmission flows from the main node through the middle subordinate nodes to the last-in-line subordinate node. Each node can consume the received downstream data slots and contribute data slots to the downstream nodes.
  - The second part of superframe is upstream. The last-in-line subordinate node starts the upstream transmission by putting a 64-bit SRF on the bus, followed by upstream data slots. The upstream transmission flows from the last-in-line subordinate node through the middle subordinate nodes to the main node. Each node can consume received upstream data slots and can contribute data slots to the upstream nodes.

The ***A^{2}B Superframe*** figure shows the basic frame structure (superframe) on the A^{2}B bus, through which all nodes communicate with each other.

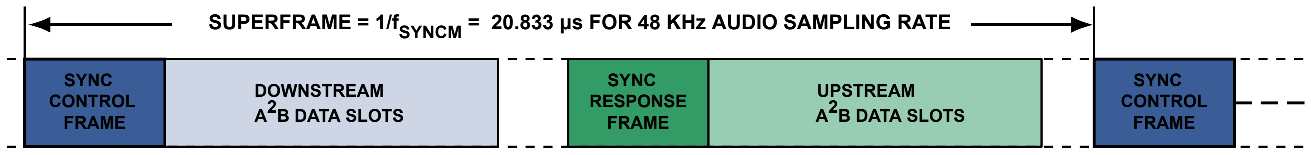

SUPERFRAME = 1/fSYNCM = 20.833 µs FOR 48 KHz AUDIO SAMPLING RATE
SYNC CONTROL FRAME
DOWNSTREAM A2B DATA SLOTS
SYNC RESPONSE FRAME
UPSTREAM A2B DATA SLOTS
SYNC CONTROL FRAME

Figure 4-2: A^{2}B Superframe

Synchronization Control Frame

The 64-bit SCF field marks the start of a superframe packet and the downstream traffic. The A^{2}B main node generates the SCF. It includes a synchronization signal, commands, and control information for the subordinate nodes. The SCF passes through the middle subordinate nodes to the last-in-line subordinate node; subordinate nodes do not modify the field. The ***SCF*** figure show the SCF field of a superframe.

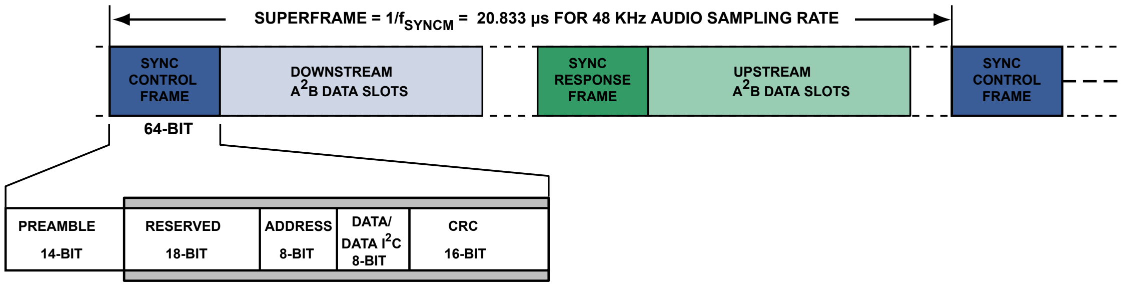


Figure 4-3: SCF

The main node uses the SCF field for following purposes:

- I^2C/SPI commands – when a host processor accesses the registers of a subordinate transceiver or a remote peripheral. The host processor initiates the access on the local I^2C/SPI interface. The A^2B main node embeds this command (register address for a read command and register data for a write command) into the SCF and passes it to the targeted subordinate node over the A^2B bus.
- Discovery frames – during the discovery process, the main node sends discovery frames to the targeted subordinate node. The SCF includes the node ID and the response cycle of node to-be-discovered. Upon discovery, the targeted node can respond with the SRF fields at a specified time. These frames have the lowest priority.
- GPIO over distance frames – when GPIO pins of the transceivers in the A^2B chain are mapped using the GPIO-over-distance feature, the GPIO pin status of a node is automatically reflected on the GPIO pins of other nodes. The A^2B main node evaluates the state of the virtual ports and communicates the information to subordinate nodes using SCF field. Refer to GPIO Over Distance for details. The GPIO over distance frames have a higher priority than I^2C/SPI commands (all internal operations).
- Interrupt handling frames – when a subordinate node raises an interrupt, it internally informs the A^2B main node using the SRF field. The A^2B main node then queries the subordinate node for details about the cause of the interrupt (INTTYPE) through the SCF field. These frames have the highest priority.

The 64-bit SCF field includes:

- a 14-bit preamble field to provide clock information to subordinate node PLL. It is a known pattern and not protected by CRC.
- a 18-bit control field. The control field consists of:
  - Bits related to the frame types (normal mode I^2C access, broadcast mode access, discovery mode access, and GPIO over distance)
  - 4-bit node field that indicates the targeted node
  - R/W bit to indicate read/write access
  - 2-bit header count field (used to keep track of header count synchronization)
- an 8-bit register address field

- an 8-bit data field for a write operation
- a 16-bit CRC field to protect the SCF field (except the preamble field)

Synchronization Response Frame

The 64-bit SRF field marks the upstream transmission. The last-in-line subordinate node generates the SRF. The SRF includes a synchronization signal, control, and status information for the main node. The SRF passes upstream from the last-in-line subordinate node through middle subordinate nodes to the main node. The middle subordinate nodes can modify the SRF field to communicate an interrupt or a response to a command from the main node. Therefore, the nodes closer to the main node have higher priority when conveying information to the main node. The ***SRF*** figure show the SRF field of a superframe.

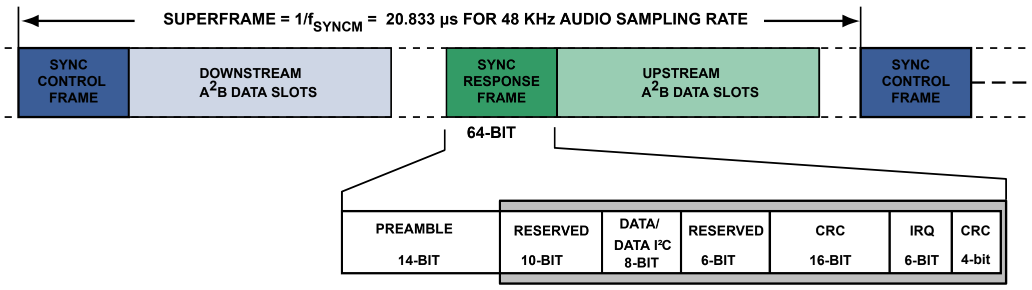


Figure 4-4: SRF

The SRF field is used for following purposes:

- Response to I^{2}C/SPI commands – when the host processor accesses the registers of a subordinate transceiver or remote peripheral. For a read command, the subordinate transceiver provides the data through the SRF field. It also conveys the ACK, NACK, or wait status for the I^{2}C access.
- GPIO over distance frames – when the GPIO input pin of the subordinate transceiver is mapped to a GPIO over distance virtual port and the input pin status changes. The subordinate transceiver conveys this status information to the main node over the SRF field.
- Interrupt reporting – the subordinate nodes reports the active interrupt and its details (INTTYPE) to the A^{2}B main node using the SRF field.

The 64-bit SRF field includes:

- a 14-bit preamble field to mark the start of the upstream part of the superframe. This field is used by upstream nodes for synchronization purposes and to detect a node drop condition. It is a known pattern and not protected by CRC. Unlike the preamble field in the SCF, the preamble field in the SRF is not used as an input for the node PLL.
- a 10-bit control field. It consists of:
    - access status bits for the acknowledging command received in the previous SCF (ACK, NACK, wait, retry)

  - a 4-bit node field to indicate which node generated or modified the SRF
  - a 2-bit header count field (used to keep track of header count synchronization)
- an 8-bit data field that provides a data byte for read operations
- a 6-bit reserved field
- a 16-bit CRC field that protects the 10-bit control, 8-bit data and 6-bit reserved fields. It does not protect the 14-bit preamble and 6-bit IRQ fields.
- a 6-bit IRQ. The subordinate node communicates to the main node that it has detected an interrupt. The field is comprised of:
  - a 1-bit IRQ bit to indicate that an active interrupt is available
  - a 4-bit NODE ID to indicate the subordinate node ID that generated the interrupt
- a 4-bit CRC. There is a separate CRC protection for the 6-bit IRQ field.

NOTE: The separate IRQ field along with CRC protection allows the upstream node to convey interrupt information without modifying the previous bits of the SRF. So, one node can respond to the I^2C command through the initial part of the SRF and another node can add interrupt information in a later part of the SRF.

Data Slots

The nodes exchange I^2S/TDM and SPI data using the data slot fields of the superframes. The superframes have two half duplex parts; the initial part is the downstream transmission and the later part is the upstream transmission. Data can be exchanged with any node in the A^2B system. The ***Data Slots*** figure shows the downstream and upstream data slots of a superframe with parity bits indicated in red.

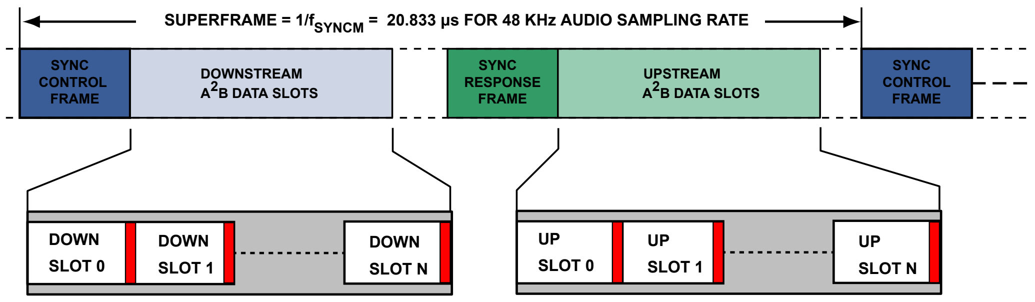


Figure 4-5: Data Slots

During first part of superframe, the A^2B main node starts the downstream by putting the SCF field on the bus, followed by downstream data slots that it must send to other A^2B nodes. The node receives this data from TDM RX pins and from the SPI port (if a SPI data tunnel operation is used) and puts it on the A^2B bus. Each subordinate transceiver receives the downstream data on its A-port. It can consume the required data slots from the bus and pass the data to downstream nodes using the B-port. It can also add the data slots that it must send to downstream nodes. The downstream traffic passes through the subordinate nodes towards last-in-line subordinate node.

During next part of the superframe, the last-in-line subordinate node starts the upstream flow by putting the SRF field on the bus, followed by upstream data slots that it must send to other A^2B nodes. Each subordinate transceiver receives the upstream data on its B-port. It can consume the required data slots from the bus and pass them up to upstream nodes using the A-port. It can also add the data slots that it must send to upstream nodes. The traffic goes through the subordinate nodes and reaches the main node, marking the end of upstream flow and the superframe. The downstream and upstream sequence repeats in each superframe. In this way, nodes can send their data to other nodes in the same superframe, irrespective of their position.

The A^2B bus data slot size can be: 8-bit, 12-bit, 16-bit, 20 bit, 24-bit, 28 bit, or 32-bit. The downstream slot size and upstream slot size can be independently configured such that it is not required to have same size for downstream and upstream data slots. However, all downstream slots must have same size as configured in the `A2B_SLOTFMT.DNSIZE` field; all upstream slots must have same size as configured in the `A2B_SLOTFMT.UPSIZE` field. The `A2B_SLOTFMT` is a, main node only, auto-broadcast register. So, when the host processor configures this register, the main node automatically broadcasts the downstream and upstream slot size to all subordinate nodes.

The slot formatting bits control the formatting of data slots. Downstream and upstream data slots format can be independently configured using the `A2B_SLOTFMT.DNFMT` and `A2B_SLOTFMT.UPFMT` bits. The normal (default) format of both upstream and downstream data slots is the data followed by a single parity bit, having an odd parity scheme. However, alternate formats supporting floating-point compression or ECC protection are also available by configuring the `A2B_SLOTFMT.DNFMT` and `A2B_SLOTFMT.UPFMT` bits.

ECC Protection for Data Slots

When data slot formatting is enabled for a 24-bit or 32-bit A^2B bus data slot size, ECC protection is added for each data slot (instead of parity bit protection). The ECC field is 6 bits for a 24-bit data slot size and 7 bits for a 32-bit data slot size. The ECC offers 1-bit error correction and multibit error detection. However, it increases the bus bandwidth usage.

ECC protection is useful in an environment where strong noise interferences (shorter than the superframe) are present, which otherwise can generate bit errors. ECC can be used in addition to the audio data error correction (repeat of last known good data). However, it may only be used for non-audio data because it requires extra bus bandwidth.

Floating-Point Data Compression

When data slot formatting is enabled for a 12-bit, 16-bit, or 20-bit slot size, the A^2B transceiver provides floating-point (FP) data compression and decompression in order to use less bandwidth on the A^2B bus for a given data size. At the data transmitter node, the 16-bit, 20-bit or 24-bit data channels on TDM bus are compressed to prepare 12-bit, 16-bit, or 20-bit A^2B bus slots, respectively. At the receiver node, the 12-bit, 16-bit, or 20-bit A^2B bus slots are decompressed to prepare the 16-bit, 20-bit, or 24-bit I^2S/IDM data channel, respectively.

The compression encodes the number of leading sign bits in the source data as a 3-bit field and concatenates the sign bit itself, followed by *N*-4 bits of data (where *N* is the A^{2}B data size). An example of 16-bit to 12-bit compression is shown in the ***16-Bit to 12-Bit Compression Example*** table. In the table, `s` is the sign bit and `~s` is the inverse of the sign bit.

Table 4-1: 16-Bit to 12-Bit Compression Example

16-Bit Data																-->	*12-Bit FP Data*											
s	~s	x	x	x	x	x	x	x	x	***y***	***y***	***y***	***y***	***y***	***y***	-->	0	0	0	s	x	x	x	x	x	x	x	x
s	s	~s	x	x	x	x	x	x	x	x	***y***	***y***	***y***	***y***	***y***	-->	0	0	1	s	x	x	x	x	x	x	x	x
s	s	s	~s	x	x	x	x	x	x	x	x	***y***	***y***	***y***	***y***	-->	0	1	0	s	x	x	x	x	x	x	x	x
s	s	s	s	~s	x	x	x	x	x	x	x	x	***y***	***y***	***y***	-->	0	1	1	s	x	x	x	x	x	x	x	x
s	s	s	s	s	~s	x	x	x	x	x	x	x	x	***y***	***y***	-->	1	0	0	s	x	x	x	x	x	x	x	x
s	s	s	s	s	s	~s	x	x	x	x	x	x	x	x	***y***	-->	1	0	1	s	x	x	x	x	x	x	x	x
s	s	s	s	s	s	s	~s	x	x	x	x	x	x	x	x	-->	1	1	0	s	x	x	x	x	x	x	x	x
s	s	s	s	s	s	s	s	~s	x	x	x	x	x	x	x	-->	1	1	1	s	x	x	x	x	x	x	x	x

Data decompression reverses the process. The LSB of the compressed data (***L*** in the ***12-Bit to 16-Bit Data Decompression Example*** table) is used to generate any remaining LSBs of the decompressed data that are not stored in the compressed format.

Table 4-2: Example of Data Decompression: 12 Bit to 16 Bit

12-Bit FP Data												-->	*16-Bit Decompressed Data*															
0	0	0	s	x	x	x	x	x	x	x	***L***	-->	s	~s	x	x	x	x	x	x	x	***L***	***L***	***L***	***L***	***L***	***L***	***L***
0	0	1	s	x	x	x	x	x	x	x	***L***	-->	s	s	~s	x	x	x	x	x	x	x	***L***	***L***	***L***	***L***	***L***	***L***
0	1	0	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	~s	x	x	x	x	x	x	x	***L***	***L***	***L***	***L***	***L***
0	1	1	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	s	~s	x	x	x	x	x	x	x	***L***	***L***	***L***	***L***
0	0	0	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	s	s	~s	x	x	x	x	x	x	x	***L***	***L***	***L***
0	0	1	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	s	s	s	~s	x	x	x	x	x	x	x	***L***	***L***
0	1	0	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	s	s	s	s	~s	x	x	x	x	x	x	x	***L***
0	1	1	s	x	x	x	x	x	x	x	***L***	-->	s	s	s	s	s	s	s	s	x	x	x	x	x	x	x	***L***

Selecting FP compression is a good method to reduce the data slot size. It is beneficial in systems that requires multiple data channels. Reducing the slot size also reduces the current draw, which can be important in bus-powered nodes.

The full dynamic range (24 bit = 144.49 dB) of the audio signal is preserved when data compression is enabled. The human ear can listen to sounds near the noise level in a quiet environment, but the human ear masks very quiet audio content in the presence of very loud audio content. The floating-point compression (to 20 bit) takes advantage of this psychoacoustic effect and removes low-level content in the presence of high-level audio content. The floating-point compression preserves all low-level content (here, 16-bits = 96.33 dB for 20-bit data slots) when there is no high-level audio content and supports the full dynamic range for strong audio signals (up to 144.49 dB for 20 bit data slots), always with 16 bit = 96.33 dB resolution.

NOTE: When the SPI tunnel is used, do not use the compression/decompression option.

The ***Slot Formatting*** table shows the number of data slot bits on the A^{2}B bus for different sizes of TDM data and data formatting types (ECC protection or FP compression).

Table 4-3: Slot Formatting

Slot Size	Slot Formatting (FMT)	TDM Data Size	Slot Protection	Bits per Data Slot
8-bit	0 (Normal)	8-bit	1-bit parity	9 (8-bits data + 1-bit parity)
	1 (N/A)	N/A	N/A	N/A
12-bit	0 (Normal)	12-bit	1-bit parity	13 (12-bits data + 1-bit parity)
	1 (FP)	16-bit	1-bit parity	13 (16-bits data compressed to 12-bits + 1 parity bit)
16-bit	0 (Normal)	16-bit	1-bit parity	17 (16-bits data + 1-bit parity)
	1 (FP)	20-bit	1-bit parity	17 (20-bits data compressed to 16-bits + 1 parity bit)
20-bit	0 (Normal)	20-bit	1-bit parity	21 (20-bits data + 1-bit parity)
	1 (FP)	24-bit	1-bit parity	21 (24-bits data compressed to 20-bits + 1 parity bit)
24-bit	0 (Normal)	24-bit	1-bit parity	25 (24-bits data + 1-bit parity)
	1 (ECC)	24-bit	6-bit ECC	30 (24-bits data + 6-bits ECC)
28-bit	0 (Normal)	28-bit	1-bit parity	29 (28-bits data + 1-bit parity)
	1 (N/A)	N/A	N/A	N/A
32-bit	0 (Normal)	32-bit	1-bit parity	33 (32-bits data + 1-bit parity)
	1 (ECC)	32-bit	7-bit ECC	39 (32-bits data + 7-bits ECC)

NOTE: In the ***Slot Formatting*** table, the I^{2}S/TDM data size indicates the width of the actual data being exchanged over the I^{2}S/TDM/PDM port in MSB first format. Data sizes in this column from 8 to 16 bits require that the `A2B_I2SGCFG.TDMSS` bit is set (16-bit TDM channel data width). Data sizes from 20 to 32 bits require that the `A2B_I2SGCFG.TDMSS` bit is cleared (32-bit TDM channel data width). Refer to I^{2}S /TDM Interface for details.

Mapping Between TDM Channels and A2B Slots

A2B nodes exchange control information over the SCF and SRF fields and the data over the data slots field. The I2S/TDM port is typically used to exchange data locally between the A2B transceiver and the connected audio peripherals. This configuration allows the transmission of I2S data over distance. The following sections describe how I2S/TDM data channels are mapped to and from the A2B bus data slots.

NOTE: The transceiver also supports SPI data over distance. The transceiver can receive data from a local SPI peripheral and send it over the A2B bus to a remote SPI peripheral using the data slots fields of a superframe. This transmission is known as an SPI tunnel operation; it shares the bus data slots bandwidth with I2S/TDM channels. See the SPI Interface for details about the mapping of SPI data to/from the A2B bus.

I2S /TDM Interface

I2S/TDM is a digital audio format used for audio data transfer. Typically, it consists of a bit clock (BCLK), frame synchronization signal (SYNC), data transmit lines (DTXn), and data receive lines (DRXn). The I2S protocol has two channels per frame, whereas the TDM interface supports multiple channels with TDM2/TDM4/TDM8/TDM16/TDM32 modes. The ***I2S/TDM Example Timing*** figure shows a typical waveform for TDM8 mode. There are eight channels per frame. Each channel is 16 or 32-bits wide. The A2B transceiver has five data pins that supports up to four transmit data pins and up to four receive data pins. For more information, see I2S/TDM Interface.

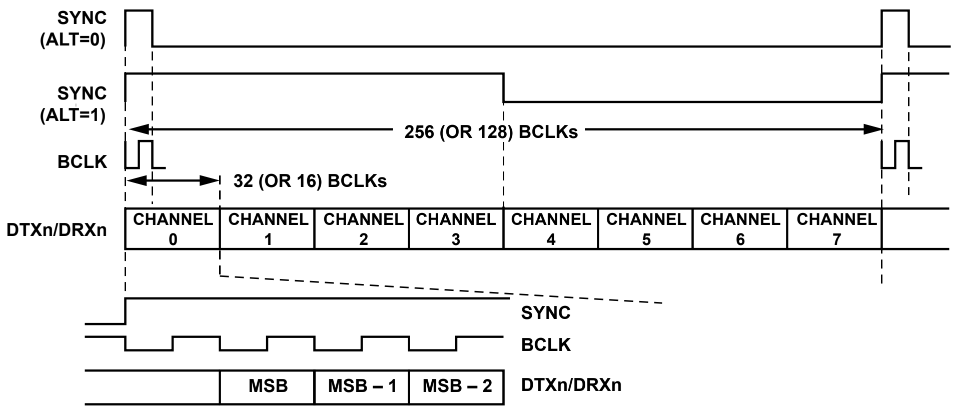


Figure 4-6: I2S/TDM8 Example Timing

If two data pins are used for transmit or receive operations, the channels can be mapped in an interleaved or non-interleaved format using the `A2B_I2SCFG` register. The ***Channel Mapping*** figure show the interleaved and non-interleaved channel options when using TDM8 mode and two data lines.

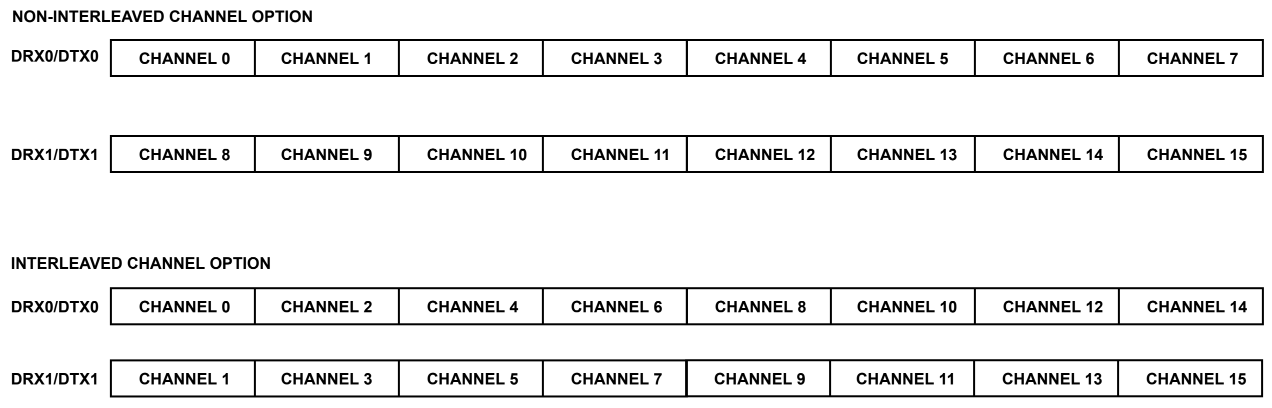


Figure 4-7: Channel Mapping

Channel Mapping

Typically, there is a one-to-one mapping between data channels on I²S/TDM bus and data slots on the A²B bus. The ***Default Channel Mapping*** figure shows the mapping in TDM8 mode using a single data line (DTX0/DRX0).

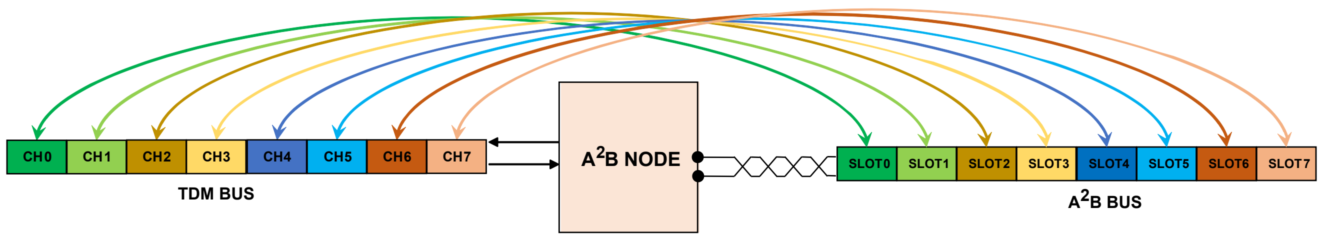


Figure 4-8: Default Channel Mapping

NOTE: When a node receives data from the A²B bus, the arrow direction shown in the ***Default Channel Mapping*** figure is from the A²B bus to the DTX0 pin; when a node sends data to other nodes using the A²B bus, the arrow direction is from the DRX0 pin to the A²B bus.

If two transmit or receive data pins are used, the mapping between data channels on the I²S/TDM bus and data slots on the A²B bus depend on whether the interleaved or non-interleaved option is selected in the `A2B_I2SCFG` register. The ***Multipin Channel Mapping*** figures show these options when using TDM4 mode with two data lines.

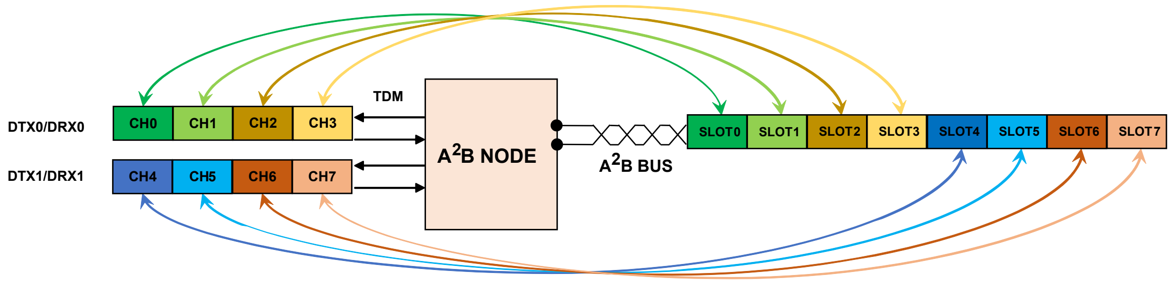


Figure 4-10: Multipin Channel Mapping - Non-Interleaved

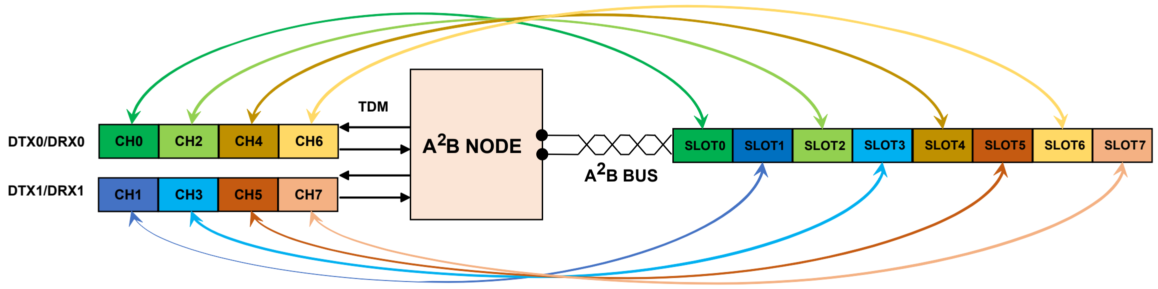


Figure 4-11: Multipin Channel Mapping - Interleaved

NOTE: Transmit data pins (DTXn) and receive data pins (DRXn) can be configured independently with interleave options.

A^{2}B transceivers support I^{2}S/TDM data channel sizes of either 16 bits or 32 bits. The data is in MSB bit first, left-justified format. For example, if an A^{2}B transceiver must pass 24-bit audio data over the A^{2}B bus to other nodes, the TDM channel size should be configured as 32 bits. The data should be presented to the DRX pins of the A^{2}B transceiver in MSB bit first, left-justified format (TDM data [31:8] as actual data). The A^{2}B transceiver picks the MSB bits based on the A^{2}B data slots size (8 bit, 12 bit, 16 bit, 20 bit, 24 bit, 28 bit, or 32 bit) and ignores the remaining bits of the TDM channel. Similarly, the receiving node gets the A^{2}B slots from the bus and presents the data on the DTX pins in MSB bit first, left-justified format (drives the data bits in TDM data[31:8] and the lower bits as 0). In this case, the A^{2}B slot size should be configured as 24 bits. An A^{2}B slot size of 32 bits also works; but, the size wastes A^{2}B bus bandwidth because all of the 32-bit data (including the padded 0 bits) is transmitted over bus. If the TDM interfaced device cannot work in left-justified data, and, instead, has right-justified data format, the A^{2}B transceiver must be configured for a 32-bit A^{2}B bus slot. In this case, it can transmit all 32 bits including LSB bits that have data.

I^{2}S /TDM Frame Buffers

Each A^{2}B transceiver features two internal frame buffers for storing I^{2}S/TDM data temporarily. The RX frame buffer is on the DRXn pins and the TX frame buffer is on the DTXn pins. These buffers are 32 locations deep; each location is 32-bits wide.

RX Frame Buffer

The RX frame buffer is populated on the data receive pins (DRXn) of the I^{2}S/TDM port. In each superframe, the data channels on the DRXn pins are stored in the RX frame buffer (based on the number of enabled receive pins and the pin interleaving option). The data is placed on the A^{2}B bus in the next superframe based on the configuration of the SLOT registers.

The ***RX Frame Buffer*** figure shows how data is received into the buffer and transmitted up or downstream. The buffer is 32 locations deep because any transceiver can contribute up to 32 slots on the A^{2}B bus (depending on bus bandwidth availability). Each location of frame buffer is 32-bit wide, which is the maximum bus slot size supported by A^{2}B transceivers.

- The main node passes the RX frame buffer contents downstream
- The last-in-line subordinate node passes the RX frame buffer contents upstream
- The middle subordinate nodes can pass the RX frame buffer contents downstream, upstream, or in both directions.

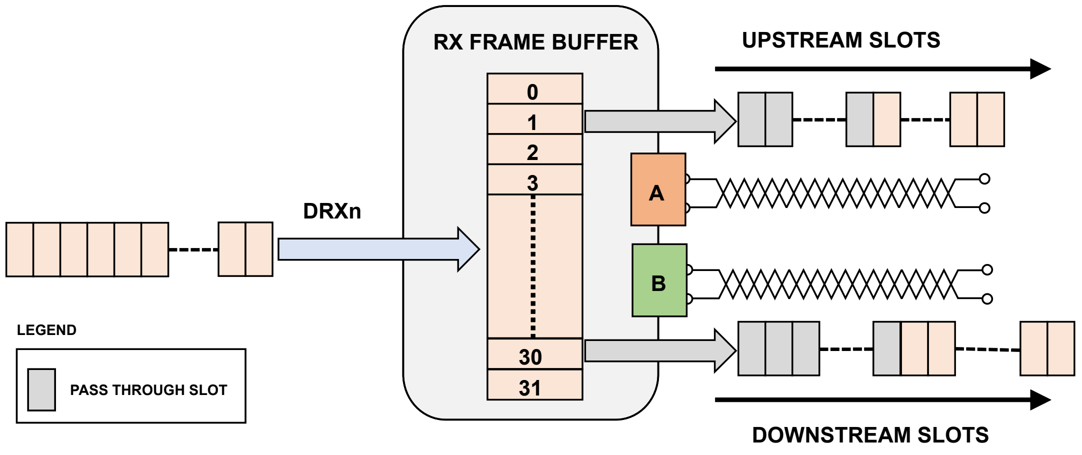


Figure 4-11: RX Frame Buffer

NOTE: Only the middle subordinate nodes have the gray colored pass-through slots shown in the ***RX Frame Buffer*** figure. The main node and the last-in-line subordinate node do not have pass-through slots. The slots contributed by middle node are appended after these pass-through slots.

TX Frame Buffer

The TX frame buffer is populated on the data transmit pins (DTXn) of the I^{2}S/TDM port. In each superframe, the node receives downstream and/or upstream data slots from the A^{2}B bus based on the configuration of the SLOT registers. The node stores data in the TX frame buffer. The frame buffer contents are then placed on the DTXn pins of the TDM port in the next superframe (based on the number of transmit data pins enabled and the selected pin interleaving option).

The ***TX Frame Buffer*** figure shows how data is received into the buffer from upstream or downstream slots. The buffer is 32 locations deep. Any transceiver can consume up to 32 combined downstream and upstream slots from the A^{2}B bus. If a transceiver is configured to consume more than 32 bus slots, the additional data is dropped. Each location of frame buffer is 32-bit wide, which is maximum bus slot size supported by A^{2}B transceivers.

- The main node populates the TX frame buffer contents with the received upstream data
- The last-in-line subordinate node populates the TX frame buffer contents with the received downstream data
- The middle subordinate node populates the TX frame buffer contents with either downstream slots, upstream slots, or both. The consumed downstream slots are stored first, occupying the lower-order locations of the buffer. Consumed upstream slots are then stored in the higher-order locations of the TX frame buffer.

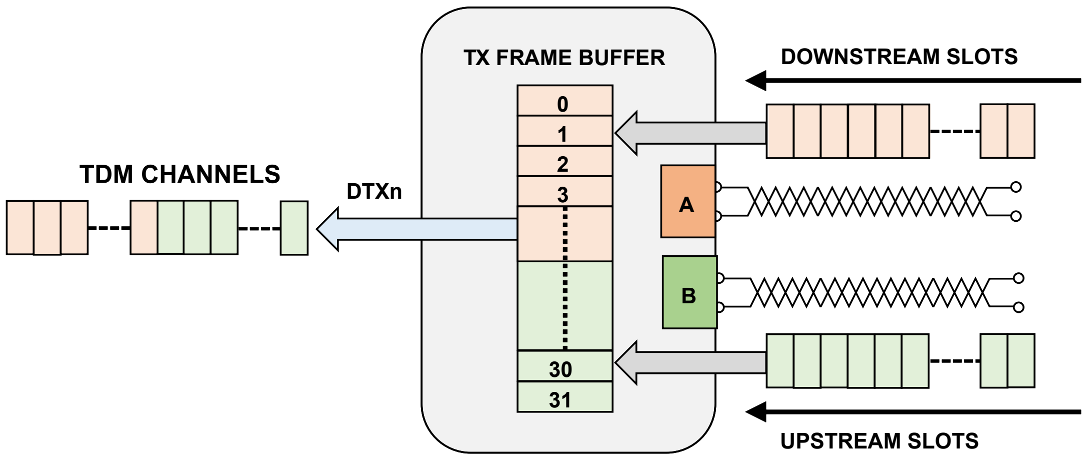


Figure 4-12: TX Frame Buffer

NOTE: The TX and RX frame buffers are populated in each superframe and drained in next superframe. There exists a ping-pong mechanism to support the availability of data in each superframe. However, for simplicity, the ***TX Frame Buffer*** figure shows a single frame buffer.

I^{2}S /TDM Flexible Mapping

The A^{2}B transceiver provides a versatile option for mapping data between the TDM port and the A^{2}B bus. To reorder mapping, the transceiver uses two I^{2}S crossbars that exist between the TDM pins and the internal TX/RX frame buffers.

I^{2}S TX Crossbar

When a node receives data over the A^{2}B bus and places it on the I^{2}S/TDM interface, by default, there is a one-to-one mapping between the data received from the A^{2}B bus and the data placed on DTXn pins of the I^{2}S/TDM port. For example, channel 0 on the DTXn pin comes from TX frame buffer location 0, which is populated from the first received bus slot, channel 1 on the DTXn pin comes from TX frame buffer location 1, which is populated from the second received bus slot, and so on.

The ***TX Crossbar*** figures shows the flow of data through the TX crossbar. Using the I^{2}S TX crossbar, the channels on the I^{2}S/TDM interface can be reordered. The TX crossbar block is placed after the TX frame buffer and before the DTXn pins of the I^{2}S/TDM port.

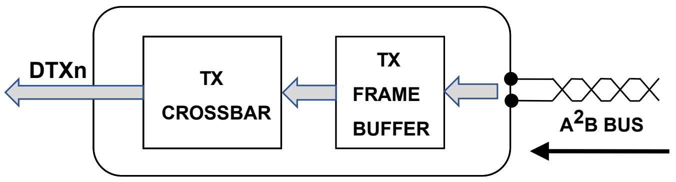


Figure 4-13: I^{2}S TX Crossbar

Once data is received from the A^{2}B bus (based on the configuration of the SLOT registers), it is placed in the TX frame buffer. The reordering of data is performed in TX crossbar block and then the data is presented on the DTXn pins of the I^{2}S/TDM port. Each transmit channel on a DTXn pin is assigned with a crossbar register that defines the frame buffer location for the corresponding channel. There are total of 32 transmit crossbar registers (`A2B_TXXBAR0` through `A2B_TXXBAR31`), one for each I^{2}S/TDM transmit channel. For example, writing the `A2B_TXXBAR7` register with 0x2 maps the second received bus slot (TX frame buffer location 2) on channel 7 of the I^{2}S/TDM port.

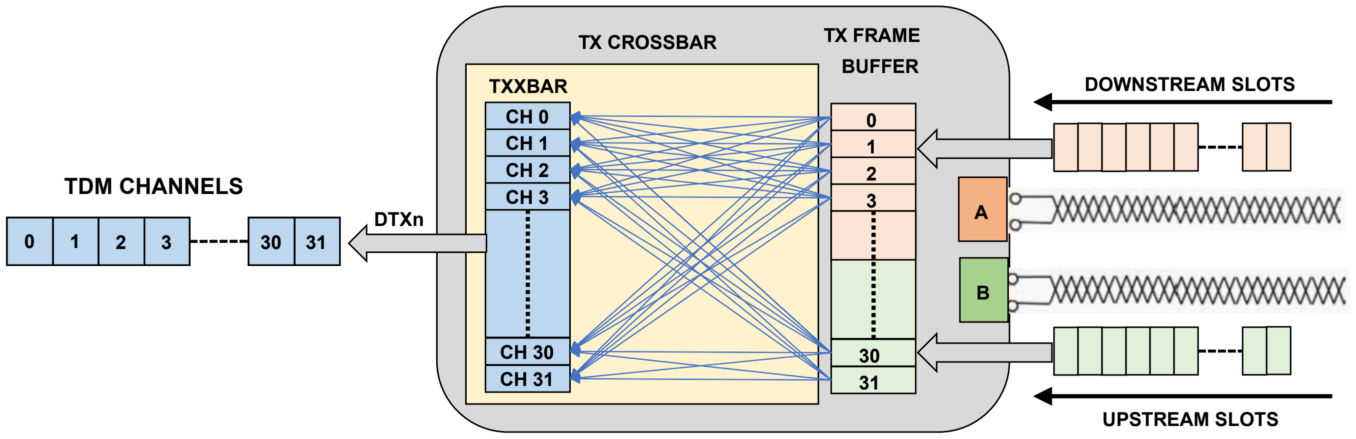


Figure 4-14: I^{2}S TX Crossbar - TDM Channels

NOTE: The default values of the TX crossbar registers match the legacy behavior for AD24xx parts.

The transceiver supports up to 32 transmit channels on DTXn pins. Higher numbered transmit channels are driven with zero. The ***TX Channel Assignments*** figure shows the I^{2}S/TDM channel assignments for a 3-pin transmission in TDM16 mode.

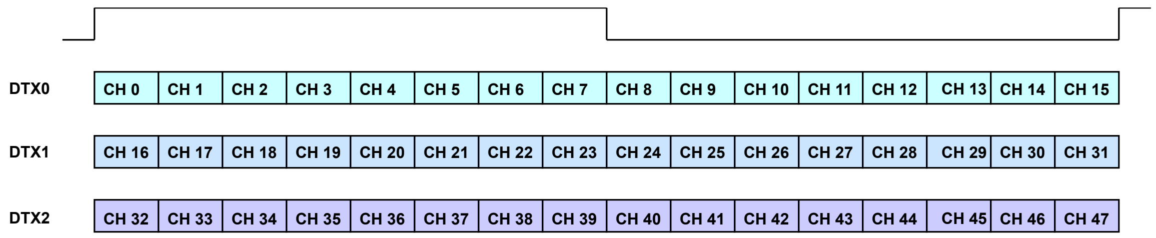

DTX0
CH 0 CH 1 CH 2 CH 3 CH 4 CH 5 CH 6 CH 7 CH 8 CH 9 CH 10 CH 11 CH 12 CH 13 CH 14 CH 15
DTX1
CH 16 CH 17 CH 18 CH 19 CH 20 CH 21 CH 22 CH 23 CH 24 CH 25 CH 26 CH 27 CH 28 CH 29 CH 30 CH 31
DTX2
CH 32 CH 33 CH 34 CH 35 CH 36 CH 37 CH 38 CH 39 CH 40 CH 41 CH 42 CH 43 CH 44 CH 45 CH 46 CH 47

Figure 4-15: TX Channel Assignments

The ***TX Frame Buffers*** figure shows the 32 frame buffer entries. Channels 0 through 31 can be driven with default mapping (one-to-one with TX frame buffer locations) or with flexible mapping using the TX crossbar. Any not-driven channels (channels 32 through 47) are fed with zero data.

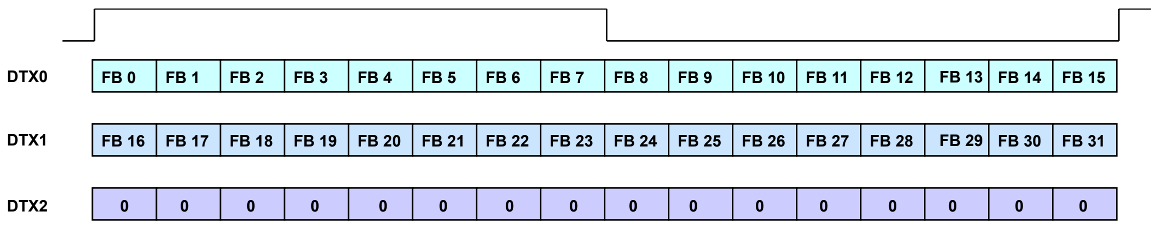

DTX0
FB 0 FB 1 FB 2 FB 3 FB 4 FB 5 FB 6 FB 7 FB 8 FB 9 FB 10 FB 11 FB 12 FB 13 FB 14 FB 15
DTX1
FB 16 FB 17 FB 18 FB 19 FB 20 FB 21 FB 22 FB 23 FB 24 FB 25 FB 26 FB 27 FB 28 FB 29 FB 30 FB 31
DTX2
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4-16: TX Frame Buffers

TDM Transmit Channel Offset

The A^{2}B main transceiver provides an option to insert a channel offset on the DTXn pin. The `A2B_I2STXOFFSET` register controls the number of I^{2}S transmit channels to be skipped before the main node begins transmitting data. The ***TX Frame Buffers with Offset*** figure shows the default mapping between the TX frame buffer and I^{2}S/TDM channels with `A2B_I2STXOFFSET.TXOFFSET`=10.

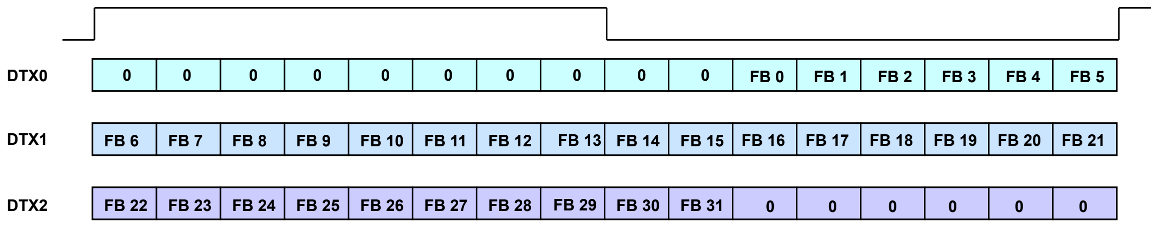

DTX0
0 0 0 0 0 0 0 0 0 0 FB 0 FB 1 FB 2 FB 3 FB 4 FB 5
DTX1
FB 6 FB 7 FB 8 FB 9 FB 10 FB 11 FB 12 FB 13 FB 14 FB 15 FB 16 FB 17 FB 18 FB 19 FB 20 FB 21
DTX2
FB 22 FB 23 FB 24 FB 25 FB 26 FB 27 FB 28 FB 29 FB 30 FB 31 0 0 0 0 0 0

Figure 4-17: TX Frame Buffers with Offset

The function of the `A2B_I2STXOFFSET` register depends on the pin interleaving option. To understand this, consider an example where the main node is receiving eight upstream data slots from the A^{2}B bus and drives the data on the DTX0 and DTX1 pins in TDM4 mode. The ***Data Channels*** figure shows how the pin interleaving option and the A2B_I2STXOFFSET register affect the mapping. The transceiver supports the data channels interleaved (enable/disable) option for 2-pin TX mode only. The 1-pin, 3-pin and 4-pin TX operations support the non-interleaved option only. Two transmit pins are used in this example. If multiple data pins are used, the channels are placed and shifted accordingly.

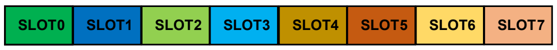


Figure 4-18: Data Slots on the Bus

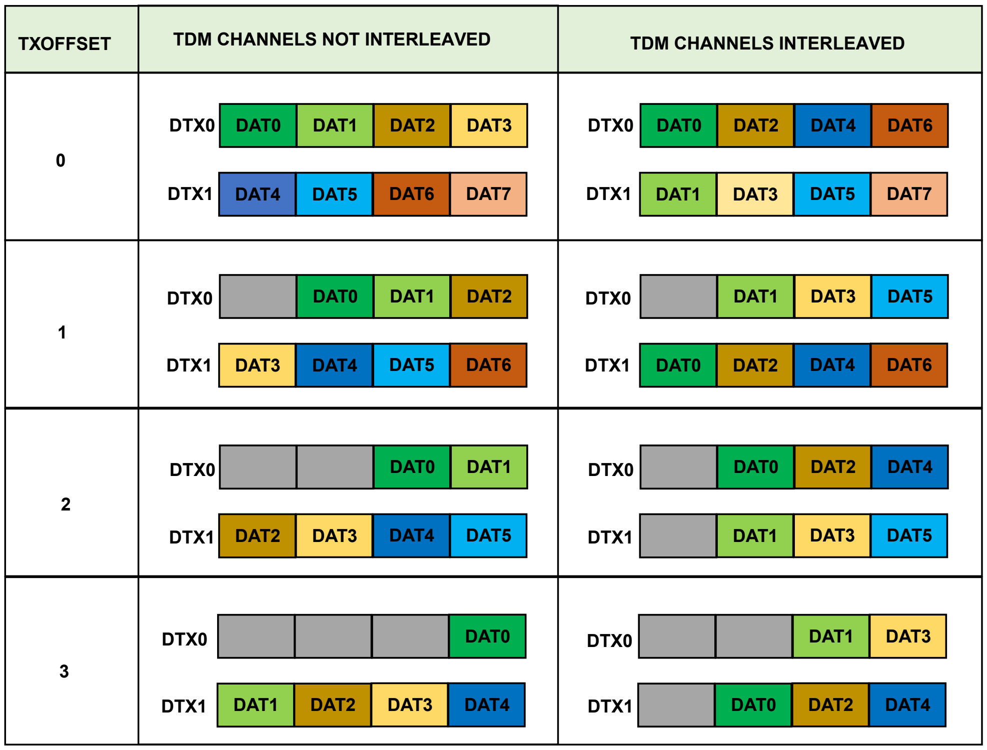


Figure 4-19: Data Channels

NOTE: There is no transmit pin offset support in subordinate transceivers. The A2B_I2STXOFFSET register is for the main node only.

TX crossbar settings will take effect without setting the A2B_CONTROL.NEWSTRCT bit.

I^{2}S RX Crossbar

By default, the I^{2}S/TDM channels are received continuously from channel 0 into the RX frame buffer and mapped one-to-one onto the A^{2}B bus. Therefore, when using TDM16 mode, the two data receive pins (DRX0/1) fill the 32 locations in the frame buffer. Using 3-pin RX and 4-pin RX modes can restrict TDM mode settings. It is possible that not all channels on the DRX pins are valid and the default node configurations cannot skip reception of those channels.

The ***RX Crossbar*** figures shows the flow of data through the RX crossbar. Using the I^{2}S RX crossbar, TDM channels can be selectively received from a total of 64 channels that span multiple TDM DRXn lines. The RX crossbar block is placed after the DRXn pins of I^{2}S/TDM port and before the RX frame buffer.

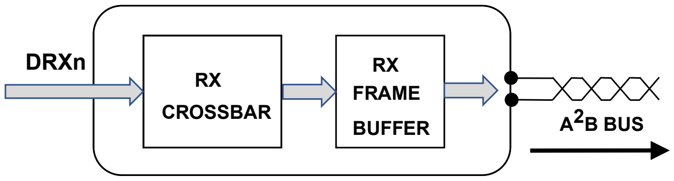


Figure 4-20: RX Crossbar

The RX crossbar registers (`A2B_RXMASK0` through `A2B_RXMASK7`) select which TDM channels are received into the RX frame buffer. These registers contain a mask bit for each TDM channel. Eight registers provide mask bits for 64 receive channels. Setting the mask bit indicates that the corresponding RX TDM channel is valid; the channel is loaded into the RX frame buffer, and data is sent over A^{2}B bus upstream or downstream.

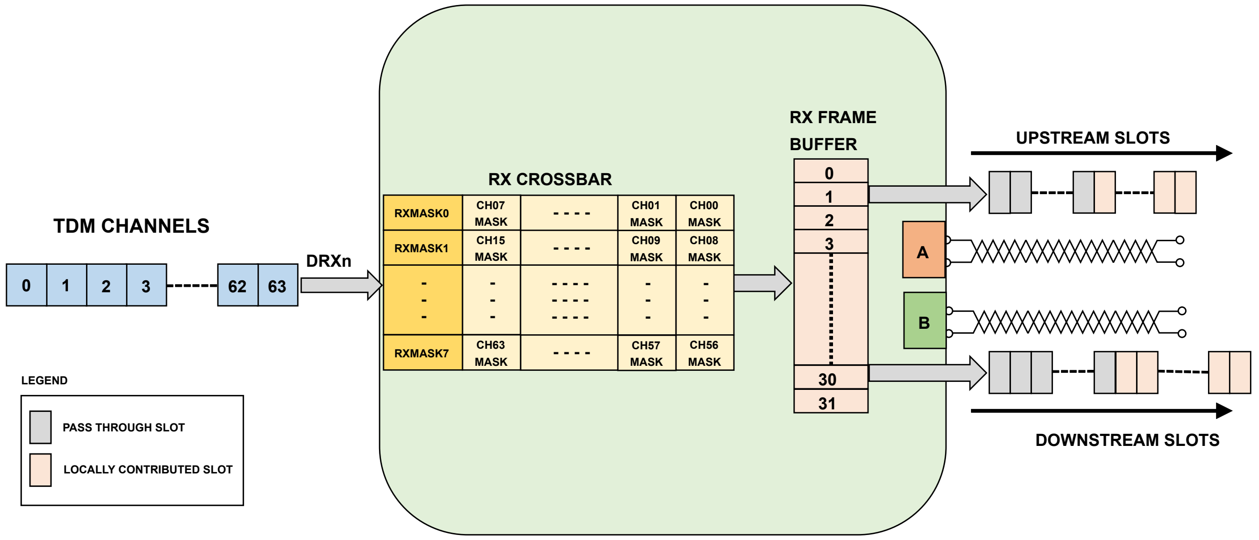


Figure 4-21: RX Crossbar - TDM Channels

NOTE: By default, all bits of the `RXMASK[7:0]` registers are set (=1). The RX frame buffer receives all TDM channels until it is completely full (legacy behavior). Bits can be cleared (= 0) to have the associated TDM data slot ignored.

The ***RXMASK Assignments*** figures show the RXMASK to I^{2}S/TDM data channel assignments for eight channels (TDM8) using non-interleaved and interleaved data retrieval and an example of the frame buffer contents for `A2B_RXMASK0` = 0xC3, `A2B_RXMASK1` = 0x66 settings. Only two data lines (DRX0/1) are enabled in TDM8 mode.

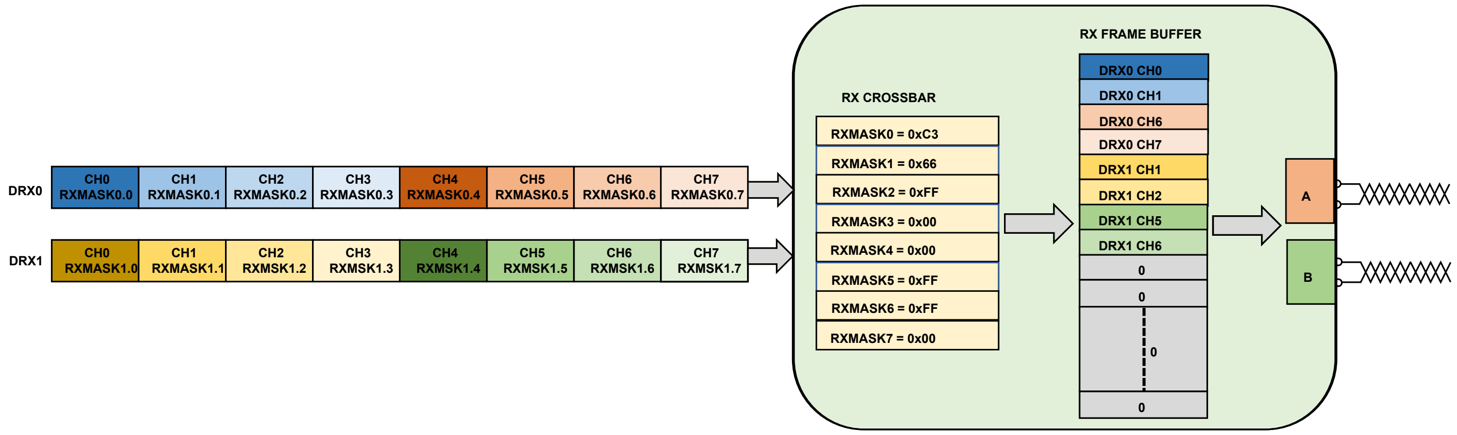


Figure 4-22: RXMASK Assignments - Non-Interleaved

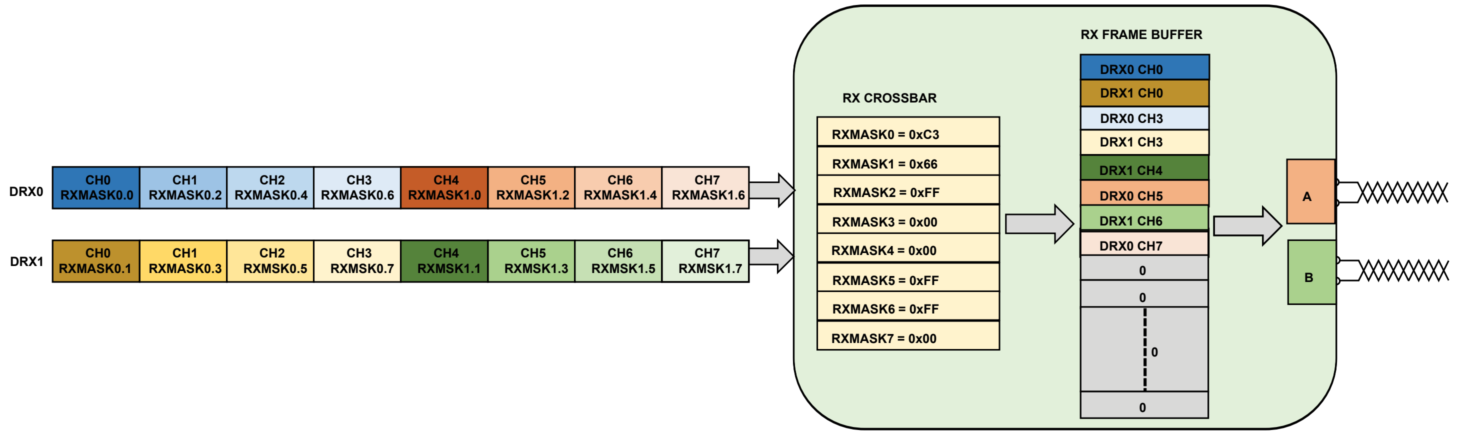


Figure 4-23: RXMASK Assignments - Interleaved

NOTE: RX crossbar settings take effect without setting the `A2B_CONTROL.NEWSTRCT` bit.

TDM Receive Channel Offset

In A²B subordinate transceivers, there are two registers to support offset on the DRXn pins for downstream and upstream data contribution. These offset registers do not work on TDM pins directly. Instead, the offsets are applied at the RX frame buffer level. The frame buffer is filled from channel 0 as per RX crossbar settings. The programmed offset registers are applied while placing the downstream and upstream data on the bus.

- The `A2B_DNOFFSET` register defines the number of RX frame buffer locations to be skipped before the channels are passed downstream on the A²B bus
- The `A2B_UPOFFSET` register defines the number of RX frame buffer locations to be skipped before the channels are passed are upstream on the A²B bus

NOTE: It is possible to send the same data channels (for example, microphone data) to both downstream and upstream nodes. In a subordinate node, there is no register to insert the offset on the DTXn pins.

Slot Register Configuration

The nodes can be configured for slot settings that define the number of slots :

- Consumed from the A^{2}B bus
- Contributed to the A^{2}B bus
- Passing through to other nodes

The downstream and upstream traffic on the A^{2}B bus depend on the configuration of the SLOT registers. Like most of the configuration registers, the SLOT registers are configured during the discovery and initialization process and take effect after writing to the `A2B_CONTROL.NEWSTRCT` bit.

NOTE: It is possible to change the SLOT registers during run time. However, this can result in unpredictable TDM channel mapping for a few superframes. Changes to the SLOT registers take effect after applying the `A2B_CONTROL.NEWSTRCT` bit. The change occurs through the I^{2}C/SPI interface and is asynchronous with the TDM interface.

Main Node Slot Configuration

In the A^{2}B main node, two slot registers are used to define the A^{2}B bus traffic on the B-port. These registers define the I^{2}S/TDM channels only and do not include the SPI tunnel slots.

- Downstream Slots Register (`A2B_DNSLOTS`) – This register defines the number of I^{2}S/TDM slots the node sends downstream to the subordinate nodes. The TDM channels are received from TDM receive (DRXn) pins of the transceiver through the RX crossbar block. The data channels are placed in the RX frame buffer and then put on the A^{2}B bus during the downstream slots of the superframes. The data from slot 0 to slot `A2B_DNSLOTS.DNSLOTS` -1 is put on the bus.
- Upstream Slots Register (`A2B_UPSLOTS`) –This register defines the number of upstream slots being received from the A^{2}B bus. During upstream portion of the superframe, the node receives the slots from the A^{2}B bus and places the slots in the TX frame buffer. The frame buffer contents are then placed on the TDM transmit (DTXn) pins of the transceiver through the TX crossbar. An optional channel offset is defined in the `A2B_I2STXOFFSET` register. The upstream slots are received from slot 0 to slot `A2B_UPSLOTS.UPSLOTS` -1 on the bus. There is no option for selecting slots like the one available in the subordinate node `A2B_UPMASK0` - `A2B_UPMASK3` registers.

The ***Main Node SLOT Registers*** figure shows the flow of data through the slot registers of the main node. All other SLOT registers are used only for the subordinate node registers.

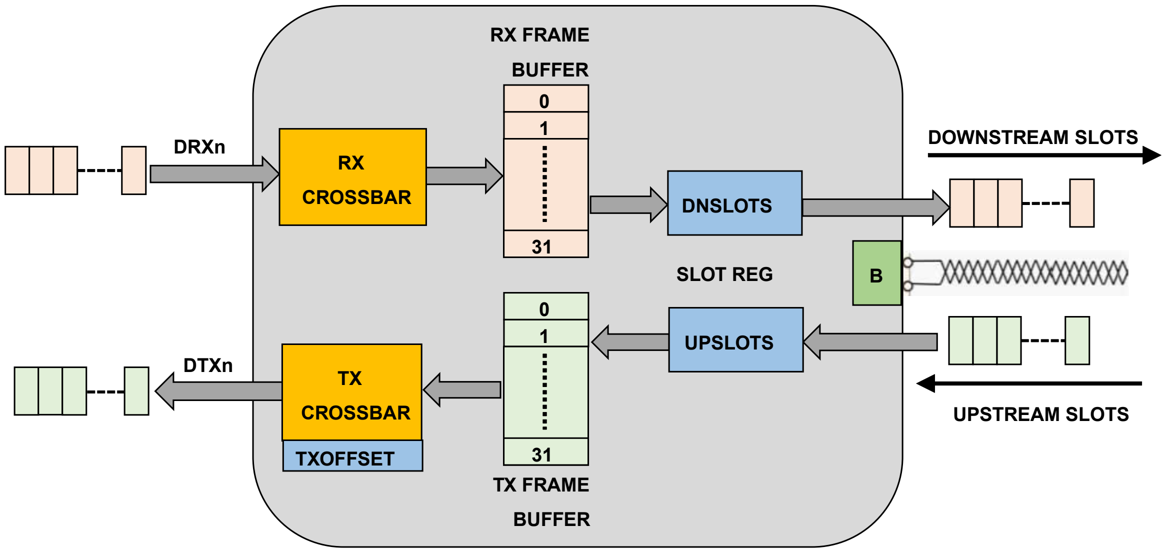


Figure 4-24: Main Node SLOT Registers

The following register DO NOT apply to the main node;

- Broadcast Downstream Slot register (`A2B_BCDNSLOTS`)
- Local Downstream Slot register (`A2B_LDNSLOTS`)
- Local Upstream Slot register (`A2B_LUPSLOTS`)
- Downstream Data RX Mask 0 through 3 ((`A2B_DNMASK0` - `A2B_DNMASK3`)
- Upstream Data RX Mask 0 through 3 (`A2B_UPMASK0`- `A2B_UPMASK3`)
- Local Upstream Channel Offset register (`A2B_UPOFFSET`)
- Local Downstream Channel Offset register (`A2B_DNOFFSET`).

Subordinate Node Slot Configuration

While the main node has a single port (B-port) for downstream transmission and upstream reception, the A^{2}B subordinate nodes use both the A-port and B-port during the downstream and upstream part of the superframe. The subordinate node can consume bus slots, contribute its own slots, and pass through the slots from one port to another. The subordinate node has more SLOT registers than the main node (which has only two SLOT registers).

NOTE: The following registers define the I^{2}S/TDM channels only and do not include SPI tunnel slots.

Downstream

The subordinate node uses the following registers for downstream transmission:

- Downstream Slots register (`A2B_DNSLOTS`) – This register defines the number of bus slots passed from the A-port to the B-port by the local node during the downstream portion of the superframe. Bus slots from slot 0 to slot (`A2B_DNSLOTS.DNSLOTS` -1) are passed down.

- Broadcast Downstream Slots register (A2B_BCDNSLOTS) – This register defines the number of data slots which are consumed by the node and passed downstream (B-port) as broadcast data to the next node.
- Downstream Data RX Mask 0 through 3 (A2B_DNMASK0 - A2B_DNMASK3) – These registers provide one bit for each possible downstream data slot. Four registers address up to 32 downstream slots. These downstream mask bits select which downstream slots are consumed by the transceiver and placed in its TX frame buffer for output over the I^2S/TDM port.
- Local Downstream Channel Offset register (A2B_DNOFFSET) – This register is used when the subordinate transceiver is configured to contribute downstream data. Data is placed in the enabled downstream slots starting with the beginning of the RX frame buffer (unless the A2B_DNOFFSET register has been programmed to apply an offset into the RX frame buffer from which it begins populating the enabled downstream slots).
- Local Downstream Slots register (A2B_LDNSLOTS) – The meaning of the A2B_LDNSLOTS register depends on the downstream broadcast mask enable (A2B_LDNSLOTS.DNMASKEN) bit.
  - If A2B_LDNSLOTS.DNMASKEN = 0 (default), the A2B_LDNSLOTS register defines the number of data slots that the local node consumes during the downstream portion of the superframe; the slots are not passed downstream to the next node. Therefore, the node receives A2B_BCDNSLOTS + A2B_DNSLOTS + A2B_LDNSLOTS downstream data slots on the A-port and transmits A2B_BCDNSLOTS + A2B_DNSLOTS downstream data slots on the B-port. In this case, the node does not contribute any slots to the downstream nodes. The A2B_DNMASK0 - A2B_DNMASK3 registers are ignored.

    The ***LDNSLOTS Register - DNMASKEN = 0*** figure shows the local downstream slots when A2B_LDNSLOTS.DNMASKEN = 0.

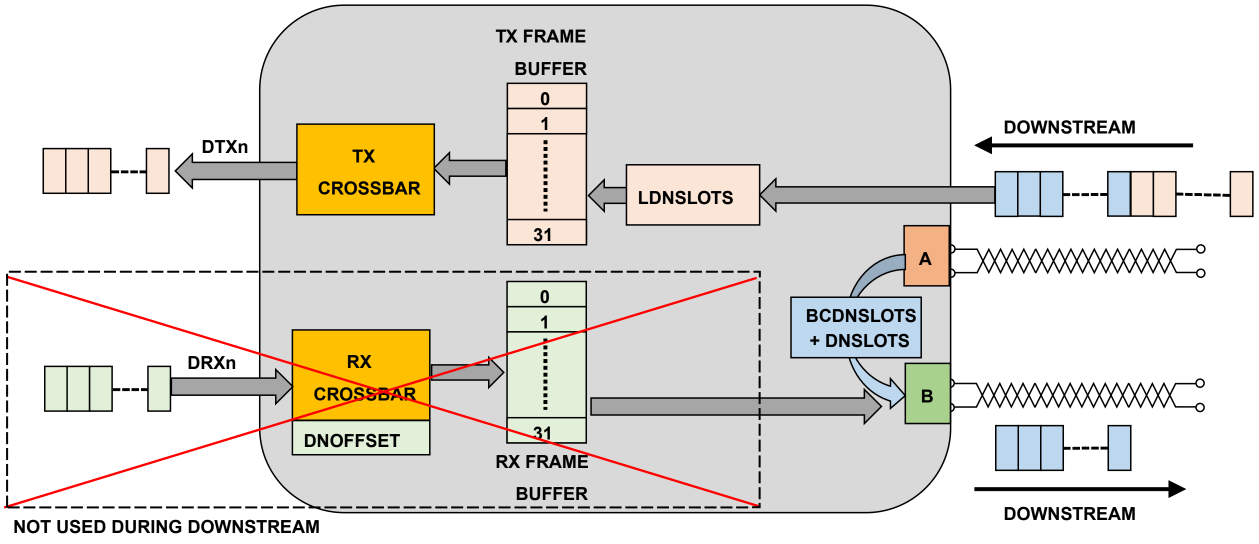


Figure 4-25: LDNSLOTS Register - DNMASKEN = 0

  - If A2B_LDNSLOTS.DNMASKEN = 1, the A2B_LDNSLOTS register defines the number of data slots that the local node contributes during the downstream portion of the superframe. These slots are appended to the A2B_DNSLOTS data slots that are passed downstream by the node. In this case, the downstream data RX mask (A2B_DNMASK0 - A2B_DNMASK3) registers can be configured to define the slots to be consumed.

The most significant bit that is set in the (A2B_DNMASK0 - A2B_DNMASK3) registers determines the number of slots that must be received by the transceiver (dnmaskrx) for it to then identify which individual slots are placed in its RX frame buffer for output over the I^2S/TDM port. A subordinate node receives MAX (A2B_DNSLOTS, dnmaskrx) downstream data slots on the A-port and transmits A2B_DNSLOTS + A2B_LDNSLOTS downstream data slots on the B-port. The A2B_BCDNSLOTS register is ignored in this case.

The ***LDNSLOTS Register - DNMASKEN = 1*** figure shows the local downstream slots when A2B_LDNSLOTS.DNMASKEN = 1.

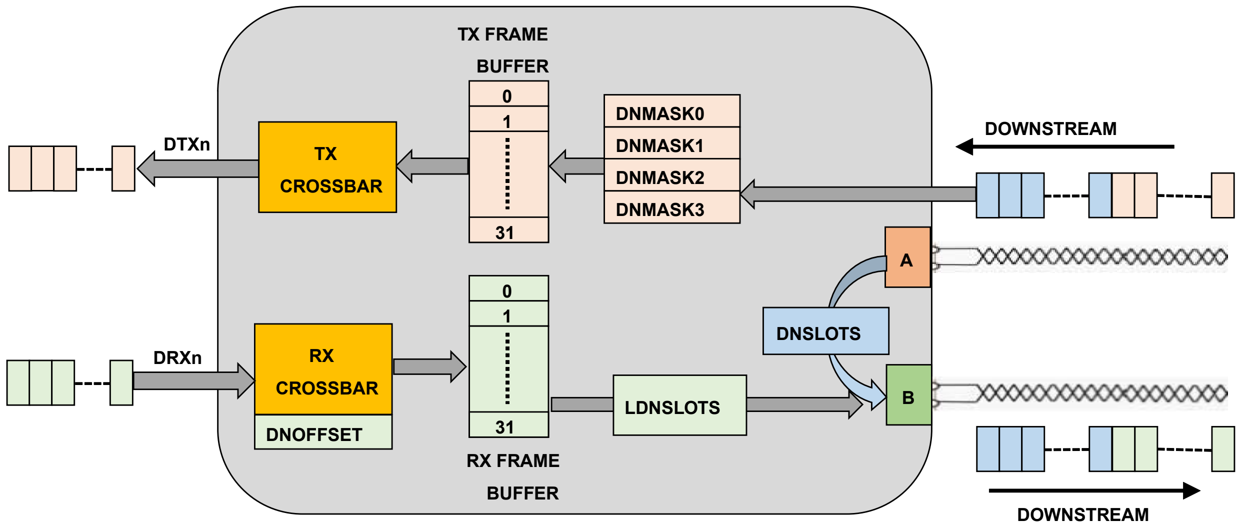


Figure 4-26: LDNSLOTS Register, DNMASKEN = 1

The ***Downstream Data Slots*** figure provides an example of how downstream data slots are used in a subordinate transceiver after programming the A2B_DNSLOTS, A2B_DNMASK0 - A2B_DNMASK3, A2B_LDNSLOTS, A2B_DNOFFSET registers when A2B_LDNSLOTS.DNMASKEN = 1 with default I^2S TX/RX crossbar settings.

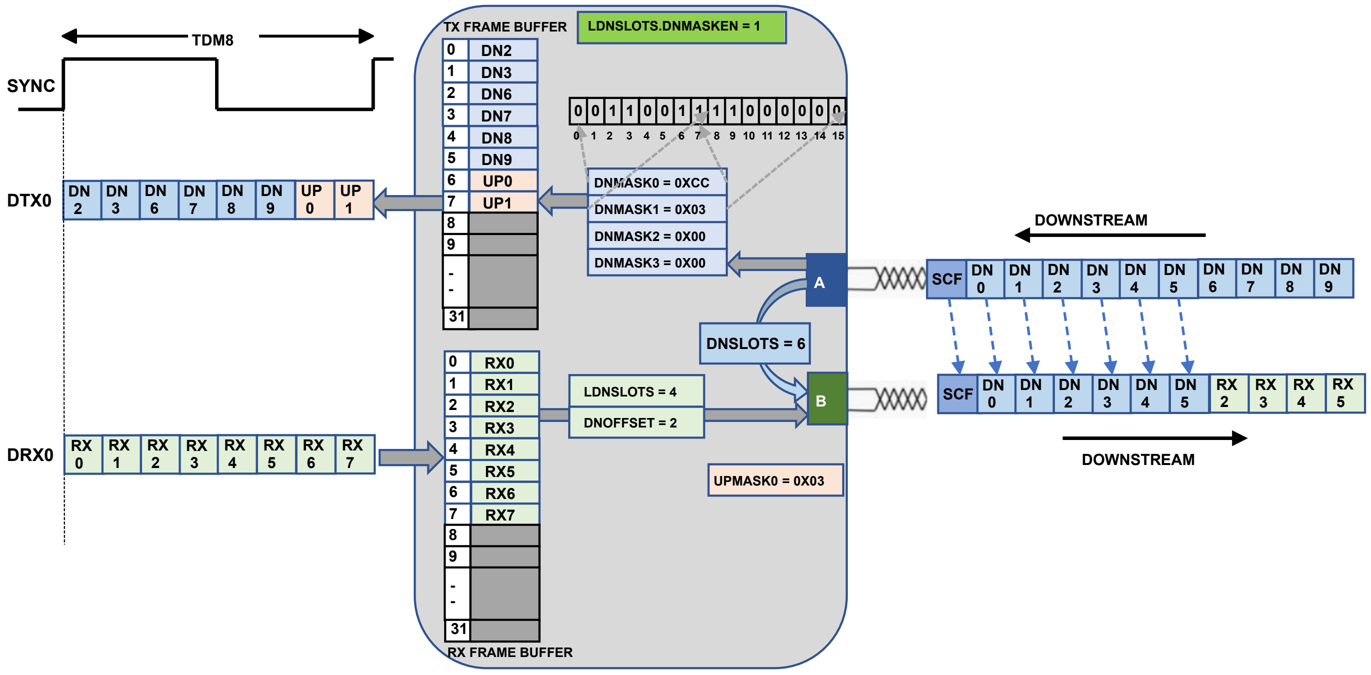


Figure 4-27: Downstream Data Slots

Note the following points in the ***Downstream Data Slots*** figure:

- A2B_DNSLOTS.DNSLOTS = 6 indicates that six downstream slots (slot 0 to slot 5) are passed down from the A-port to the B-port
- A2B_DNMASK0 - A2B_DNMASK3 = [0xCC, 0x03, 0x00, 0x00] indicates that the node consumes downstream slots 2, 3, 6, 7, 8, and 9 arriving on the A-port. The node receives MAX (A2B_DNSLOTS, dnmaskrx) = 9 slots from the A-port.
- The consumed downstream slots are placed in the TX frame buffer from location 0. The consumed upstream slots, if any, are placed after the downstream slots. Other locations of the frame buffer are not used.
- TDM8 mode is used in the example with the pin interleaving option disabled. A total of eight slots (six downstream slots and two upstream slots) are driven on the DTX0 pin.
- A2B_LDNSLOTS.LDNSLOTS = 4 indicates that node contributes four downstream slots on the bus. The data is locally received from the DRX0 pins of the I^2S/TDM/PDM port.
- A2B_DNOFFSET.DNOFFSET = 2 indicates that received TDM channels with an offset of 2 are mapped to the A^2B bus slots (starting from channel 2).
- The node transmits ten downstream slots on the B-port (six slots passed down from the A-port and four slots locally contributed).

Upstream

The subordinate node uses the following registers for upstream transmission:

- Upstream Slots register (A2B_UPSLOTS) – This register defines the number of data slots that the local node passes up from the B-port to the A-port during the upstream portion of the superframe. The number of upstream data slots are passed upstream immediately after the SRF (from slot 0 to slot (A2B_UPSLOTS.UPSLOTS-1)) from the B-port to the A-port of transceiver regardless of whether or not that subordinate transceiver uses the information contained in the slots.
- Local Upstream Slots register (A2B_LUPSLOTS) – This register defines the number of data slots that the transceiver adds during the upstream portion of the superframe. The node begins adding these slots after passing up the number of slots defined in the A2B_UPSLOTS register from the B-port to the A-port.

  The data placed in the upstream data slots comes from the internal RX frame buffer of the transceiver. The buffer is populated through the RX crossbar using the I^{2}S/TDM/PDM port. The ***Upstream Data Slots - RX Crossbar*** figure show the flow of data slots through the RX crossbar.

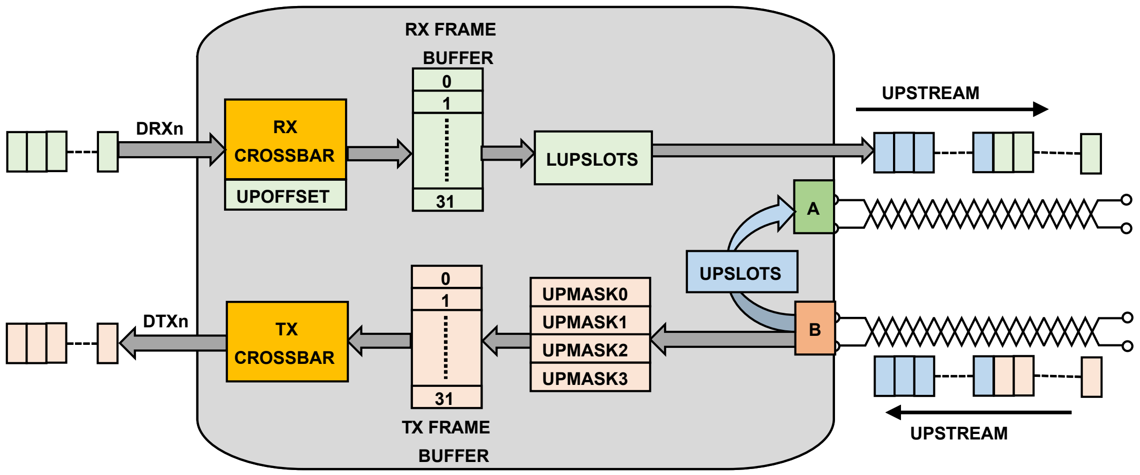


Figure 4-28: Upstream Data Slots - RX Crossbar

- Upstream Data RX Mask 0 through 3 (A2B_UPMASK0 - A2B_UPMASK3) – the subordinate transceiver can selectively receive the upstream A^{2}B bus data slots into the TX frame buffer for output onto its DTXn pins. These four registers provide a mask bit for each upstream data slot (up to 32 upstream slots). These upstream slots are placed in the TX frame buffer after storing the consumed downstream slots.

  The most significant bit that is set in the A2B_UPMASK0 - A2B_UPMASK3 registers defines the number of slots (upmaskrx) that the transceiver must receive in order to appropriately place the enabled slots in the TX frame buffer for I^{2}S/TDM Transmission Latency output to the I^{2}S/TDM port. A subordinate transceiver receives MAX (A2B_UPSLOTS, upmaskrx) upstream data slots on the B-side of the transceiver. It then transmits A2B_UPSLOTS + A2B_LUPSLOTS upstream data slots on the A-side of the transceiver.
- Local Upstream Channel Offset register (A2B_UPOFFSET) – This register is used when the subordinate transceiver is configured to contribute upstream data. Data is placed in the enabled upstream slots starting with the beginning of the RX frame buffer (unless the A2B_UPOFFSET register has been programmed to apply an offset in the RX frame buffer from which it begins populating the enabled upstream slots).

The *Upstream Data Slots* figure provides an example of how upstream data slots are used in a subordinate transceiver after programming the A2B_UPSLOTS, A2B_UPMASK0 - A2B_UPMASK3, A2B_LUPSLOTS and A2B_UPOFFSET registers with default I^2S TX/RX crossbar settings.

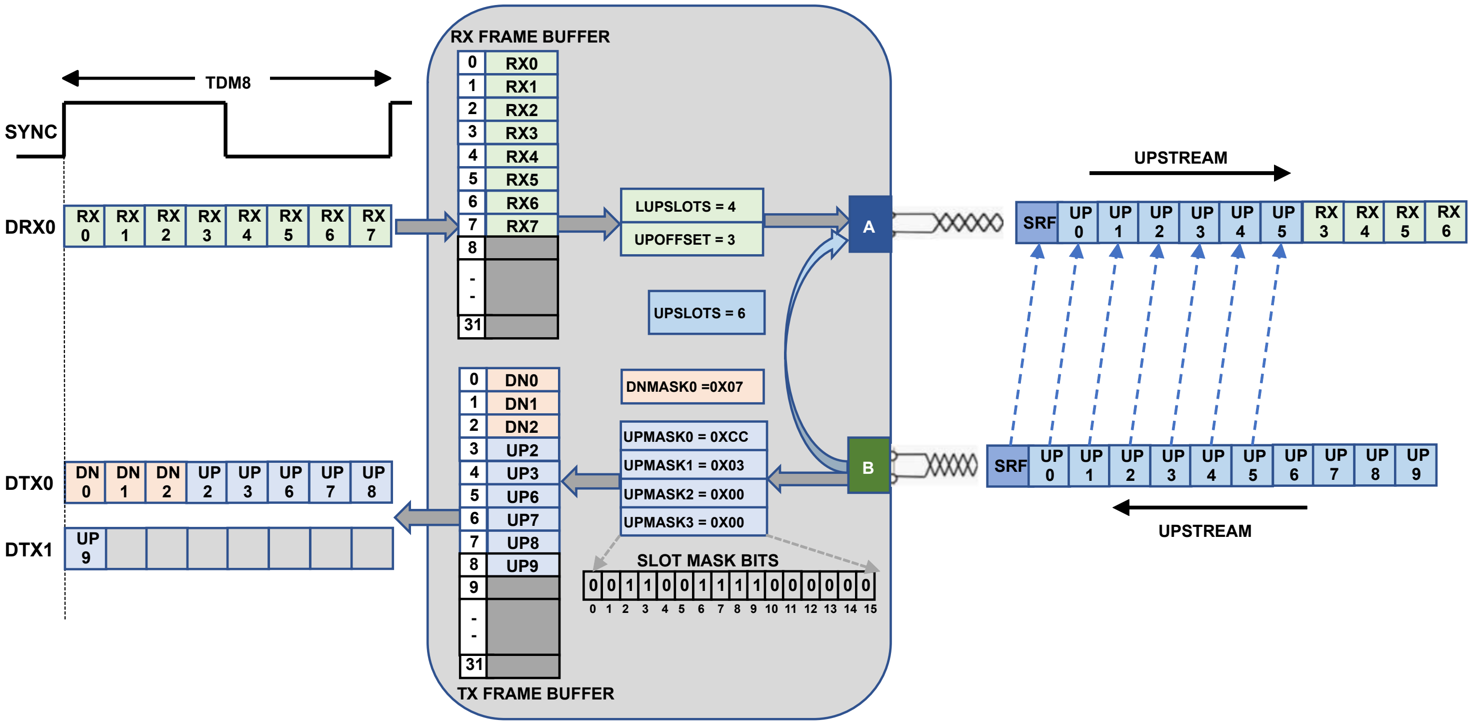


Figure 4-29: Upstream Data Slots

Note the following points in the *Upstream Data Slots* figure:

- A2B_UPSLOTS.UPSLOTS = 6 indicates that six upstream slots (slot 0 to slot 5) are passed up from the B-port to the A-port
- A2B_UPMASK0 - A2B_UPMASK3 = [0xCC, 0x03, 0x00, 0x00] indicates that the node consumes upstream slots 2, 3, 6, 7, 8 and 9 arriving on the B-port. The node receives MAX (A2B_UPSLOTS, upmaskrx) = 9 slots from the B-port.
- The consumed upstream slots are placed in the TX frame buffer after the downstream slots are placed in the first part of the superframe. Other locations in the frame buffer are not used.
- TDM8 mode is used with the pin interleaving option disabled. A total of nine slots (three downstream slots and six upstream slots) are driven on DTX0 and DTX1.
- A2B_LUPSLOTS.LUPSLOTS = 4 indicates that the node contributes four upstream slots on the bus. The data is locally received from the DRX0 pins of the I^2S/TDM/PDM port.
- A2B_UPOFFSET.UPOFFSET = 3 indicates that the received TDM channels with an offset of 3 are mapped to the A^2B bus slots (starting from channel 3).

- The node transmits ten upstream slots on the A-port (six slots are passed up from the B-port and four slots are locally contributed).

To understand slots consumption and contribution at the A^{2}B system level, consider a typical 4-node system, in which all nodes must send two slots to every other node. See the ***Slot Consumption*** figure. In an actual application, some A^{2}B nodes (like AMP) may only consume the slots and do not contribute any slots; some A^{2}B nodes (like microphones) may only contribute slots and do not consume any bus slots.

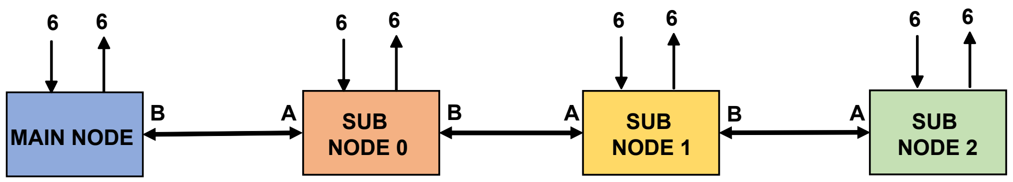


Figure 4-30: Slot Consumption

Refer to the ***Superframe Slot Consumption*** figure. In the figure, the slots are color-coded and named using a source-destination acronym. For example, the M – S2 slot is sent by the main node to subordinate node 2, the S0 – S1 slot is sent by subordinate node 0 to subordinate node 1.

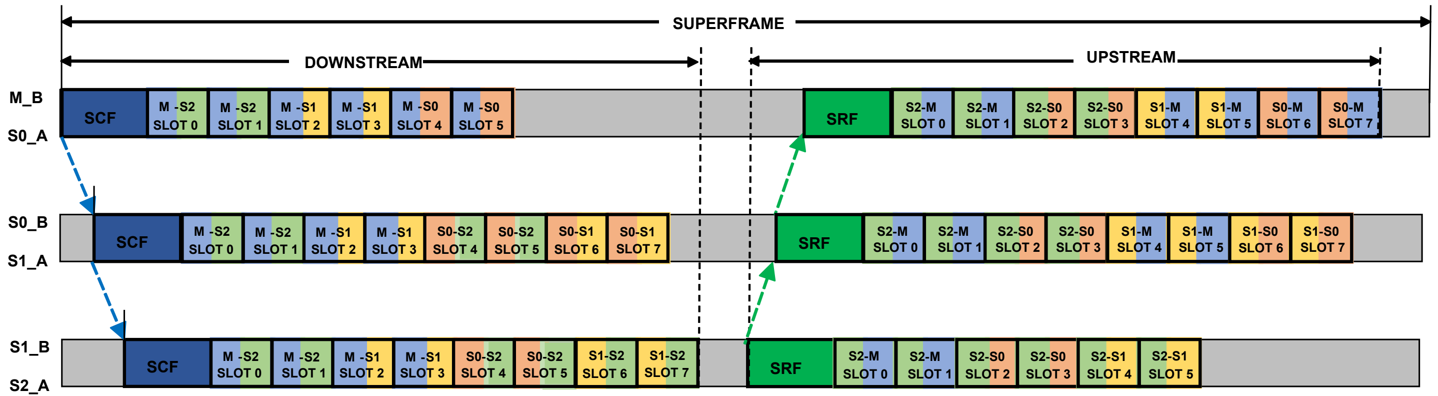


Figure 4-31: Superframe Slot Consumption

The following downstream sequence is shown in the ***Superframe Slot Consumption*** figure:

1. The main node takes six channels (two for each subordinate node) from the DRXn pin of its I^{2}S/TDM interface and puts the channels on the A^{2}B bus after the SCF field. `A2B_DNSLOTS.DNSLOTS` = 6. The slots for the last-in-line subordinate node (subordinate 2) are first to place data slots on the A^{2}B bus, followed by the middle subordinate nodes (subordinate 1) and finally the slots for subordinate 0. This sequence saves A^{2}B bandwidth because closer nodes can consume the last slots from the bus and remove the slots to free up the bus bandwidth. The host processor at main node should accordingly send the channels on DRXn pins.
2. The M – S0 downstream traffic with six slots is received at the A-port of subordinate node 0. Subordinate node 0 consumes the two intended slots (slot 4 and slot 5) and removes the slots from the bus to free up space. The first four slots intended for subordinate 1 and subordinate 2 are passed as-is from the A-port to the B-port.

subordinate node 0 contributes four downstream slots (two each for subordinate node 1 and subordinate node 2). These slots are appended to the slots that the node passed from the A-port to the B-port. The slots for the last-in-line subordinate node (subordinate 2) are placed on the A^{2}B bus first; slots for the closer nodes (subordinate node 1) are placed at the end.

In this example, the SLOT registers settings for subordinate node 0 are:

- `A2B_DNSLOTS.DNSLOTS` = 4 indicates the four slots passed down from the A-port to the B-port
- `A2B_DNMASK0` - `A2B_DNMASK3` = [0x30, 0x00, 0x00, 0x00] indicates the consumption of slot 4 and slot 5 from the A-port
- `A2B_LDNSLOTS.LDNSLOTS`= 0x84 indicates the contributed four downstream slots. `A2B_LDNSLOTS.DNMASKEN` = 1.

Subordinate node 0 passes a total of eight downstream slots on the B-port.

3. The S0 – S1 downstream traffic with eight downstream slots is received at the A-port of subordinate node 1. subordinate node 1 consumes the intended four downstream slots (slot 2, slot 3 from the main node, and slot 6 and slot 7 from subordinate node 0). subordinate node 1 can remove slot 6 and slot 7 which appear at the end of downstream; but, it cannot remove slot 2 or slot 3 because slot 4 and slot 5 are intended for subordinate node 2. So, subordinate node 1 must pass downstream the first six slots from the A-port to the B-port. subordinate node 1 contributes two downstream slots for subordinate node 2. These slots are appended to the slots that the node passed from the A-port to the B-port.

   In this example, the SLOT registers settings for subordinate node 1 are:

   - `A2B_DNSLOTS.DNSLOTS` = 6 indicates the six slots passed down from the A-port to the B-port
   - `A2B_DNMASK0` - `A2B_DNMASK3` = [0xCC, 0x00, 0x00, 0x00] indicates the consumption of slots 2, 3, 6, and 7 from the A-port
   - `A2B_LDNSLOTS.LDNSLOTS` = 0x82 indicates the contributed two downstream slots. `A2B_LDNSLOTS.DNMASKEN` = 1.

   subordinate node 1 passes a total of eight downstream slots on the B-port.

4. The S1 – S2 downstream traffic with eight downstream slots is received at the A-port of subordinate node 2. subordinate node 1 consumes the intended six downstream slots (slot 0, slot 1 from the main node, slot 4 and slot 5 from subordinate node 0, and slot 6 and slot 7 from subordinate node 1). This node is the last-in-line subordinate; it does not pass down any slots on the B-port.

   In this example, the SLOT registers settings for subordinate node 2 are:

   - `A2B_DNSLOTS.DNSLOTS` = 0 indicates that no slot is passed down from the A-port to the B-port
   - `A2B_DNMASK0` - `A2B_DNMASK3` = [0xF3, 0x00, 0x00, 0x00] indicates the consumption of slots 0, 1, 4, 5, 6, and 7 from the A-port
   - `A2B_LDNSLOTS.LDNSLOTS`= 0x80. `A2B_LDNSLOTS.DNMASKEN` = 1

The following upstream sequence is shown in the *Superframe Slot Consumption* figure:

1. subordinate node 2 takes six channels (two for each subordinate node and main node) from the DRXn pin of its I^{2}S/TDM interface. It places the channels on the A^{2}B bus after the SRF field. A2B_LDNSLOTS = 6. The slots for the main node are placed first on the A^{2}B bus, followed by the slots for the middle subordinate nodes (subordinate 0), and finally, the slots for the closer subordinate node (subordinate 1). This sequence saves A^{2}B bandwidth because closer nodes can consume (and remove) the last slots from the bus to free up bandwidth. The I^{2}S/TDM peripheral at the nodes should organize the channels accordingly on the DRXn pins.
2. The S2 – S1 upstream traffic with six up slots is received at the B-port of subordinate node 1. subordinate node 1 consumes the two intended up slots (slot 4 and slot 5) and removes them from the bus to free up the space. The first four slots intended for the main node and subordinate node 0 are passed upstream as-is from the B-port to the A-port.

   subordinate node 1 contributes four upstream slots (two each for the main node and subordinate node 0). These slots are appended to the slots that the node passed up from the B-port to the A-port. The slots for main node are placed first; slots for closer nodes (subordinate node 0) are placed on the A^{2}B bus at the end.

   In this example, the SLOT registers settings for subordinate node 1 are:

   - A2B_UPSLOTS.UPSLOTS = 4 indicates the four slots passed from the B-port to the A-port
   - A2B_UPMASK0 - A2B_UPMASK3 = [0x30, 0x00, 0x00, 0x00] indicates the consumption of slot 4 and slot 5 from the B-port
   - A2B_LUPSLOTS.LUPSLOTS = 0x04 indicates the contribution of four upslots
   - A2B_UPOFFSET register is programmed based on channels on the DRXn pins of the node.

   subordinate node 0 passes a total of eight upstream slots on the A-port.
3. The S1 – S0 upstream traffic with eight up slots is received at the B-port of subordinate node 0. subordinate node 0 consumes the four intended up slots (slot 2 and slot 3 from subordinate node 2, and slot 6 and slot 7 from subordinate node 1). subordinate node 1 can remove slot 6 and slot 7 which appear at the end of the upstream; but, it cannot remove slot 2 or slot 3, because slot 4 and slot 5 are intended for the main node. Therefore, subordinate node 0 passes upstream the first six slots from the B-port to the A-port.

   subordinate node 0 contributes two upstream slots for the main node. These slots are appended to the slots that the node passed up from the B-port to the A-port.

   In this example, the SLOT registers settings for subordinate node 1 are:

   - A2B_UPSLOTS.UPSLOTS = 6 indicates the six slots passed up from the A-port to the B-port
   - A2B_UPMASK0 - A2B_UPMASK3 = [0xCC, 0x00, 0x00, 0x00] indicates the consumption of slots 2, 3, 6, and 7 from the B-port
   - A2B_LUPSLOTS.LUPSLOTS = 0x02 indicates the contribution of two upstream slots
   - A2B_UPOFFSET register is programmed based on the channels on the DRXn pins of the node

subordinate node 0 passes a total of eight upstream slots on the A-port.

4. The S0 – M upstream traffic with eight upslots is received at the B-port of the main node. The main node consumes the slots and places the data on its DTXn pins using the TDM interface. `A2B_UPSLOTS.UPSLOTS` = 8. The main node does not have mask registers to select the required slots. Therefore, all eight slots are received and placed in the TX frame buffer of the node, including two slots sent by subordinate node 2 to subordinate node 0 (which subordinate 0 could not remove from the bus after consuming). The TX crossbar can be used to place the required slots on the DTXn pins and mask unnecessary slots coming out on the DTXn pins of the transceiver.

I^{2}S/TDM Transmission Latency

There is a sample delay incurred for data moving between the A^{2}B bus and the I^{2}S/TDM interfaces. Data is received and transmitted over the I^{2}S/TDM each sample period (typically 48 kHz). This timing relationship between samples is shown in the ***Data Transfer*** figure. The figure shows the data transfer between the main node and a subordinate node. A similar data transfer occurs between any two subordinate nodes because all nodes receive downstream and upstream slots in a superframe.

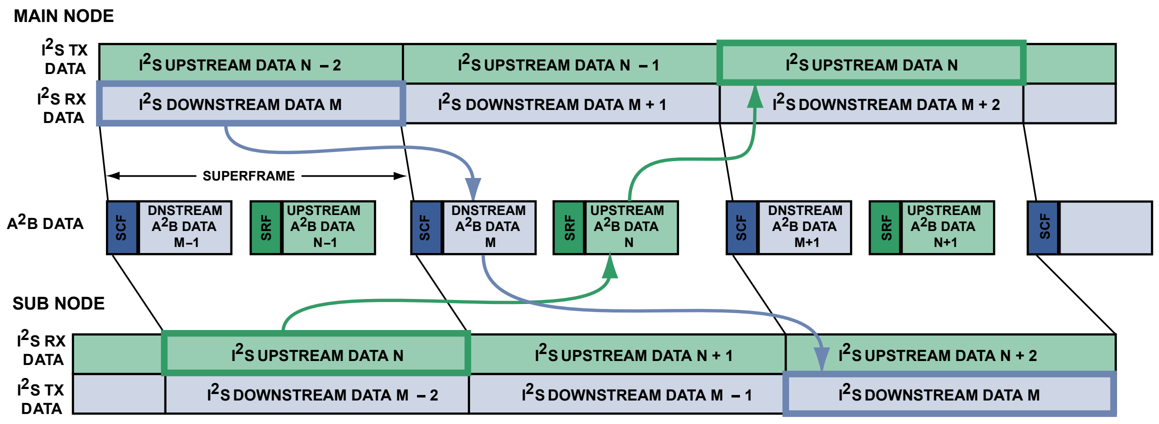


Figure 4-32: Data Transfer

In the ***Data Transfer*** figure, both downstream and upstream samples are named for the frame where they enter the A^{2}B system as follows:

- I^{2}S/TDM data received by the main node transceiver in superframe M creates downstream data M. The data is transmitted over the A^{2}B bus in the next superframe.
- I^{2}S/TDM data received by the subordinate node transceivers in superframe N creates upstream data N. The data is transmitted over the A^{2}B bus in the next superframe.
- Data received from the A^{2}B bus is transmitted on the I^{2}S/TDM interface of an A^{2}B transceiver in the following superframe.
- Data transmitted across the A^{2}B bus (from any node to any node) has two frames of latency plus any internal delay that has accumulated in the transceivers, as well as delays due to wire length. Therefore, overall latency is

slightly over two samples (less than 50 µs at 48 kHz sample periods) from the I^2S/TDM interface in one A^2B transceiver to the I^2S/TDM interface of another A^2B transceiver.

The I^2S/TDM data channels have latency of approximately two superframes, which is less than 50 µs for 48 KHz audio rate. This latency is deterministic and does not vary in any case. Therefore, A^2B systems are suitable for ANC/RNC applications, where microphone data from different places are sent with minimal latency to the host processor for further processing.

Synchronizing Subordinate Nodes

The A^2B main node receives the SYNC signal at the audio rate (44.1 KHz and 48 KHz) from the host processor and locks its PLL based on this signal. The main node sends the synchronization signal at the same audio rate to all the subordinate nodes over the A^2B bus. The subordinate nodes lock the PLL based on this synchronization signal and generate clock signals (BCLK and SYNC) for the interfaced devices. Therefore, all nodes are clock synchronous. However, there is a phase difference (SYNC delay) between the nodes. For a system with minimal cable delay, the subordinate node SYNC relationship with respect to the main node SYNC signal is shown in the ***SYNC Delay*** table. Subordinate 0 SYNC is approximately 13-bit times later than the main SYNC signal. Subordinate N+1 SYNC is approximately 7-bit times later than subordinate *n* SYNC.

1 bus time = 1 / (1024 x f_{M_SYNC})

Table 4-4: SYNC Delay

Node	SYNC Delay (in bus bit times SYSBCLK)
Main	0
Subordinate 0	13
Subordinate 1	20
Subordinate 2	27
…	…
Subordinate n	13+(n 7)

The ***Node Synchronization*** figure shows the SYNC relationship between nodes. Cable latency adds to synchronization time. Depending on the cable length between nodes, 6.5 ns/m is added to the delay.

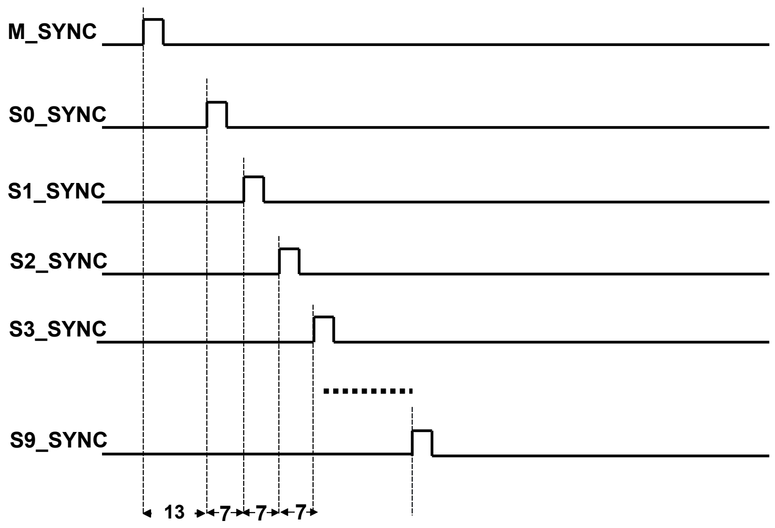


Figure 4-33: Node Synchronization

The SYNC signal is used for framing the TDM channels. It can also used for audio sampling purposes by connected peripherals (such as an audio codec). For a PDM microphone interface, the PDM microphones do not need a SYNC signal. The A^{2}B transceiver tries to frame the PDM stream based on an internal SYNC signal. Some applications require an exact signal sampling point on different subordinate nodes. For example, in ANC/RNC applications, the PDM microphones can be distributed at different places that are connected with multiple A^{2}B nodes. In this case, it may be necessary to sample the PDM microphones at the same time without any delay.

A^{2}B subordinate nodes can be configured to align the SYNC signal on all the nodes by individually compensating for their propagation delay in the `A2B_SYNCOFFSET` register. This configuration allows the exact same sample time on different A^{2}B nodes. Writing a non-zero value to this register adjusts the A^{2}B bus clock ($f_{SYSBCLK}$) cycle on which the SYNC pin indicates the start of an audio frame for the particular subordinate transceiver. The register has 8-bit signed two's complement representation of the integer number of SYSBCLK cycles between the superframe start time and the active edge of the SYNC signal. The `A2B_SYNCOFFSET` register of each subordinate node must be configured according to the SYNC delay specified in the ***SYNC Delay*** table and considering the cable delay.

The maximum value that can be programmed into the `A2B_SYNCOFFSET` register defines a SYNC signal to occur 104 SYSBCLK cycles before the start of the superframe (-104 = 0x98). This value is only valid for the last-in-line subordinate node (furthest away from the main in a fully populated A^{2}B network topology with `A2B_NODEADR.NODE` = 0x09). For any subordinate node *n* that is closer to the main node, the valid ranges supporting a predictable transfer of I^{2}S/TDM data to A^{2}B slots are a function of the location of subordinate node *n* in the network. The range is governed by the formula:

(-32 8n) ≤ `A2B_SYNCOFFSET` ≤ 0

The ***Supported SYNC Offset*** table summarizes the valid settings for the `A2B_SYNCOFFSET` register for any given subordinate node in SYSBCLK cycles (Offset Range).

Table 4-5: Supported SYNC Offset

Subordinate Node n	Offset Range	A2B_SYNCOFFSET Range
0	-32 to 0	0xE0 to 0x00
1	-40 to 0	0xD8 to 0x00
2	-48 to 0	0xD0 to 0x00
3	-56 to 0	0xC8 to 0x00
4	-64 to 0	0xC0 to 0x00
5	-72 to 0	0xB8 to 0x00
6	-80 to 0	0xB0 to 0x00
7	-88 to 0	0xA8 to 0x00
8	-96 to 0	0xA0 to 0x00
9	-104 to 0	0x98 to 0x00

5 A^{2}B Operation and Configuration

The A^{2}B bus is programmable at a high-level. It can address many use cases. A^{2}B systems are easy to configure, based on knowledge of the system, nodes, and peripherals. The exact system configuration can be gained by collecting information individually from each subordinate node. As an example, the same A^{2}B module can be supplied by different vendors, with each of the modules having unique register programming requirements. One module can use TDM4 as an audio interface, while another one uses TDM8. One module can provide two upstream channels, while another can provide three upstream channels, all with the host not having prior knowledge of how many nodes are connected.

IMPORTANT: Ensure that the register programming results in a valid system configuration.

Analog Devices provides free SigmaStudio^{TM} (http://www.analog.com/SigmaStudio) tools featuring an intuitive graphical user interface to architect, configure, and set up the A^{2}B bus. The tools also generate driver code for embedded software.

I^{2}C Interface

The I^{2}C interface is an important block of the A^{2}B transceiver. The transceiver registers are accessible and can be programmed for node operations using the I^{2}C interface. Typically, the host processor on the main node configures and controls the whole A^{2}B system using the I^{2}C port. The host processor can access (read/write) the registers of the main node, the subordinate nodes, and I^{2}C peripherals connected to subordinate nodes. No intelligence or processor stack is needed at the subordinate node. A local processor such as an AMP node or eCall unit can be connected at the subordinate node. The local processor can access the node registers using a local I^{2}C bus, when required.

The ***I^{2}C Interface*** figure shows the I^{2}C interfaces in a typical A^{2}B system.

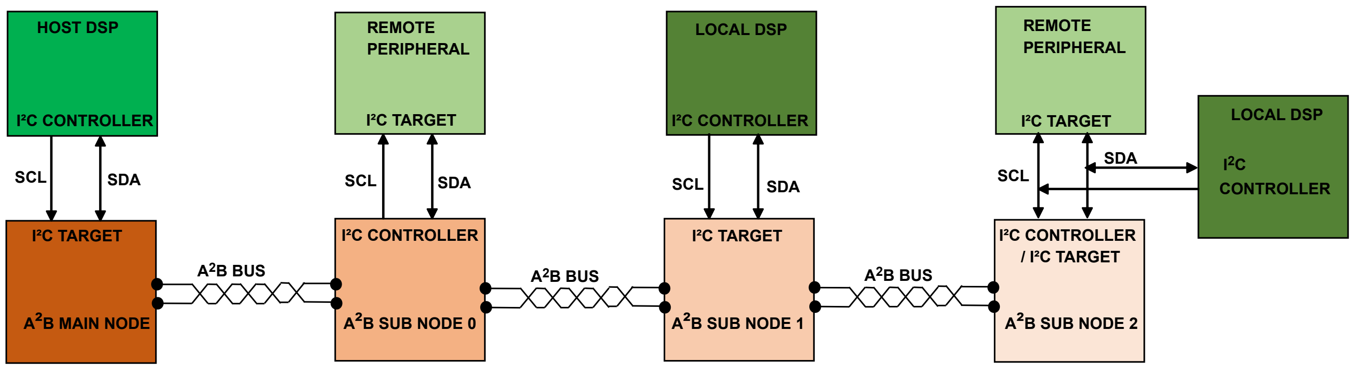


Figure 5-1: I^{2}C Interface

The I^{2}C port of the A^{2}B controller node is always the I^{2}C target; it accepts commands from the host processor.

The host can initiate the following accesses (read/write):

- A^{2}B main node register access
- A^{2}B subordinate node register access – the access goes over A^{2}B bus between the nodes
- Register access of a remote peripheral connected to any A^{2}B subordinate node. The access goes over A^{2}B bus between the nodes and is replicated on the local I^{2}C port of a subordinate node. This access is known as an *I^{2}C over distance* access.

The I^{2}C port of an A^{2}B subordinate node supports both I^{2}C controller and I^{2}C target operations. It also supports a multi controller I^{2}C environment. It becomes the I^{2}C main when the host processor accesses the registers of a remote peripheral (connected to the subordinate node). It becomes the I^{2}C target when a local processor requires access. The local processor can access the registers of a connected subordinate node transceiver using the I^{2}C bus. The A^{2}B transceiver supports single-byte register accesses as well as burst type accesses (read/write). The supported I^{2}C bit rates are: 100 kbps, 400 kbps, or 1 Mbps.

NOTE: The host processor that is connected to the main A^{2}B transceiver must support I^{2}C clock stretching. The subordinate transceiver supports clock stretching when in I^{2}C controller mode.

Transceiver I^{2}C Device Address Selection

Like other I^{2}C devices, the A^{2}B transceiver has a 7-bit I^{2}C device address (excluding the R/W bit) that the locally connected processor uses to access the transceiver registers. This device address is established by the logic levels on the ADR1 and ADR2 pins at the power-on RESET state. All nodes latch their ADR1 and ADR2 pins to set their own device address, known as BASE_ADDR. They can be independent of each other. The connected processor use the device address to access the registers of the local node. For example, the host processor can use this device address to access the main node registers. If the processor is connected to a subordinate node using the I^{2}C port, it can access the registers of the local subordinate node using the BASE_ADDR of the subordinate node. The main node

BASE_ADDR and subordinate node BASE_ADDR can be different. The main transceiver also recognizes another I^{2}C device address known as BUS_ADDR. This device address is used for addressing the subordinate node and the remote peripheral. The BUS_ADDR differs from the BASE_ADDR of the main node by the least significant bit only. It does not depend on the BASE_ADDR of any subordinate node.

The ***Device Address Selection*** figure shows how registers are accessed using BASE_ADDR and BUS_ADDR.

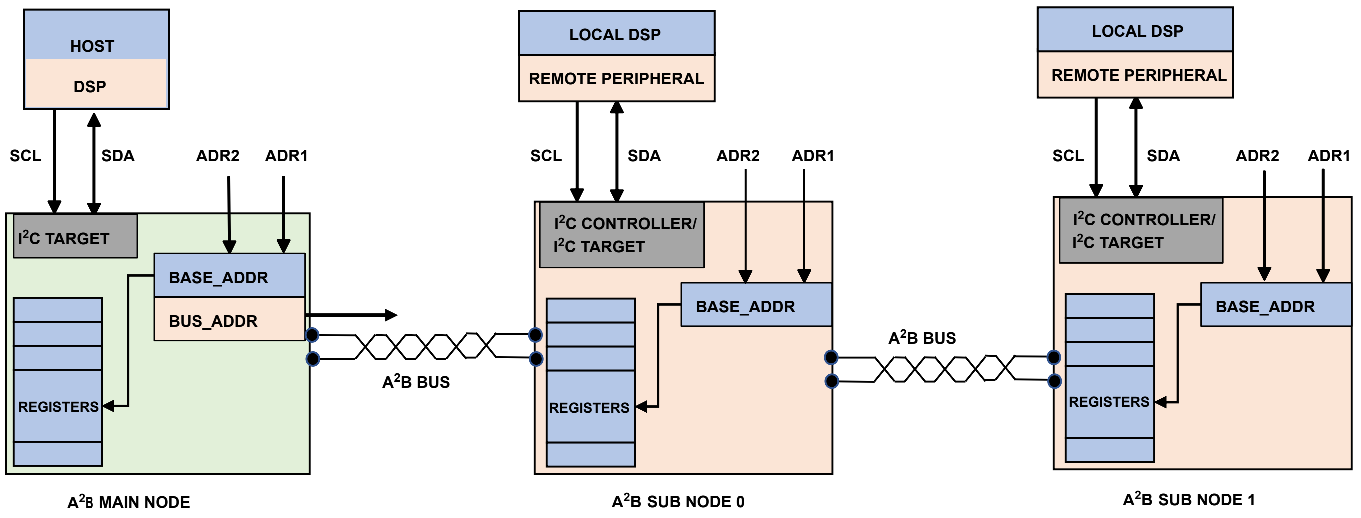


Figure 5-2: Device Address Selection

The ***Device Addresses*** tables shows the state of the ADR1 and ADR2 pins during power-up and the corresponding BASE_ADDR and BUS_ADDR.

Table 5-1: I^{2}C Device Address

Bit Number	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	1	1	0	1	ADR2	ADR1	BASE_ADDR = 0 BUS_ADDR =1

Table 5-2: I^{2}C Device Address

ADR2/ADR1	BASE_ADDR	BUS_ADDR
00	0x68	0x69
01	0x6A	0x6B
10	0x6C	0x6D
11	0x6E	0x6F

A subordinate configured transceiver does not recognize the access with BUS_ADDR. The host processor connected to the main node can access the registers of any subordinate node or remote peripheral connected to a subordinate node. However, the local processor connected to subordinate node can only access the registers of a local node; it cannot access registers of other nodes on the A^{2}B bus.

NOTE: In the Power-up state, A^{2}B transceivers have very reduced amount of logic active, including remote I^{2}C management. The node in Power-up state acknowledges accesses with both BASE_ADDR and BUS_ADDR. The read accesses to the node with BUS_ADDR returns the value 0x00 without any NAK to access or I2CERR.

Transceiver I^{2}C Accesses

The A^{2}B transceiver supports the following I^{2}C accesses:

- Main Node Access
- Host Processor to Subordinate Node Access
- Remote Peripheral Access
- Local Processor to Subordinate Node Access

Main Node Access

Refer to the ***Main Node Access*** figure. The host processor can access the main node registers directly using the I^{2}C bus with the BASE_ADDR of the transceiver. The main node I^{2}C interface is always an I^{2}C target; the host processor can access (read/write) a single register or multiple sequential registers in burst mode. In burst access mode, the transceiver automatically increments the register address pointer after each data byte. Therefore, sequential data registers can be accessed without reprogramming the address.

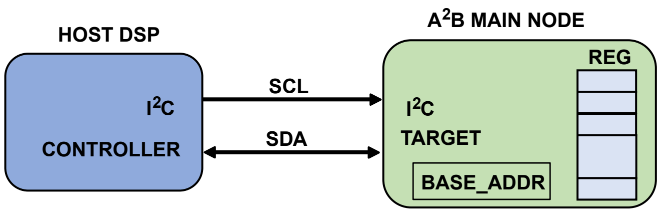


Figure 5-3: Main Node Access

The ***Main Node Access - Bit Sequence*** figure shows the bit sequence for different types of accesses to the main node registers.

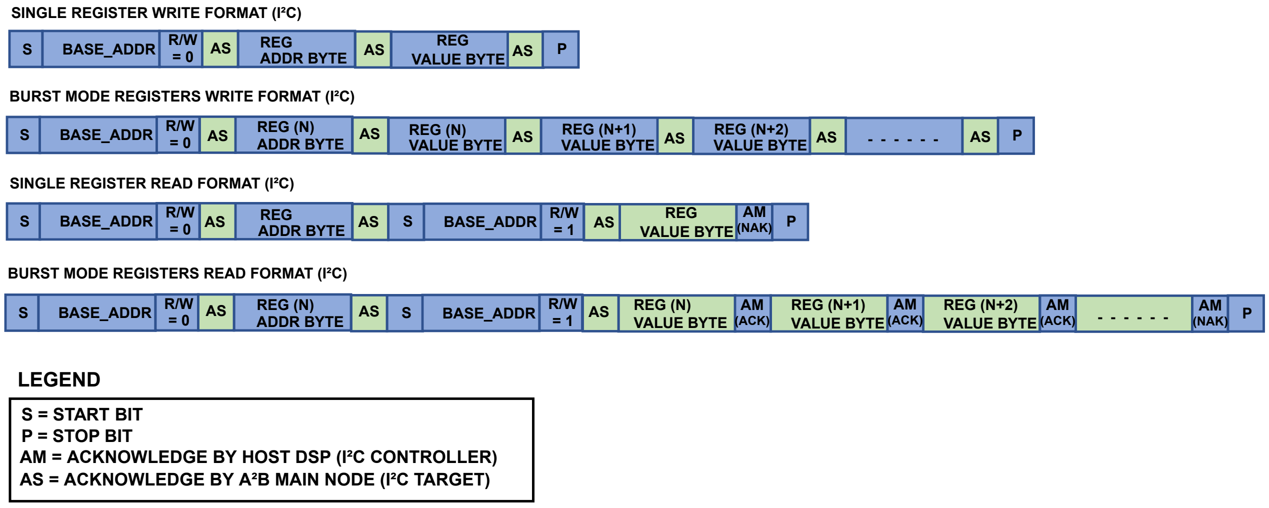

SINGLE REGISTER WRITE FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | REG VALUE BYTE | AS | P
BURST MODE REGISTERS WRITE FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | REG (N) VALUE BYTE | AS | REG (N+1) VALUE BYTE | AS | REG (N+2) VALUE BYTE | AS | - - - - - - | AS | P
SINGLE REGISTER READ FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | S | BASE_ADDR | R/W = 1 | AS | REG VALUE BYTE | AM (NAK) | P
BURST MODE REGISTERS READ FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | S | BASE_ADDR | R/W = 1 | AS | REG (N) VALUE BYTE | AM (ACK) | REG (N+1) VALUE BYTE | AM (ACK) | REG (N+2) VALUE BYTE | AM (ACK) | - - - - - - | AM (NAK) | P
LEGEND
S = START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY HOST DSP (I²C CONTROLLER)
AS = ACKNOWLEDGE BY A²B MAIN NODE (I²C TARGET)

Figure 5-4: Main Node Access - Bit Sequence

Some main node registers (for example, `A2B_SLOTFMT`, `A2B_DATCTL`, and `A2B_I2SRRATE`) are auto-broadcast types. When the host processor writes to these registers, the configuration is automatically broadcast to all discovered subordinate nodes over the A^{2}B bus (using the SCF field). The main node stretches the I^{2}C bus until the access completes and then provides the ACK bit. The main node aborts the access when it does not complete within superframes. NAK is provided after the timeout. The host processor connected to the main A^{2}B transceiver must support I^{2}C clock stretching.

NOTE: The I^{2}C interface of the main node acts as the I^{2}C subordinate node. It supports a clock rate up to 1 MHz. It is not required to set the `A2B_I2CCFG.DATARATE` field for the expected I^{2}C frequency. This bit field applies to the subordinate node only when the subordinate node becomes the I^{2}C controller (in order to drive out the clock).

Host Processor to Subordinate Node Access

The host processor can access the registers of any subordinate node on the bus using a combination of the I^{2}C bus and the A^{2}B bus. It initiates the I^{2}C access locally with the A^{2}B main node using the BUS_ADDR of the transceiver. The A^{2}B main node relays this access to the targeted subordinate node over the A^{2}B bus (using the SCF fields of the superframe). For a register read command, the subordinate node responds with the register contents (using SRF field of the superframe). The main node provides the data to the ongoing I^{2}C access.

The ***Subordinate Node Access*** figure shows a host processor to subordinate node access.

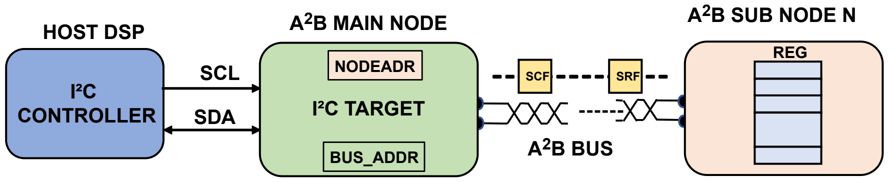


Figure 5-5: Subordinate Node Access

The *Subordinate Node Accesses - Bit Sequence* figure shows the bit sequence for different types of accesses to the subordinate node registers.

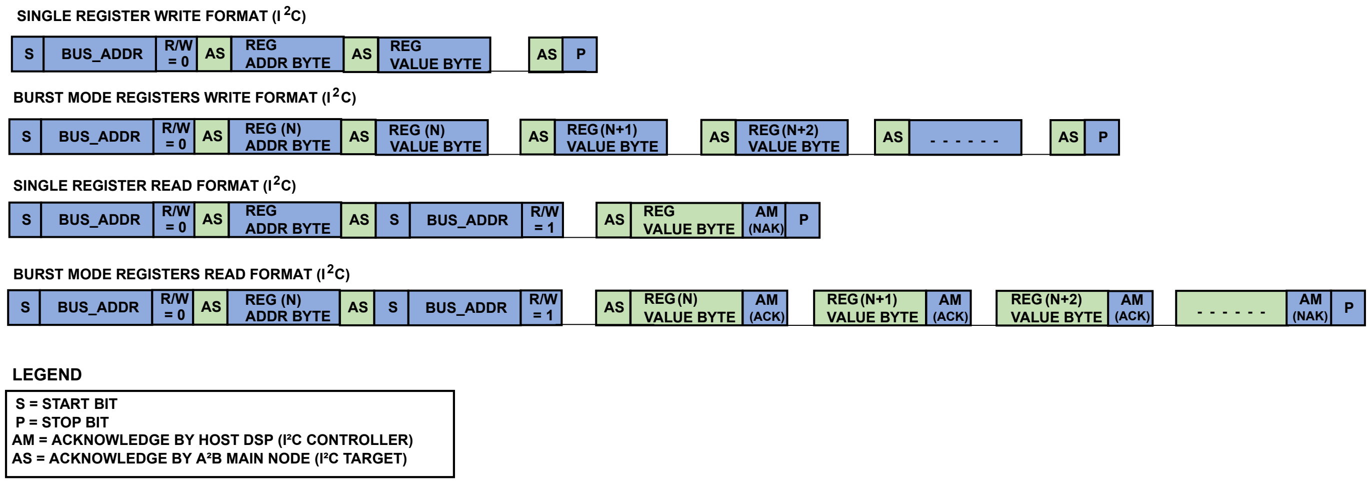

SINGLE REGISTER WRITE FORMAT (I²C)
S | BUS_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | REG VALUE BYTE | AS | P
BURST MODE REGISTERS WRITE FORMAT (I²C)
S | BUS_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | REG (N) VALUE BYTE | AS | REG (N+1) VALUE BYTE | AS | REG (N+2) VALUE BYTE | AS | - - - - - - | AS | P
SINGLE REGISTER READ FORMAT (I²C)
S | BUS_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | S | BUS_ADDR | R/W = 1 | AS | REG VALUE BYTE | AM (NAK) | P
BURST MODE REGISTERS READ FORMAT (I²C)
S | BUS_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | S | BUS_ADDR | R/W = 1 | AS | REG (N) VALUE BYTE | AM (ACK) | REG (N+1) VALUE BYTE | AM (ACK) | REG (N+2) VALUE BYTE | AM (ACK) | - - - - - - | AM (NAK) | P
LEGEND
S = START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY HOST DSP (I²C CONTROLLER)
AS = ACKNOWLEDGE BY A²B MAIN NODE (I²C TARGET)

Figure 5-6: Subordinate Node Accesses - Bit Sequence

The host processor can access (read/write) a single register of a subordinate transceiver or multiple sequential registers in burst mode. In burst access mode, the subordinate transceiver automatically increments the register address pointer after each data byte. Therefore, sequential data registers can be accessed without reprogramming the address.

When the A^{2}B main node receives enough information about the I^{2}C access, it stretches the I^{2}C bus with the host processor as shown in the *Subordinate Node Accesses* figure. The main node internally communicates with the targeted subordinate node to complete the access (for a write access) or get the required register value data (for a read access). SCF and SRF fields of the superframe are used for internal communication between the main node and subordinate nodes. The I^{2}C bus is released after the successful completion of the internal transaction with the subordinate node or after a 30 superframe timeout (when the subordinate node does not respond to the internal request).

The host processor connected to the main A^{2}B transceiver must support I^{2}C clock stretching. The clock stretching latency results from switching the I^{2}C access to and from A^{2}B bus access. The clock stretch time depends on the superframe timing with respect to the I^{2}C access. If the main node receives the required I^{2}C access details before start of SCF field of the superframe, it communicates with subordinate node immediately in the same SCF field and waits for the node response in the SRF. In this case, there is less latency. But, if the main node receives the required I^{2}C access details after the start of the superframe, it communicates with the subordinate node in the next superframe. The latency also depends on bus availability for the access.

The `A2B_I2CCFG.EACK` bit determines the behavior of clock stretching and acknowledge time. When `A2B_I2CCFG.EACK` is cleared (=0) (default), the I^{2}C transactions are clock-stretched until they are complete in the system; the I^{2}C interface can generate a correct ACK/NACK signal. When the `A2B_I2CCFG.EACK` bit is set (=1), the I^{2}C interface provides early acknowledge signals to writes addressed to a subordinate node before the write has completed on the A^{2}B bus. If there is an error (for example, a timeout or address error), the I2CERR error is raised. However, the error may be raised after the failed I^{2}C transaction completes. Therefore, the host processor must verify the previous access when this error occurs.

The ***Subordinate Access Latency*** table provides the typical latency number for accessing subordinate node registers. These numbers are on the slow side of the average, based on simulation.

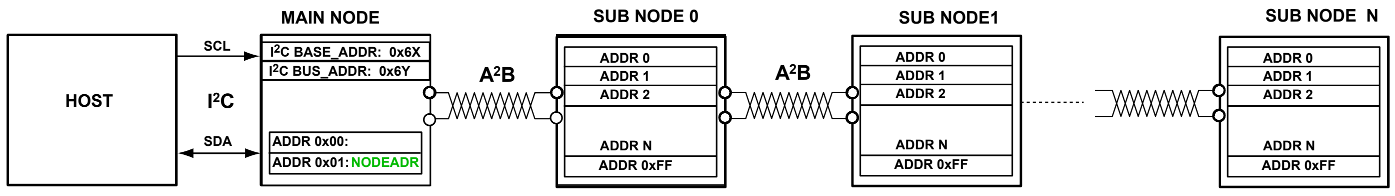
Table 5-3: Subordinate Access Latency

I^{2}C Access	Number of Bytes	**`A2B_I2CCFG.EACK`**	I^{2}C Frequency (KHz)	Bus Latency (us)
Write	N	0	Any	N x 22
Read	N	0	Any	N x 22
Write	>1	1	400	2
Write	N	1	100	0

NOTE: The host accesses the registers of the subordinate node transceiver using the A^{2}B bus. This access is independent of the I^{2}C path that the local processor uses for accessing the node registers. When a write contention occurs, both writes complete. However, the order in which the write operations complete is unpredictable.

There can be multiple subordinate nodes in the A^{2}B chain. To address a particular subordinate node in the chain, the `A2B_NODEADR` register of the main node must be configured appropriately before initiating the access with BUS_ADDR. The `A2B_NODEADR.NODE` bit field selects a subordinate node using its internal address. Addresses are assigned based on the position in the A^{2}B topology, starting with address 0 for the node connected directly to the main node.

For example, if the host needs to access a register of subordinate node 1, the `A2B_NODEADR` register of the main node must be configured first to select node 1 (`A2B_NODEADR.NODE` = 1). Then, the host processor can access the subordinate node 1 registers using BUS_ADDR as the device address. The `A2B_NODEADR.PERI` bit must be 0 for subordinate transceiver register accesses.



Code Example: Subordinate 0 I^{2}C Access

```
// Set sub node number in NODEADR register of main node
<I2C ADDR: BASE_ADDR >R/W=0 <ADDR:0x01> <PERI=0, NODE=0>
// Read or write directly to or from sub 0 registers
<I2C ADDR: BUS_ADDR >R/W <ADDR> <Data>
```

Code Example: Subordinate n I²C Access

```
// Set sub node number in NODEADR register of main node
<I2C ADDR: BASE_ADDR >R/W=0 <ADDR:0x01> <PERI=0, NODE=n>
// Read or write directly to or from sub node n registers
<I2C ADDR: BUS_ADDR >R/W <ADDR> <Data>
```

Once the A2B_NODEADR register is set, the subsequent I^{2}C accesses with BUS_ADDR go to the same subordinate transceiver. The accesses continue until the A2B_NODEADR register is changed. Therefore, if multiple registers of a subordinate node must be accessed, it is not required to set the A2B_NODEADR register before each access. However, if there are many context switches in the software, it is important to configure and confirm the node address before each subordinate node register access. Otherwise, there is a risk of addressing the wrong node.

The A2B_NODEADR.BRCST bit provides a broadcast mode option for register writes. When set, subsequent I^{2}C write accesses initiated with BUS_ADDR are targeted to all the nodes (including the main node).

NOTE: I^{2}C accesses with BASE_ADDR are not broadcast. The broadcast bit does not affect I^{2}C read accesses. Accesses to remote peripherals cannot be broadcast. Therefore, the A2B_NODEADR.PERI bit must be cleared (= 0) when the A2B_NODEADR.BRCST bit is set (=1). The value of the A2B_NODEADR.NODE field is not used when the A2B_NODEADR.BRCST bit is set.

Remote Peripheral Access

The host processor can access the registers of a remote peripheral connected to the I^{2}C port of any subordinate node. It can access (read/write) a single register of a remote peripheral or multiple sequential registers in burst mode. The I^{2}C access goes through the transceivers and over the A^{2}B bus; the access is known as *I^{2}C over distance* access.

The ***Remote Peripheral Access*** figure shows a typical remote peripheral access.

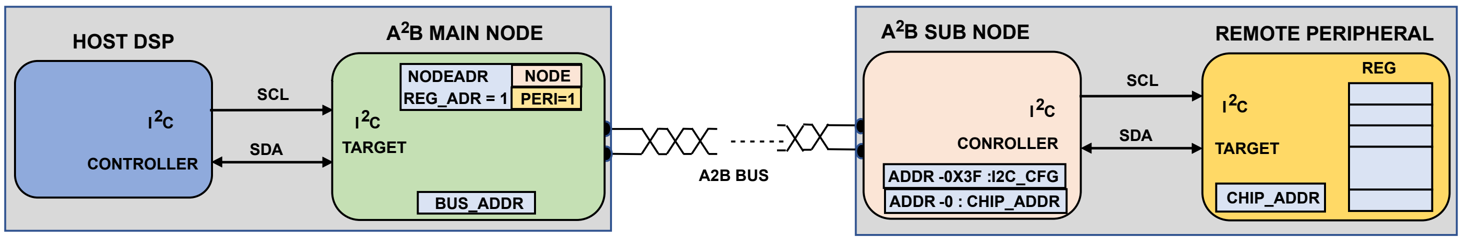


Figure 5-7: Remote Peripheral Access

The different stages of access are:

- The host processor initiates the I^{2}C access locally with the A^{2}B main node using the BUS_ADDR of the transceiver.
- The A^{2}B main node relays the access to the subordinate node transceiver (connected to the remote peripheral) over the A^{2}B bus (using the SCF fields of the superframe).
- Once the subordinate node receives the access details, it becomes the I^{2}C controller, replicates the I^{2}C access on its local I^{2}C port, and communicates with the targeted remote peripheral.
- For a register read command, the subordinate node responds back to the A^{2}B main node with the received data from remote peripheral. The data is sent to the A^{2}B main node (using the SRF field of the superframe). The main node provides this data to host processor in the ongoing I^{2}C access.

When the subordinate node becomes the I^{2}C controller to replicate the I^{2}C access locally, it generates an I^{2}C clock of 100 KHz, 400 KHz, or 1 MHz clock. The clock comes from the internal PLL which is based on the `A2B_I2CCFG.DATARATE` field. It is also required to set the `A2B_I2CCFG.FRAMERATE` field based on the superframe rate. The subordinate node uses the rate to apply appropriate dividers to the PLL clock to generate the I^{2}C clock. These fields do not apply in the main node.

Before initiating the read/write access to a remote peripheral, it is important to correctly set the `A2B_CHIP` register of the subordinate node and the `A2B_NODEADR` register of the main node. The host processor initiates the remote peripheral accesses with BUS_ADDR, but the subordinate node must replicate this access with the I^{2}C device address of the remote peripheral. The subordinate node uses the address specified in its `A2B_CHIP` register (ADDR: 0x00) to initiate the local access. Therefore, the host processor must set this register with the I^{2}C device address of the targeted remote peripheral, before accessing it.

The host processor initiates a remote peripheral access with the BUS_ADDR similar to the subordinate node register access. For the main node, the `A2B_NODEADR.PERI` bit is used to differentiate whether the access received with BUS_ADDR is targeted to the subordinate transceiver or its remote peripheral. When `A2B_NODEADR.PERI` =1 and the main node receives the I^{2}C access with BUS_ADDR, the peripheral connected to the subordinate transceiver (as defined in `A2B_NODEADR.NODE` field) is accesses. When `A2B_NODEADR.PERI` =0, the accesses with BUS_ADDR are targeted to the subordinate node transceiver registers. Before initiating the remote peripheral access, it is important to set the `A2B_NODEADR` register of main node correctly. The `A2B_NODEADR.PERI` bit must be set (= 1); the `A2B_NODEADR.NODE` field must be configured to the subordinate node with the connected peripheral. The subordinate node IDs are assigned based on the position in the A^{2}B topology, starting with 0 for the node connected directly to the main node. The access to remote peripherals cannot be broadcast, so the `A2B_NODEADR.BRCST` bit must be 0 when the `A2B_NODEADR.PERI` bit is set.

Once the `A2B_CHIP` and `A2B_NODEADR` registers are set, the subsequent I^{2}C accesses with BUS_ADDR go to same remote peripheral until those registers are changed. Therefore, if multiple registers of a remote peripheral must be accessed, it is not required to set the `A2B_CHIP` and `A2B_NODEADR` registers prior to each peripheral access. But, if there are many context switches in the software, it is important to configure/confirm at least the `A2B_NODEADR` register before the remote peripheral register access.

Refer to the following *Code Example*. If the host must access a remote peripheral connected to subordinate node 1, the `A2B_CHIP` register of subordinate node 1 must be configured with the I^{2}C device address of the remote peripheral. Then, configure the `A2B_NODEADR` register of the main node to set the `A2B_NODEADR.PERI` bit and select node 1 in the `A2B_NODEADR.NODE` field. The host processor can access the subordinate node 1 registers using BUS_ADDR as device address.

Code Example: Peripheral of Subordinate 0 I^{2}C Access

1. Set the `A2B_CHIP` register of the subordinate node to which the targeted remote peripheral is connected.

```
// Set sub node number in NODEADDR register of main node
<I2C ADDR: BASE_ADDR >R/W=0 <ADDR:0x01> <PERI=0, NODE=1>
// Set the I2C device address of remote peripheral into CHIP register of sub
node
<I2C ADDR: BUS_ADDR >R/W=0 <ADDR:0x00> <CHIP_ADDR>
```

2. Set the `A2B_NODEADR` register of the main node.

```
// Set the PERI bit, sub node number in NODEADDR register of main node
<I2C ADDR: BASE_ADDR >R/W=0 <ADDR:0x01> <PERI=1, NODE=1>
```

3. Initiate the read or write access from the remote peripheral.

```
 <I2C ADDR: BUS_ADDR >R/W=0 <ADDR> <Data>
```

The following sections describe read and write accesses to a remote transceiver. It is assumed that the `A2B_CHIP` and `A2B_NODEADR` registers are correctly configured prior to the accesses.

Remote Peripheral I^{2}C Write Access

The ***Remote I^{2}C Write Access*** figure shows a typical remote peripheral I^{2}C write access. The access is shown in terms of bytes instead of register address bytes and register value data bytes because the number of address bytes and number of value data bytes depend on the remote peripheral.

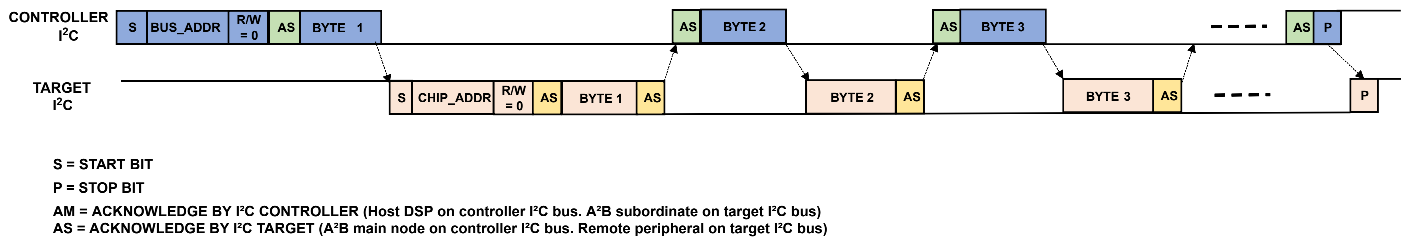

CONTROLLER I²C
S
BUS_ADDR
R/W = 0
AS
BYTE 1
AS
BYTE 2
AS
BYTE 3
AS
P
TARGET I²C
S
CHIP_ADDR
R/W = 0
AS
BYTE 1
AS
BYTE 2
AS
BYTE 3
AS
P
S = START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY I²C CONTROLLER (Host DSP on controller I²C bus. A²B subordinate on target I²C bus)
AS = ACKNOWLEDGE BY I²C TARGET (A²B main node on controller I²C bus. Remote peripheral on target I²C bus)

Figure 5-8: Remote I^{2}C Write Access

Consider the following cases:

Case 1 – The peripheral register address is 1 byte and each register is 8 bits wide. In this case, BYTE 1 represents the address byte and BYTE 2 represents the register values to be written. If the peripheral supports a register address auto-increment feature and subsequent registers must be programmed, then burst accesses can be initiated. In this case, BYTE 3 and onward represent values to be written into the next registers.

Case 2 – The peripheral register address is 2 bytes long and each register is 4 bytes long. In this case, BYTE 1 and BYTE 2 represent the address bytes. BYTE 3 to BYTE 6 represent register value to be written. The host processor must provide these bytes based on the byte order requirement (LSB byte first or MSB byte first). For the host processor, the access is as if it is directly accessing the remote peripheral register, except that the access must be initiated with BUS_ADDR as the I^{2}C device address instead of the actual remote peripheral I^{2}C address.

The flow for an I^{2}C write access is:

1. The host processor initiates the access with BUS_ADDR and R/W bit = 0. The A^{2}B main node provides the ACK for this device address.
2. The host follows with BYTE 1 data and expects ACK for this byte.
3. Before providing the acknowledgment to this byte, the main node stretches the I^{2}C bus with the host processor and internally relays the information to the targeted subordinate node (connected to the remote peripheral).
4. The subordinate node becomes the I^{2}C controller and checks whether the I^{2}C bus is free. If the local I^{2}C bus of the subordinate node is not free (due to a multi-controller environment), the subordinate node aborts the access after superframes. It reports the error to the main node. The main node provides the NAK to the host indicating that the access must be attempted again.
5. If the I^{2}C bus is free, the subordinate node initiates the access with the peripheral I^{2}C device address. The subordinate transceiver uses the value of its `A2B_CHIP` register as the I^{2}C device address for the local access.
6. When CHIP_ADDR and R/W bit = 0, the subordinate node drives BYTE 1 and waits for acknowledgment.
7. When the remote peripheral provides the ACK, the subordinate node forwards the acknowledgment to the A^{2}B main node internally (using the SRF field of the superframe). The subordinate node stretches its I^{2}C bus until further instruction from the main node.
8. The main node releases the I^{2}C bus (previously stretched) and provides the acknowledgment on the I^{2}C bus.
9. The host processor drives out the next byte (BYTE 2) on the I^{2}C bus and waits for acknowledgment.
10. Before providing the acknowledgment to this byte, the main node stretches the I^{2}C bus and internally relays the new byte to the subordinate node.
11. The subordinate node releases the I^{2}C bus (previously stretched) and drives out the byte received. It waits for acknowledgment from the connected peripheral.
12. The subordinate node receives the ACK from the remote peripheral and forwards it to the A^{2}B main node internally (using the SRF field of the superframe). The subordinate node stretches its I^{2}C bus until further instruction from the main node.
13. The main node receives the ACK. It releases the I^{2}C bus (which was stretched) and provides the acknowledgment on the I^{2}C bus.
14. Steps 9 to 13 repeat until the host processor has finished sending all bytes.

15. The host processor provides the STOP bit on the I^2C bus.
16. Once the main node receives the STOP bit for the ongoing I^2C access, it communicates the information to the subordinate node and completes the access.
17. The subordinate node releases the stretched I^2C bus, provides the STOP bit to the remote peripheral, and completes the access on its end.

For every byte from the host processor, the main node stretches the I^2C bus and waits for acknowledgment from the subordinate node. This wait state has a timeout of superframes. If the main node does not receive ACK within this period, it provides NAK to the host on the I^2C bus and internally instructs the subordinate node to abort the access. The host processor (I^2C controller) must support clock stretching.

NOTE: The clock stretching behavior is applicable when `A2B_I2CCFG.EACK` = 0 (default). The I^2C transactions are clock-stretched until they are complete in the system so that a correct ACK/NACK signal can be generated by the I^2C interface. When `A2B_I2CCFG.EACK`= 1, the main node provides early acknowledgment to write bytes before the write has completed on the A^2B bus. If there is an error (for example, a timeout or address error), the I2CERR error is raised. The error may be raised after the completion of a failed I^2C transaction. The host processor must verify the previous access when this error occurs.

The extra time added to the remote peripheral I^2C access over A^2B bus when compared with a direct I^2C access to a remote peripheral is known as access latency through the A^2B bus. The I^2C access is switched to/from A^2B access. The clock is stretched during this time. This latency is mostly dependent on the I^2C clock frequency of the subordinate node that replicates the access on the local I^2C bus. It also depends on A^2B bus availability for the I^2C access and the superframe timing with respect to I^2C accesses at both ends (main node and subordinate node). The I^2C access is converted to and from superframes on the A^2B bus. There is a variable latency of 1 or 2 superframes for internal communication between main node and subordinate node. Typically, the I^2C accesses are asynchronous with the superframes timing (which depends on the SYNC signal timing of main node).

The ***Subordinate Node Access Latency*** table provides the typical latency number for accessing subordinate node registers. These numbers are on the slow side of the average, based on simulation.

Table 5-4: Remote I^2C Peripheral Access Latency

I^2C Access	No. of Bytes	**A2B_I2CCFG. DATARATE**	I^2C Frequency	Bus Latency
Read/Write	N	0	100KHz	213 µs + (N - 1) x 113 µs
Read/Write	N	1	400KHz	70 µs + (N - 1) x 45 µs

Remote Peripheral I^2C Read Access

The ***I^2C Read Timing*** figure shows a typical remote peripheral I^2C read access.

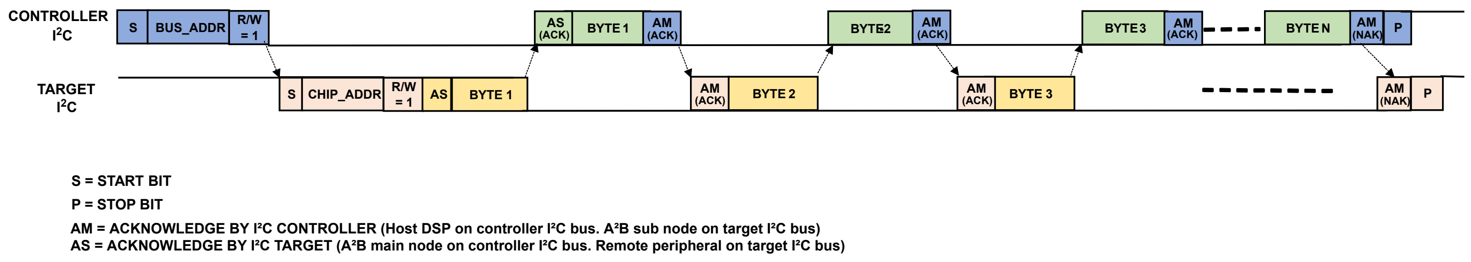

CONTROLLER I²C
S
BUS_ADDR
R/W = 1
AS (ACK)
BYTE 1
AM (ACK)
BYTE 2
AM (ACK)
BYTE 3
AM (ACK)
BYTE N
AM (NAK)
P
TARGET I²C
S
CHIP_ADDR
R/W = 1
AS
BYTE 1
AM (ACK)
BYTE 2
AM (ACK)
BYTE 3
AM (NAK)
P
S = START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY I²C CONTROLLER (Host DSP on controller I²C bus. A²B sub node on target I²C bus)
AS = ACKNOWLEDGE BY I²C TARGET (A²B main node on controller I²C bus. Remote peripheral on target I²C bus)

Figure 5-9: I^{2}C Read Timing

The access is shown in terms of bytes instead of register address bytes and register value data bytes because the number of address bytes and the register width depend on the remote peripheral.

Consider the following cases:

Case 1 – The peripheral register address is 1 byte and each register is 8 bits wide. In this case, BYTE 1 represents the address byte and BYTE 2 represents register value to be written. If a peripheral supports the register address auto-increment feature and subsequent registers must be programmed, then a burst access can be initiated. In this case, BYTE 3 and onwards represent values to be written into the next registers.

Case 2 – The peripheral register address is 2 bytes long and each register is 4 bytes wide. In this case, BYTE 1 and BYTE 2 represent the address bytes; BYTE 3 to BYTE 6 represents register value to be written. The host processor must provide these bytes based on the peripheral byte order requirement (LSB byte first or MSB byte first). For the host processor, the access is like it is directly accessing the remote peripheral register, except that the access must be initiated with BUS_ADDR as the I^{2}C device address instead of the actual remote peripheral I^{2}C address.

Sometimes, the I^{2}C read access is started with write command to set the register address. However, the access does not necessarily need a write first if the address is already set. The write to set the address is a separate I^{2}C transaction. If the register address set/write part is needed at the start, the read access can be started with a repeated start condition after the write. The ***I^{2}C Repeated Access*** figure shows an example of a read access with register set access at the start. The peripheral register address is 2 bytes long and each register is 2 bytes wide.

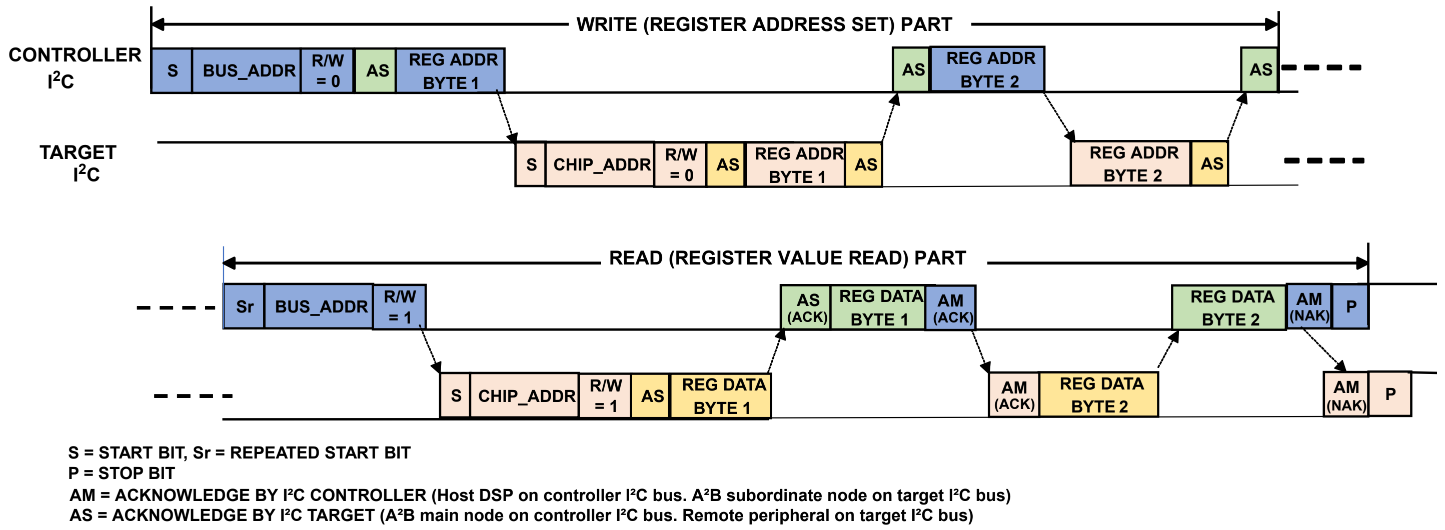

WRITE (REGISTER ADDRESS SET) PART
CONTROLLER I²C
TARGET I²C
S
BUS_ADDR
R/W = 0
AS
REG ADDR BYTE 1
REG ADDR BYTE 2
CHIP_ADDR
READ (REGISTER VALUE READ) PART
Sr
R/W = 1
AS (ACK)
REG DATA BYTE 1
AM (ACK)
REG DATA BYTE 2
AM (NAK)
P
S = START BIT, Sr = REPEATED START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY I²C CONTROLLER (Host DSP on controller I²C bus. A²B subordinate node on target I²C bus)
AS = ACKNOWLEDGE BY I²C TARGET (A²B main node on controller I²C bus. Remote peripheral on target I²C bus)

Figure 5-10: I^{2}C Repeated Access

The typical flow for an I^{2}C read access is:

1. The host processor sets the peripheral register address using the I^{2}C write sequence.
2. The host processor starts the read access with the repeated start (Sr) bit if the previous access was a register set or write part of the command. Or, it starts the read access with a normal start bit if a STOP bit was issued for an earlier access.
3. The host processor initiates the peripheral read access with BUS_ADDR and R/W bit = 1. It waits for acknowledgment from the I^{2}C subordinate (A^{2}B main node).
4. Before providing the acknowledgment to this byte, the main node stretches the I^{2}C bus with the host processor and internally relays the information to the targeted subordinate node (connected to the remote peripheral).
5. The subordinate node becomes the I^{2}C controller and checks whether the I^{2}C bus is free. If the local I^{2}C bus of the subordinate node is not free (due to a multi controller environment), the subordinate node aborts the access after superframes and reports the error to the main node. The main node provides the NAK to the host indicating that the access must be attempted again.
6. If the I^{2}C bus is free, the subordinate node initiates the access with the peripheral I^{2}C device address. The subordinate transceiver uses the value of its `A2B_CHIP` register as the I^{2}C device address for the local access. It places R/W bit = 1 on the bus.
7. The remote peripheral receives the read command. It internally fetches the register value (whose address was set during the write part of the command). If the data byte is available, the remote peripheral provides the ACK bit, conveying to the I^{2}C controller (A^{2}B subordinate node) that the data is ready. If the data is not ready, the remote peripheral stretches the I^{2}C bus until the data is available.

8. After receiving the ACK bit, the A^2B subordinate node (I^2C controller) provides the I^2C clocks so that the remote peripheral can drive out the data bits. Upon receiving ACK and the data byte from the remote peripheral, the subordinate node internally forwards the details to the A^2B main node (using the SRF field of the superframe). The subordinate node stretches its I^2C bus until further instruction from the main node.
9. Upon receiving the internal ACK and the data byte, the main node releases the I^2C bus (previously stretched) and provides the ACK to the host processor on the I^2C bus.
10. The host processor provides clocks on I^2C bus such that the A^2B main node can drive out the received data.
11. If further data bytes are needed from a remote peripheral, the host processor places the ACK bit on the I^2C bus, conveying to the main node information about the extension of access.
12. The main node internally conveys to the subordinate node the extension of the access and the requirement of further bytes
13. The subordinate node releases its I^2C bus (previously stretched) and provides the ACK bit to the remote peripheral.
14. Steps 7 to 13 repeat until the host processor receives the required number of bytes from the remote peripheral.
15. The host processor provides the NACK and the STOP bit for the transaction on the I^2C bus, thereby completing the I^2C access.
16. The main node receives the NAK and STOP bit for the ongoing I^2C access. It communicates the information to the subordinate node and completes the access.
17. The subordinate node receives the NAK bit and STOP command from the main node. It releases the stretched I^2C bus. The subordinate node provides the NAK and STOP bits to the remote peripheral and completes the access on its end.

For every byte from host processor, the main node stretches the I^2C bus and waits for acknowledgment from the subordinate node. This wait state has a timeout of superframes. If the main node does not receive acknowledgment or the data within this period, it provides NAK to the host on the I^2C bus and internally instructs the subordinate node to abort the access. The host processor (I^2C controller) must support clock stretching.

NOTE: The `A2B_I2CCFG.EACK` bit setting does not affect remote peripheral I^2C read transactions.

The time added to the remote peripheral I^2C access over the A^2B bus when compared with a direct I^2C access to a remote peripheral is known as access latency through the A^2B bus. The I^2C access is switched to and from the A^2B access and the clock is stretched during this time. The latency depends on:

- the I^2C clock frequency of the subordinate node (replicates the access on the local I^2C bus)
- A^2B bus availability for the I^2C access
- superframe timing with repect to the I^2C accesses at both ends (main node and subordinate node).

The I^{2}C access is converted to and from the superframe on the A^{2}B bus. Therefore, there is a variable latency of 1-2 superframes for internal communication between the main node and subordinate node. Typically, I^{2}C accesses are asynchronous with superframe timing that depends on the SYNC signal timing of the main node.

The read latency number is similar to the write latency for a remote peripheral. See the Table 5-4 Remote I^{2}C Peripheral Access Latency table. If an address must be sent first, it must be calculated separately as a write transaction.

IMPORTANT: The I^{2}C interface can be disabled by setting the `A2B_I2CCFG.DISI2C` bit to repurpose I^{2}C pins as GPIO.

Local Processor to Subordinate Node Access

A local processor can access the registers of a connected subordinate node. This local I^{2}C access occurs between the processor and the connected subordinate node transceiver. The access works similar to the host processor accessing main node registers. The ***Local I2C Access*** figure shows an access from a subordinate node to a local processor.

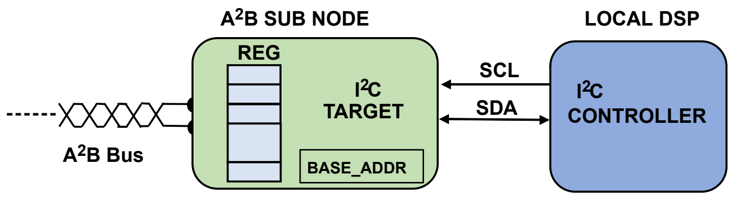


Figure 5-11: Local I^{2}C Access

The local processor can directly access the subordinate node registers via I^{2}C bus using the BASE_ADDR of transceiver. The BASE_ADDR is decided during node power-up from the status of ADR1/2 pins of the transceiver. The subordinate node I^{2}C interfaces acts as an I^{2}C target. The local processor can access (read/write) a single register or multiple sequential registers in burst mode. In burst access mode, the transceiver automatically increments the register address pointer after each data byte. Therefore, sequential data registers can be accessed without reprogramming the address.

The ***Local Accesses to Subordinate Node - Bit Sequence*** figure shows the bit sequence for different types of local accesses to subordinate node registers.

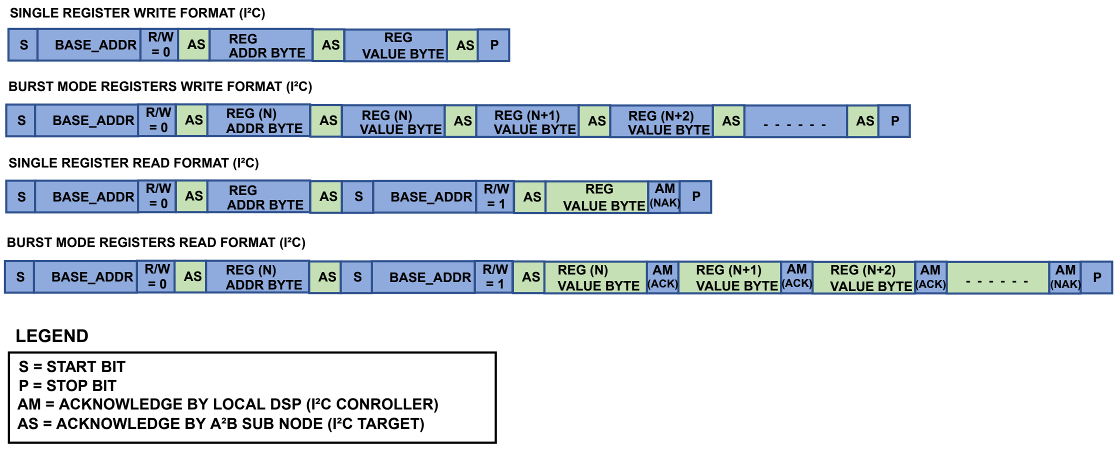

SINGLE REGISTER WRITE FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | REG VALUE BYTE | AS | P
BURST MODE REGISTERS WRITE FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | REG (N) VALUE BYTE | AS | REG (N+1) VALUE BYTE | AS | REG (N+2) VALUE BYTE | AS | - - - - - - | AS | P
SINGLE REGISTER READ FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG ADDR BYTE | AS | S | BASE_ADDR | R/W = 1 | AS | REG VALUE BYTE | AM (NAK) | P
BURST MODE REGISTERS READ FORMAT (I²C)
S | BASE_ADDR | R/W = 0 | AS | REG (N) ADDR BYTE | AS | S | BASE_ADDR | R/W = 1 | AS | REG (N) VALUE BYTE | AM (ACK) | REG (N+1) VALUE BYTE | AM (ACK) | REG (N+2) VALUE BYTE | AM (ACK) | - - - - - - | AM (NAK) | P
LEGEND
S = START BIT
P = STOP BIT
AM = ACKNOWLEDGE BY LOCAL DSP (I²C CONROLLER)
AS = ACKNOWLEDGE BY A²B SUB NODE (I²C TARGET)

Figure 5-12: Local Accesses to Subordinate Node - Bit Sequence

I²S/TDM Interface

The I²S/TDM serial port operates in full-duplex mode, where both the transmitter and receiver operate simultaneously using the same critical timing bit clock (BCLK) and frame synchronization (SYNC) signals. A²B subordinate transceivers generate the timing signals on the BCLK and SYNC output pins with frequencies based on the settings in the I²S global configuration register (A2B_I2SGCFG), the I²S rate register (A2B_I2SRATE), and the I²S reduced rate register (A2B_I2SRRATE). A²B main transceivers use the same BCLK and SYNC pins as inputs. The host drives the pins which provides the time base for the full A²B bus topology.

I²S Pin Configuration

The I²S TDM block supports a combination of I²S/TDM TX, I²S/TDM RX, and PDM microphone data that is configured with pin routing.

The internal I²S/TDM port of the base transceiver has a total of seven pins with five pins (SIO0- SIO4) available for data. The port supports I²S/TDM RX data on up to four data pins and I²S/TDM TX data on up to four data pins. The receive data streams are DRX0, DRX1, DRX2, and DRX3. The transmit data streams are DTX0, DTX1, DTX2, and DTX3.

In addition to supporting I²S/TDM RX and I²S/TDM TX data steams, data pins SIO0 and SIO1 can also be used for PDM microphone data streams. The PDM microphone data streams are PDM0 and PDM1.

The programming model provides bit fields to flexibly configure the number of I²S/TDM RX pins, the number of I²S/TDM TX pins, and the number of PDM microphone pins. However, the user is limited to configuring a total of five data steams. For proper operation, ensure that the value of A2B_PDMCTL.PDM0EN + A2B_PDMCTL.PDM1EN + (A2B_I2SCFG.RXPINS + A2B_I2SCFG.TXPINS) ≤ 5.

The data stream pin mapping depends on the state of the A2B_PDMCTL.PDM0EN and A2B_PDMCTL.PDM1EN bits as described in the *Configuration Modes* table.

Table 5-5: Configuration Modes

Pins	PDM0EN=0
	PDM1EN=0
		PDM0EN=1
		PDM1EN=0
			PDM0EN=0
			PDM1EN=1
				PDM0EN=1
				PDM1EN=1

SIO0	DRX0	PDM0	DRX0	PDM0
SIO1	DRX1/DTX3	DRX0/DTX3	PDM1	PDM1
SIO2	DRX2/DTX2	DRX1/DTX2	DRX1/DTX2	DRX0/DTX2
SIO3	DRX3/DTX1	DRX2/DTX1	DRX2/DTX1	DRX1/DTX1
SIO4	DTX0	DTX0	DTX0	DTX0

Time Division Multiplexing (TDM) Protocol

TDM mode extends an I^{2}S interface to more than a stereo 2-channel (TDM2) signal. When the transceiver is programmed in the A2B_I2SCFG register to support a certain number of TDM channels, this number of TDM channels is available on each enabled I^{2}S/TDM data pin (DRX0, DRX1, DRX2, and DRX3 or DTX0, DTX1, DTX2, and DTX3). The interface supports TDM2, TDM4, TDM8, TDM12, TDM16, TDM20, TDM24, and TDM32 modes.

The ***Data Channel Structure for TDM2 Setting*** figure shows the transmit data channels when one, two, three or four pins are enabled and TDM2 is selected. When using two data pins, the channels can have an interleaved or non-interleaved format. For 1-pin, 3-pin, and 4-pin TX or RX modes, the channels are always in non-interleaved format. The pin interleaving settings for DTX pins and DRX pins can be independently configured.

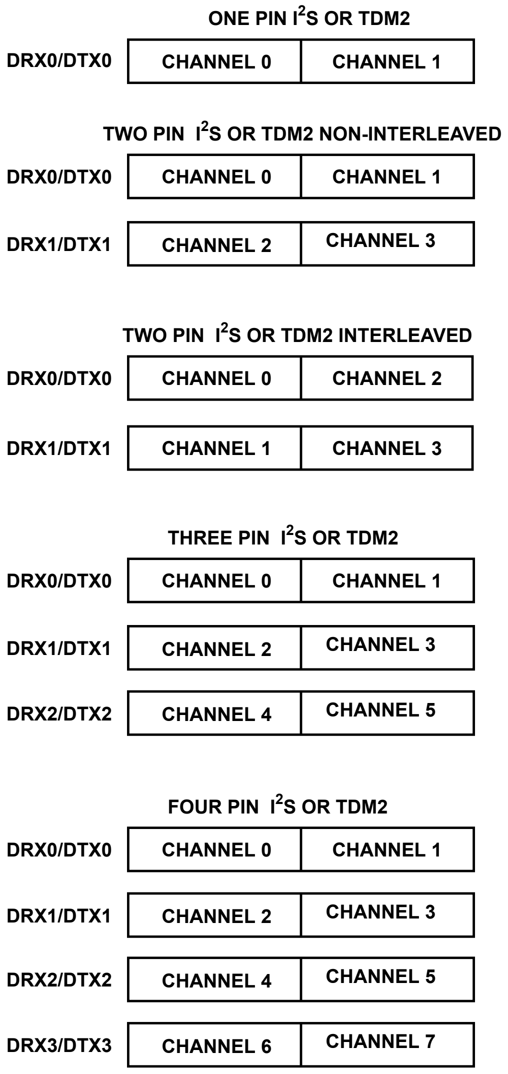


Figure 5-13: Data Channel Structure for TDM2 Setting (TDMMODE == 000)

The I^{2}S/TDM serial port supports data channel widths of 16 bits or 32 bits to carry signals of varying word length. Data words are always represented in MSB first format.

BCLK = SYNC rate x Number of channels per TDM frame x channel size

The BCLK signal frequencies for different TDM modes are shown in the ***I^{2}S/TDM Clock Frequency Settings for 48 kHz Superframe Rates*** table.

Table 5-6: I^{2}S/TDM Clock Frequency Settings for 48 kHz Superframe Rates

TDM Mode	16-bit TDM Channel Size	32-bit TDM Channel Size
	Frequency (MHz)	Frequency (MHz)
TDM2	1.536	3.072

Table 5-6: I^{2}S/TDM Clock Frequency Settings for 48 kHz Superframe Rates (Continued)

TDM Mode	16-bit TDM Channel Size	32-bit TDM Channel Size
	Frequency (MHz)	Frequency (MHz)
TDM4	3.072	6.144
TDM8	6.144	12.288
TDM12 (No subordinate node support)	9.216	18.432
TDM16	12.288	24.576
TDM20 (No subordinate node support)	15.36	30.72
TDM24 (No subordinate node support)	18.432	36.864
TDM32	24.576	49.152

The SIO0 and SIO1 input pins can be configured individually as PDM inputs. When PDM is enabled on an A^{2}B subordinate node on one or both of the SIO0 and SIO1 pins, a PDM clock running at 64 × f_{SYNCM} (3.072 MHz at 48 kHz f_{SYNCM}) is required to clock the PDM device. Either the PDMCLK/ pin or the BCLK pin can produce the required PDM clock. The transceiver can simultaneously transmit and receive TDM data while receiving PDM streams. However, when BCLK is used as the PDM clock, only I^{2}S/TDM2 and 32-bit channel widths or TDM4 with 16-bit channel widths are supported. Using PDMCLK/ instead of BCLK to clock PDM devices allows BCLK to be used for a variety of TDM modes. If both SIO0 and SIO1 are used to receive PDM data, this configuration enables the concurrent use of up to four PDM microphones and full duplex I^{2}S/TDM communication.

I^{2}S/TDM Port Programming Concepts

Programming the I^{2}S/TDM interface involves selecting the mode of operation for the port, controlling how many data pins are enabled for both transmit and receive operations, and configuring the polarity and timing of the BCLK and SYNC signals relative to data.

NOTE: Typically, the I^{2}S/TDM interface of the A^{2}B main node is the clock target, receiving BCLK and SYNC signals from the audio host.

Similarly, the I^{2}S/TDM interface of the A^{2}B subordinate nodes is the clock controller, providing BCLK and SYNC signals to the connected audio peripherals (for example, an audio codec). However, in parallel bus operation mode, the TDM interface of the subordinate node works in clock target mode, accepting BCLK and SYNC signals from the paired subordinate node. Refer to Parallel A^{2}B Buses for details.

The `A2B_I2SGCFG` and `A2B_I2SCFG` registers are used to configure the I^{2}S/TDM port to support various modes of operation. The ***Serial Mode Data and Clock Formats*** table provides a summary of the different data and clock formats supported by both main and subordinate transceivers.

Table 5-7: Serial Mode Data and Clock Formats

Bit Setting	Data and Clock Format
A2B_I2SGCFG.EARLY =0	SYNC pin changes in the same cycle as the MSB of Data Channel 0
A2B_I2SGCFG.EARLY =1	Early SYNC mode. SYNC pin changes one cycle before the MSB of Data Channel 0
A2B_I2SGCFG.ALT =0	Pulsed SYNC mode. SYNC pin is driven active for one BCLK cycle at the start of each sampling period
A2B_I2SGCFG.ALT =1	50% duty cycle SYNC mode. SYNC pin is driven high at the beginning of each sampling period and low in the middle of each sampling period
A2B_I2SGCFG.INV =0	Active high SYNC signal mode. Rising edge of SYNC references the first channel (Channel 0)
A2B_I2SGCFG.INV =1	Active low SYNC signal mode. Falling edge of SYNC references the first channel (Channel 0)
A2B_I2SCFG.RXBCLKINV =0	DRX and SYNC pins are sampled on the rising edge of BCLK
A2B_I2SCFG.TXBCLKINV =0	DTX and SYNC pins change on the rising edge of BCLK
A2B_I2SCFG.RXBCLKINV =1	DRX and SYNC pins are sampled on the falling edge of BCLK
A2B_I2SCFG.TXBCLKINV =1	DTX and SYNC pins change on the falling edge of BCLK

To support more than a stereo two-channel (TDM2) signal, the A2B_I2SGCFG.TDMMODE field must be set to enable any of the supported TDM modes of operation. Once configured, this is the operating mode used for each of the enabled data pins, as controlled by the A2B_I2SCFG.RXPINS and A2B_I2SCFG.TXPINS bits.

When two data pins in either direction are enabled, the interleaving feature can be enabled by setting the respective two-pin interleave (A2B_I2SCFG.RXPINS = 3'b111 and A2B_I2SCFG.TXPINS = 3'b111) bits. When set, the even slot data is associated with the SIO4(DTX0)/SIO0(DRX0) data pin, and the odd slot data is associated with the SIO3(DTX1)/SIO1(DRX1) data pin. When cleared, the lower half of the enabled slots are associated with the SIO4(DTX0)/SIO0(DRX0) data pin, and the upper half of the enabled slots are associated with the SIO3(DTX1)/SIO1(DRX1) data pin. For example, if the data format is set for I^{2}S or TDM2 mode, the ***Data Channel Structure for TDM2 Setting*** figure summarizes how the data is aligned.

The A2B_I2SGCFG.TDMSS bit selects between 16-bit and 32-bit serial data for the I^{2}S/TDM port. The host must ensure that the appropriate timing signals are provided to accommodate the full window of data. For example, if TDM8 mode is selected (A2B_I2SGCFG.TDMMODE = 0b010), the host must provide either 128 (8 x 16-bit, when A2B_I2SGCFG.TDMSS = 1) or 256 (8 x 32-bit, when A2B_I2SGCFG.TDMSS = 0) BCLK pulses for the data and the appropriate SYNC signal (to be either pulsed or held for a 50% duty cycle, per the setting of the A2B_I2SGCFG.ALT bit), as shown in the ***I^{2}S/TDM8 Example Timing*** figure.

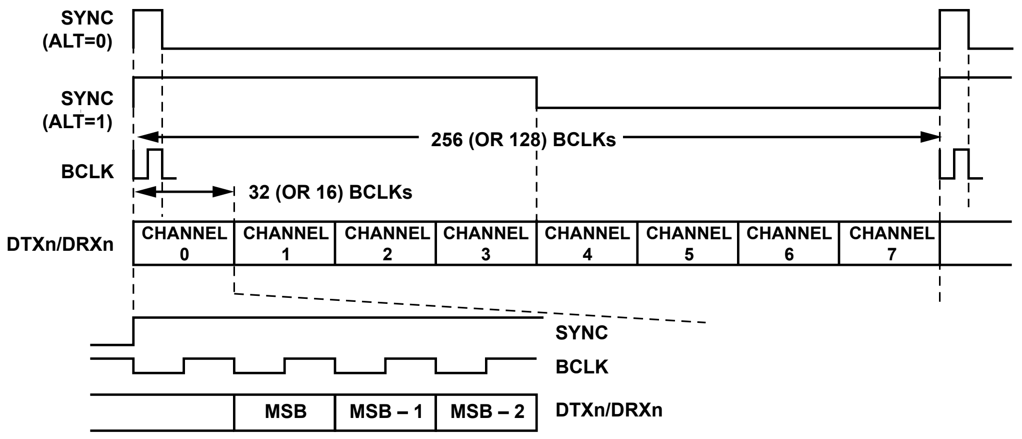


Figure 5-14: I^{2}S/TDM8 Example Timing

As shown in the ***I^{2}S/TDM8 Example Timing*** figure, the TDM channel data is in MSB-first format. When the data being exchanged over the A^{2}B bus is not exactly 32-bit, the transceiver expects the input TDM data channels to arrive in MSB-first format and disregards any lower-order bits. When outputting to the local node, the transceiver presents the received A^{2}B slot data to the I^{2}S/TDM port in MSB-first format with the unused lower-order bits zero-filled. For example, if the A^{2}B slot is configured for 12-bit data (A2B_SLOTFMT.UPSIZE = 1 for upstream slots or A2B_SLOTFMT.DNSIZE = 1 for downstream slots), the 12-bit input data must be left-justified in the TDM channel, and output data consists of the 12-bit A^{2}B slot data followed by four zero bits.

The A^{2}B transceiver supports SYNC pin change in the same cycle as the MSB bit of channel-0 or one cycle before the MSB bit of channel-0. In Early SYNC mode, the SYNC signal arrives one bit earlier, and it can be rephased to data arriving one bit later than the relevant edge on the SYNC signal.

The A^{2}B transceiver supports independent settings for driving edge and sampling edge of signals, configurable using A2B_I2SGCFG register.

The *Driving edge ≠ Sampling Edge* figure shows the typical timing for I^{2}S/TDM interface where output data is provided on one edge of BCLK and input data is sampled on the opposite edge of BCLK (A2B_I2SCFG.TXBCLKINV≠ A2B_I2SCFG.RXBCLKINV).

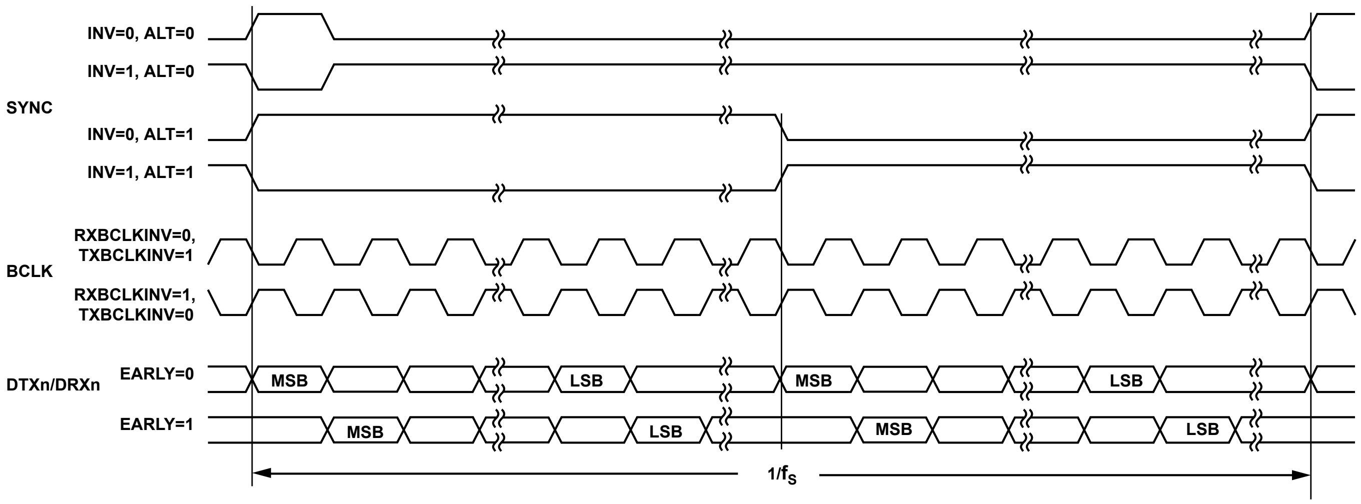


Figure 5-15: Driving Edge ≠ Sampling Edge

CAUTION: Higher TDM modes such as TDM16 or TDM32 increase the BCLK rate to a speed at which setup time and hold time violation can occur, leading to incorrect I^{2}S/TDM operation. For high TDM rates, perform a timing analysis for the TDM interface, considering the timing requirements and switching characteristics of the A^{2}B transceiver and the interfaced device. For A^{2}B transceiver timing requirements and switching characteristics, refer to the product data sheet.

For higher TDM modes, it is possible to achieve valid setup time margins by advancing the driving edge or delaying the sampling edge, as the driving edge and sampling edge settings are fully configurable.

- When the A^{2}B transceiver is configured as the I^{2}S target, if `A2B_I2SCFG.TXBCLKINV` = `A2B_I2SCFG.RXBCLKINV`, the transceiver samples the input data on a BCLK edge and drives the output data on the previous, same polarity BCLK edge, as shown in the *Driving Edge = Sampling Edge (I^{2}S Target)* figure.

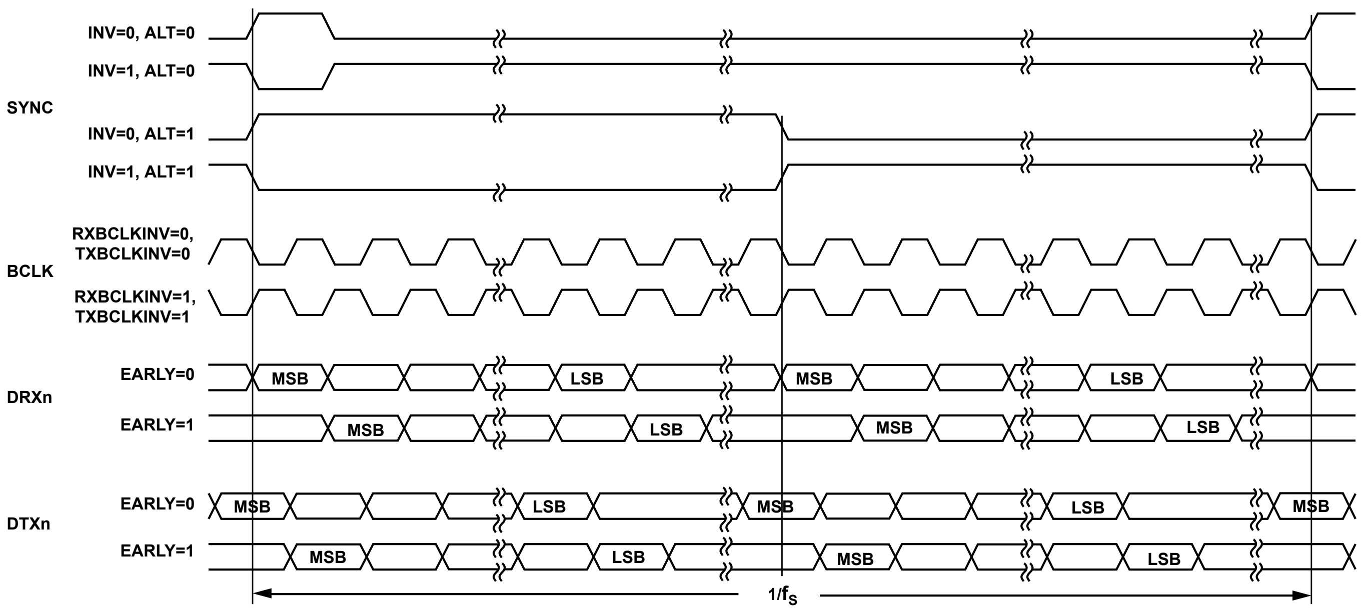


Figure 5-16: Driving Edge = Sampling Edge (I^2S Target)

- When the A^2B transceiver is configured as the I^2S controller, if `A2B_I2SCFG.TXBCLKINV` = `A2B_I2SCFG.RXBCLKINV`, the transceiver changes data on a BCLK edge and samples data on the next, same polarity BCLK edge, as shown in the *Driving Edge = Sampling Edge (I^2S Controller)* figure.

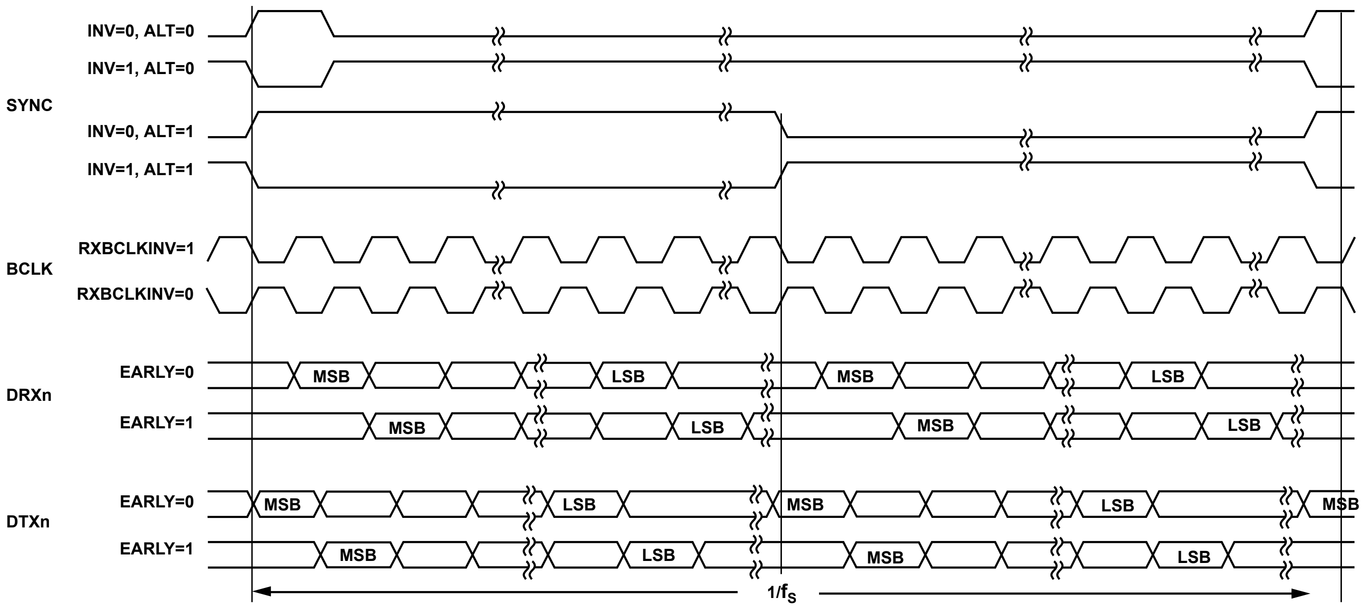


Figure 5-17: Driving Edge = Sampling Edge (I^2S Controller)

I^2S Frame Rates

Subordinate transceivers can run the I^2S/TDM/PDM interface at a reduced or increased rate frequency with respect to the superframe rate. In the increased sampling rate modes, the TDM interface of a node can run at 2x and 4x superframe rates (f_{SYNCM}). These modes are useful in premium audio applications where 96 KHz or 192 KHz audio sampling rates are needed. While in the reduced sampling rate modes, the TDM data rate is derived by dividing the superframe rate from a programmable set of values. This reduced rate is useful for PDM microphone use cases for controlling the microphone data rate. Each subordinate node transceiver can be configured to run at a different I^2S/TDM rate.

I^2S Increased Data Rate

The A^2B subordinate transceiver supports increased sampling rates at the I^2S/TDM interface with respect to the superframe rate (f_{SYNCM}). The local sampling rate of the subordinate node can be programmed to 1 f_{SYNCM}, 2 f_{SYNCM}, or 4 f_{SYNCM} in the A2B_I2SRATE register. For example, given a 48 kHz superframe frequency, the local sampling rate can be set to 48 kHz, 96 kHz, or 192 kHz, respectively. The ***Increased Data Rate*** figure shows how the downstream and upstream slots from the A^2B superframe are distributed on the DTX and DRX pins in the subordinate transceiver for different `A2B_I2SRATE` bit settings (with `A2B_I2SRATE.REDUCE` = 0) in a system with one controller and one subordinate node.

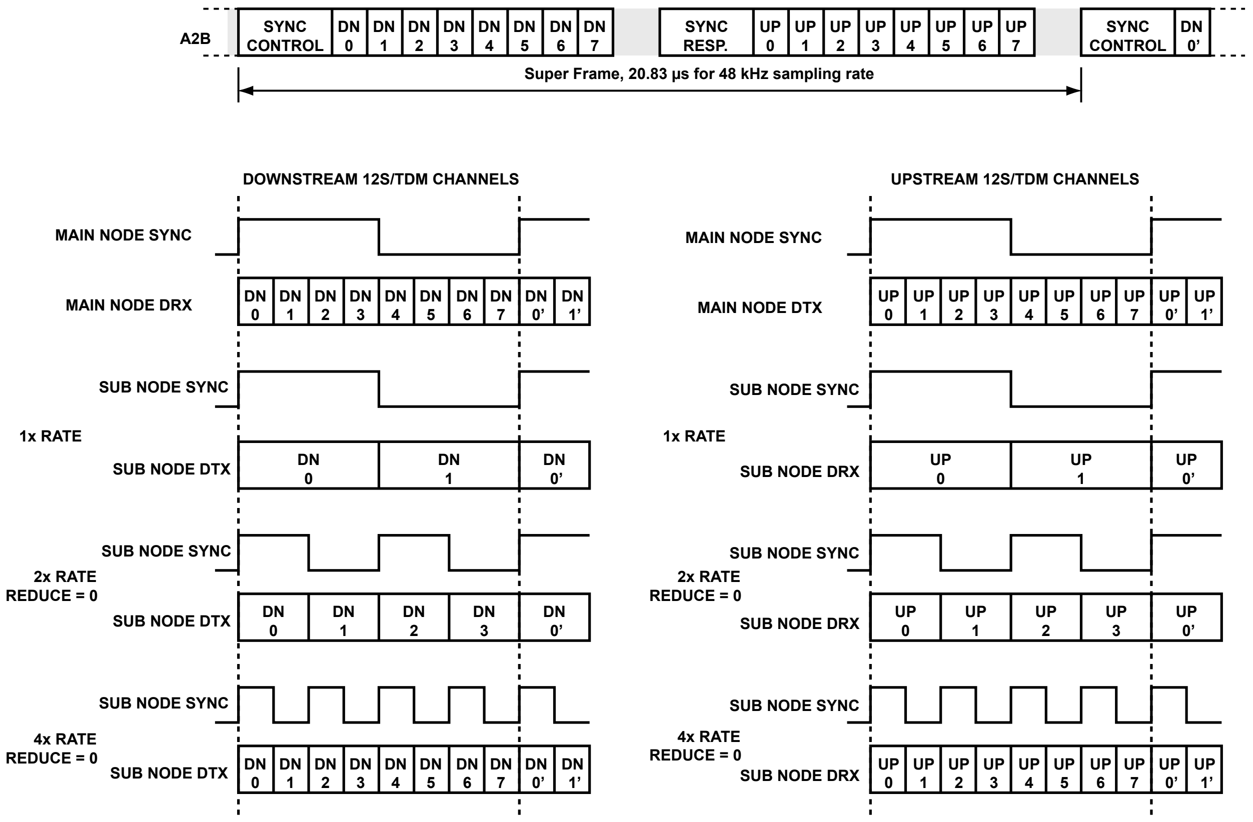


Figure 5-18: Increased Data Rate

The ***Increased Data Rate Example*** figure further illustrates the behavior of the `A2B_I2SRATE` register settings based on an example system. In the figure, both subordinate transceivers (S1 and S2) are set to 2 f_{SYNCM} rate mode. However, S1 has the `A2B_I2SRATE.REDUCE` bit set to 1. The waveforms in the figure illustrate the effect of the `A2B_I2SRATE.REDUCE` bit for both upstream and downstream slots. When the `A2B_I2SRATE.REDUCE` bit is set, only the first two channels on the DRX pin are used for the upstream slots, and the other two channels are ignored for 2 f_{SYNCM} rate. For the DTX transmitter, the two local downstream slots are duplicated on the DTX pins for a 2 f_{SYNCM} rate when the `A2B_I2SRATE.REDUCE` bit is set.

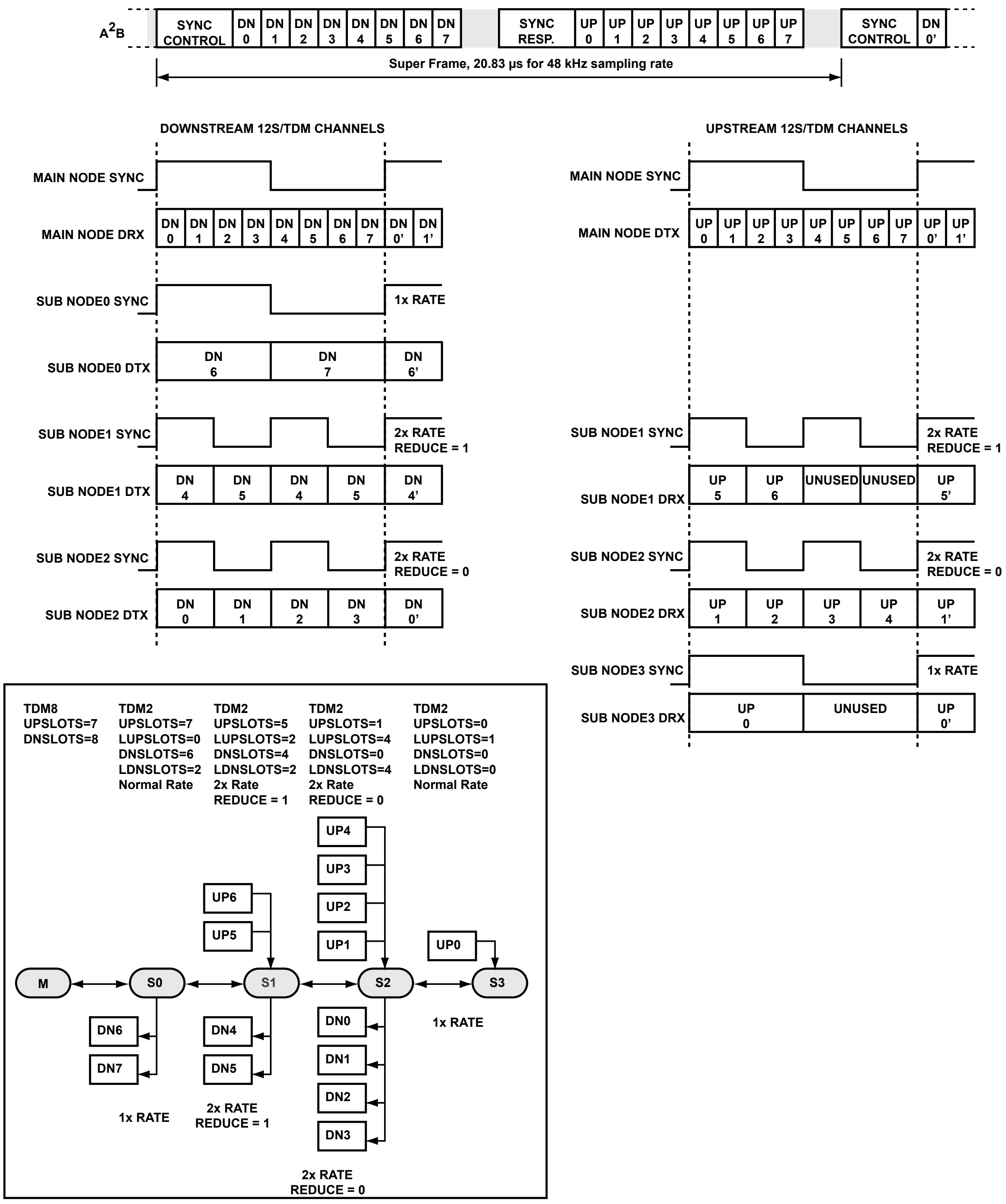


Figure 5-19: Increased Data Rate Example

NOTE: The increased data rate feature is not a sample rate converter. The mapping of frame buffer entries to TDM slots happens sequentially, so increased rate TDM requires enough bus slots to get the data aligned.

Alternatively, the TX crossbar feature can be used produce a expected TX mapping that looks like sample conversion.

I^2S Reduced Data Rate

Subordinate nodes can also run the I^2S/TDM interface at a reduced rate frequency with respect to the superframe rate (f_{SYNCM}). The reduced-rate frequency is derived by dividing the superframe rate by a programmable set of values. Different subordinate nodes can be configured to run at different reduced I^2S/TDM rates.

The `A2B_I2SRATE.I2SRATE` bit field is used to divide the superframe A^2B rate down to the reduced I^2S rate. It also provides a control bit, `A2B_I2SRRATE.RBUS`, to enable reduced-rate data slots on the bus. The A^2B data slots on the bus are transmitted only once every `A2B_I2SRRATE.RRDIV` superframes.

The `A2B_I2SRATE.I2SRATE` bit field can be used to program the division factor to 2, 4, or as configured in the `A2B_I2SRRATE.RRDIV` field. The `A2B_I2SRATE.SHARE` bit enables the shared A^2B bus slots in a reduced-rate subordinate node, provided the node has the I^2S transmit disabled.

The `A2B_I2SRRCTL` register provides bits to allow a processor to track the full-rate audio frame, which contains new reduced-rate samples. The pin can be used as a strobe by setting the `A2B_I2SRRCTL.ENSTRB` bit, which indicates the audio frame where reduced-rate data is updated. The `A2B_I2SRRCTL.STRBDIR` bit configures the direction of the pin when used as a strobe. The reduced rate strobe output at the main node is based on the `A2B_I2SRRATE.RRDIV` field setting. When the `A2B_I2SRRATE.RRDIV` field is not one, the reduced rate count is maintained in each node, and the strobe output signal is generated accordingly. When the strobe is an input, it is sampled on the active edge of SYNC, and the reduced rate count is synchronized to it. The user must create a strobe signal that matches the `A2B_I2SRRATE.RRDIV` setting.

The `A2B_I2SRRSOFFS` register provides a bit field to move the SYNC edge in a reduced-rate subordinate node in superframe increments.

The ***Reduced Data Rate*** figure shows how the upstream slots from the transceiver can reduce the superframe rate on the bus, allowing the subordinate nodes to run at a reduced-sample frequency with both sharing disabled (`A2B_I2SRATE.SHARE` = 0) and enabled (`A2B_I2SRATE.SHARE` = 1). This figure is drawn for a system with one main node and one subordinate node.

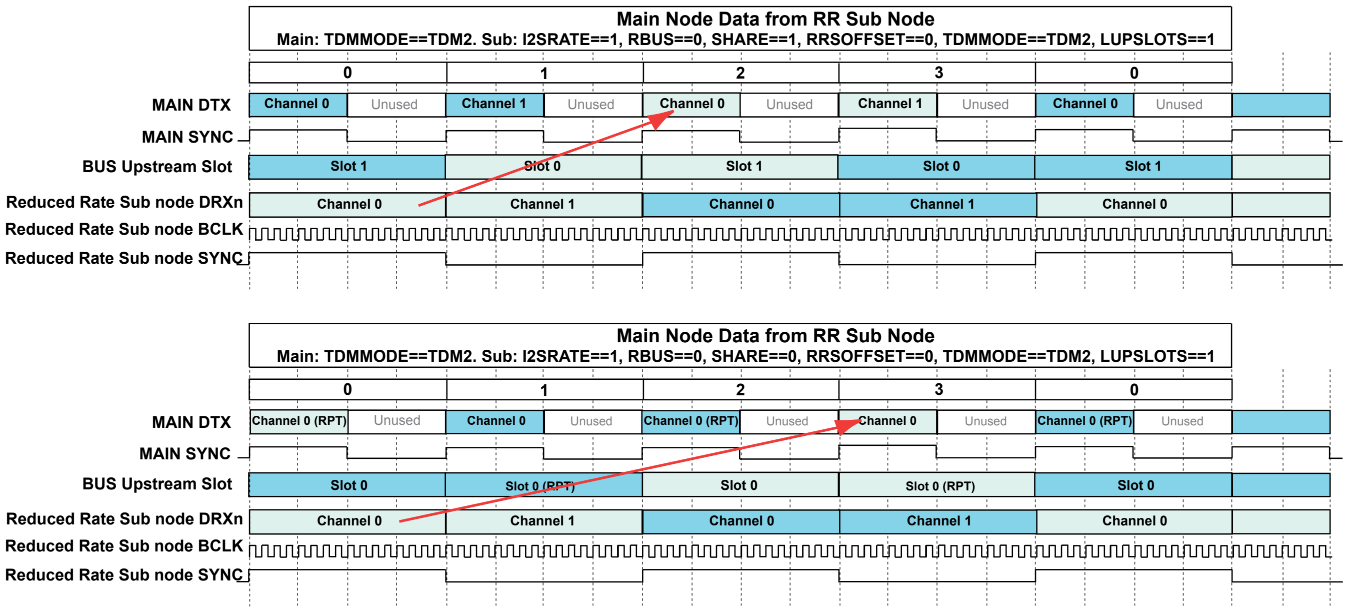

Main Node Data from RR Sub Node
Main: TDMMODE==TDM2. Sub: I2SRATE==1, RBUS==0, SHARE==1, RRSOFFSET==0, TDMMODE==TDM2, LUPSLOTS==1
MAIN DTX
MAIN SYNC
BUS Upstream Slot
Reduced Rate Sub node DRXn
Reduced Rate Sub node BCLK
Reduced Rate Sub node SYNC
Channel 0
Channel 1
Unused
Slot 0
Slot 1
Main Node Data from RR Sub Node
Main: TDMMODE==TDM2. Sub: I2SRATE==1, RBUS==0, SHARE==0, RRSOFFSET==0, TDMMODE==TDM2, LUPSLOTS==1
Channel 0 (RPT)
Slot 0 (RPT)

Figure 5-20: Reduced Data Rate

The following table shows the I^{2}S/TDM sampling rates categorized into system modes for the reduced rate.

Table 5-8: I^{2}S/TDM Sampling Rates Categorized into System Modes for Reduced Rate

Mode	Host I^{2}S/TDM Rate	Bus Data Slots	Subordinate Rate(s)	Channels
1	Set in `A2B_I2SRRATE.RRDIV`	Set in `A2B_I2SRRATE.RRDIV`	Set in `A2B_I2SRRATE.RRDIV`	1 - 32
2	48 kHz	Set in `A2B_I2SRRATE.RRDIV`	Set in `A2B_I2SRRATE.RRDIV`	1 - 32
3	48 kHz	48 kHz	Set in `A2B_I2SRRATE.RRDIV`	1 - 32
4	48 kHz	48 kHz	Set in `A2B_I2SRRATE.RRDIV`	1 - 128
5	48 kHz	48 kHz	Set in `A2B_I2SRRATE.RRDIV`, 1/4x, 1/2x, 1x, 2x, 4x	1 - 32

The reduced rate feature allows system designers to add the following functionality:

1. Subordinate nodes can run the I^{2}S/TDM interface at a reduced rate divided from the superframe rate, as divided down from the superframe rate. For example, reduced rates for a 48 kHz superframe rate are 24 kHz, 12 kHz, 6 kHz, 4 kHz, 3 kHz, 2.4 kHz, 2 kHz, 1.71 kHz, or 1.5 kHz. The I^{2}S/TDM RX data on the subordinate node can be sent either upstream or downstream at the reduced rate.

   Different subordinate nodes can run at different reduced I^{2}S/TDM rate.

2. The SYNC signal of the reduced-rate subordinate node can be adjusted in superframe increments to ensure minimum latency on the delivery of reduced-rate data.
3. Control of the BCLK signal generation can minimize a delay by quick sampling at the reduced-rate I^2S data (for example, within a 48 kHz I^2S/TDM frame) or sampling at the reduced I^2S/TDM rate.
4. Options to notify a processor when the reduced-rate I^2S/TDM data channels are updated.
5. Option to run the bus data slots at the full, continuous audio rate (nominally 48 kHz) or a reduced rate. The rate can be reduced by:
   a. Skipping data slots for superframes that do not contain data (for example, only reduced sampling rate microphone nodes on the A^2B bus). This approach saves power by reducing the bus activity level but does not increase channel bandwidth on the bus. When the same A^2B data slots are shared between multiple I^2S/TDM channels in a node, the program cannot skip the A^2B data slots.
   b. Time-dividing bus data slots of a node into multiple I^2S/TDM channels and not skipping data slots for superframes. This approach is used if different types of subordinate nodes connecting on the same A^2B bus (for example, a multi-axis accelerometer node with a microphone or amp nodes on the same bus). The bus must run at the full-data rate to allow for A^2B data slot sharing. This approach provides for increased channel bandwidth on the bus by allowing reduced-rate subordinate nodes to time-multiplex I^2S/TDM data words over bus data slots.
      - Subordinate nodes running at ½ rate can use 2:1 time multiplexing (two I^2S/TDM channels in the same subordinate node alternate on one A^2B slot).
      - Subordinate nodes running at lower rates can use 4:1 time multiplexing (four I^2S/TDM channels in the same subordinate node alternate on one A^2B slot).
      - Time multiplexing of A^2B data slots beyond 4:1 is not supported.
      - Time multiplexing of A^2B data slots between nodes is not supported.
      - The bus must be run with A^2B data slots at the full, continuous audio rate for data slots to be shared.
      - The I^2S/TDM RX reduced rate data can be transmitted upstream or downstream.

I^2S Reduced Rate Restrictions

Observe the following general restrictions when using the I^2S reduced rate feature.

- Each subordinate node can only run at a single I^2S/TDM rate.
- Configure subordinate nodes running at a reduced I^2S/TDM rate for the I^2S/TDM RX data, not the I^2S/TDM TX data. This means that the reduced-rate subordinate nodes must have `A2B_I2SCFG.TXPINS` = 0.

- If A2B_I2SRRATE.RBUS is set and a reduced rate is configured (A2B_I2SRRATE.RRDIV > 1), subordinate nodes must have an A2B_I2SRATE.I2SRATE value of 0 (SFF x 1) or 3 (SFF / A2B_I2SRRATE.RRDIV).

Restrictions on Data Slot Sharing (A2B_I2SRATE.SHARE = 1)

Observe the following data slot sharing restrictions when using the I^{2}S reduced rate feature.

- The bus must run at the full-data rate (A2B_I2SRRATE.RBUS = 0) to allow for A^{2}B data slot sharing. A^{2}B data slot skipping cannot be used when the same A^{2}B data slots are shared between multiple I^{2}S/TDM channels in a node.
- Data slots on the A^{2}B bus produced by a reduced-rate subordinate node with A2B_I2SRATE.SHARE = 1 must be received from the A^{2}B bus by full- or increased-rate nodes.
- If the A2B_I2SRATE.SHARE bit is set in a reduced-rate subordinate node, the maximum synchronization offset is one superframe (A2B_I2SRRSOFFS.RRSOFFSET must be 0 or 1).

If the A2B_I2SRATE.SHARE bit is set in a reduced-rate subordinate node and there is no synchronization offset (A2B_I2SRRSOFFS.RRSOFFSET = 0), there is a further constraint on the node programming relative to *N* (the number of usable up and down slots). For example, if TDMS is the number of slots per frame on one pin of a reduced-rate subordinate node (which is 2, 4, 8, 16, or 32), *N* is calculated as shown in the following table:

I^{2}S/TDM Divide Ratio	*Number of Slots (N)*
2	TDMS >> 1
4	(TDMS >> 1) + (TDMS >> 2)
> 4	(TDMS >> 1) + (TDMS >> 2) + (TDMS >> 3)

If the reduced-rate subordinate node is generating downstream data slots (A2B_LDNSLOTS.DNMASKEN= 1), the same constraint applies to "A2B_LDNSLOTS + A2B_DNOFFSET".

If the reduced-rate subordinate node has the A2B_I2SCFG.RXPINS bits all set, "A2B_LUPSLOTS + A2B_UPOFFSET" must be ≤ 2*N*. Otherwise, "A2B_LUPSLOTS + A2B_UPOFFSET" must be ≤ I2 *N*.

Restrictions on Alternate BCLK Rate (A2B_I2SRATE.BCLKRATE)

Observe the following alternate BCLK rate restrictions when using the I^{2}S reduced rate feature.

- In a reduced-rate subordinate node, if the I^{2}S rate setting is SFF / 2 (A2B_I2SRATE.I2SRATE = 1), do not set the BCLK frequency to SYNC x 4096 (A2B_I2SRATE.BCLKRATE != 2).
- If the system-level reduced rate divisor is 1 (A2B_I2SRRATE.RRDIV = 1) and the I^{2}S rate setting is "SFF / A2B_I2SRRATE.RRDIV" (A2B_I2SRATE.I2SRATE = 3), do not set the BCLK frequency to "SYNC x 2048" (A2B_I2SRATE.BCLKRATE = 1) or "SYNC x 4096" (A2B_I2SRATE.BCLKRATE = 2).

- If the system-level reduced rate divisor is 2 (A2B_I2SRRATE.RRDIV = 2) and the I^{2}S rate setting is "SFF / A2B_I2SRRATE.RRDIV" (A2B_I2SRATE.I2SRATE = 3), do not set the BCLK frequency to "SYNC x 4096" (A2B_I2SRATE.BCLKRATE = 2).
- If the BCLK frequency is not determined by the value programmed in the A2B_I2SGCFG register (A2B_I2SRATE.BCLKRATE != 0) in a reduced rate subordinate node, the synchronization offset cannot exceed 1 superframe (A2B_I2SRRSOFFS.RRSOFFSET < 2).

Parallel A^{2}B Buses

Some systems require more bandwidth than a single A^{2}B bus can provide. Running two buses in parallel doubles the available bandwidth. The ***Parallel A^{2}B Buses*** figure illustrates how to connect the devices.

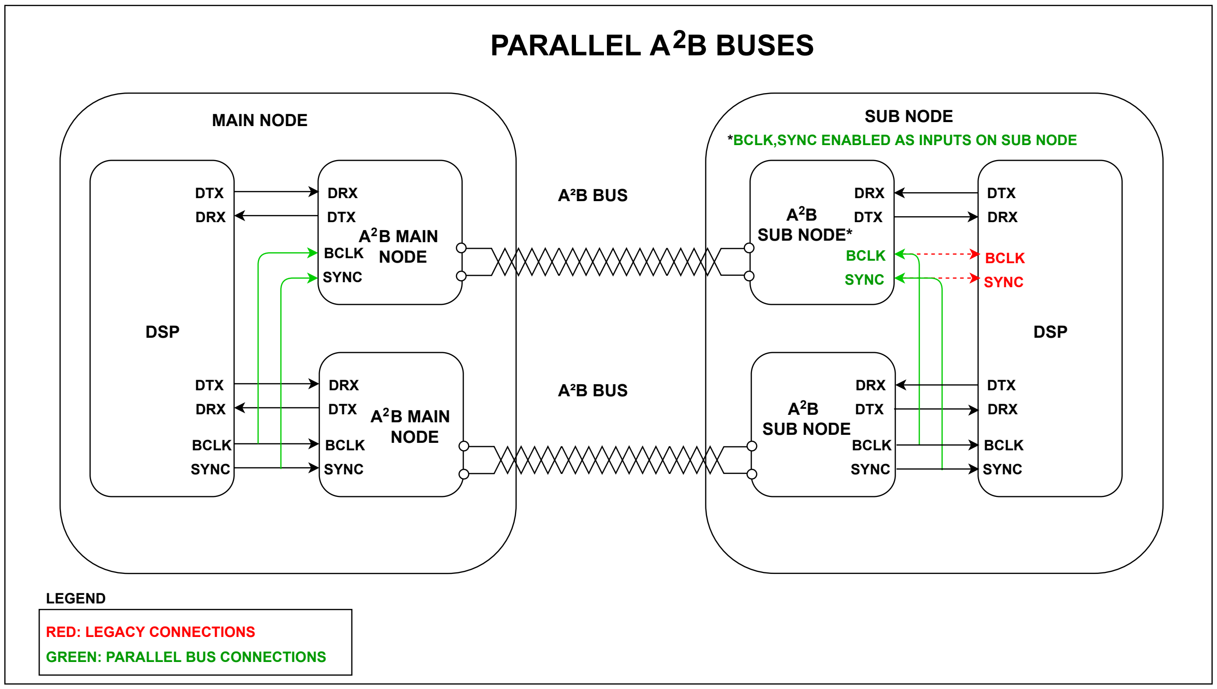


Figure 5-21: Parallel A^{2}B Buses

The parallel A^{2}B main nodes are I^{2}S targets and share a BCLK and SYNC signal. On the subordinate node, one of the A^{2}B transceivers is configured as an I^{2}S bus target. The other remains as I^{2}S bus controller (default). BCLK and SYNC signals are shared between the two A^{2}B bus subordinate nodes and the attached peripheral.

Writing 1 to the A2B_CONTROL.I2SMSINV bit configures an A^{2}B subordinate transceiver as an I^{2}S target.

The two A^{2}B subordinate nodes must be programmed with the same A2B_I2SGCFG and A2B_I2SRATE register values in this mode.

SYNC Pin Disable

When set (=1), the `A2B_I2SGCFG.SYNCDIS` bit disables the toggling of the SYNC signal. This feature can be used to hold the SYNC signal in an inactive state while the BCLK signal runs.

The `A2B_I2SGCFG.SYNCDIS` bit can only be set while the I^{2}S port is disabled. The port is disabled when:

- `A2B_I2SCFG.TXPINS`==0, and
- `A2B_I2SCFG.RXPINS`==0, and
- `A2B_PDMCTL.PDM1EN`==0, and
- `A2B_PDMCTL.PDM0EN`==0

Setting the `A2B_I2SGCFG.SYNCDIS` bit while the I^{2}S port is enabled does NOT disable the SYNC signal.

The SYNC signal is inactive while the BCLK signal runs when:

- the `A2B_I2SGCFG.SYNCDIS` bit is set (=1), and
- any of the TX/RX/PDM pins are enabled

The `A2B_I2SGCFG.SYNCDIS` bit can be cleared (=0) at any time. The SYNC signal edge starts running at the next active edge of the internal frame sync.

When the `A2B_I2SGCFG.SYNCDIS` bit is set:

- I^{2}S receive data is ignored (zeros put on bus)
- I^{2}S transmit data is populated with zeros

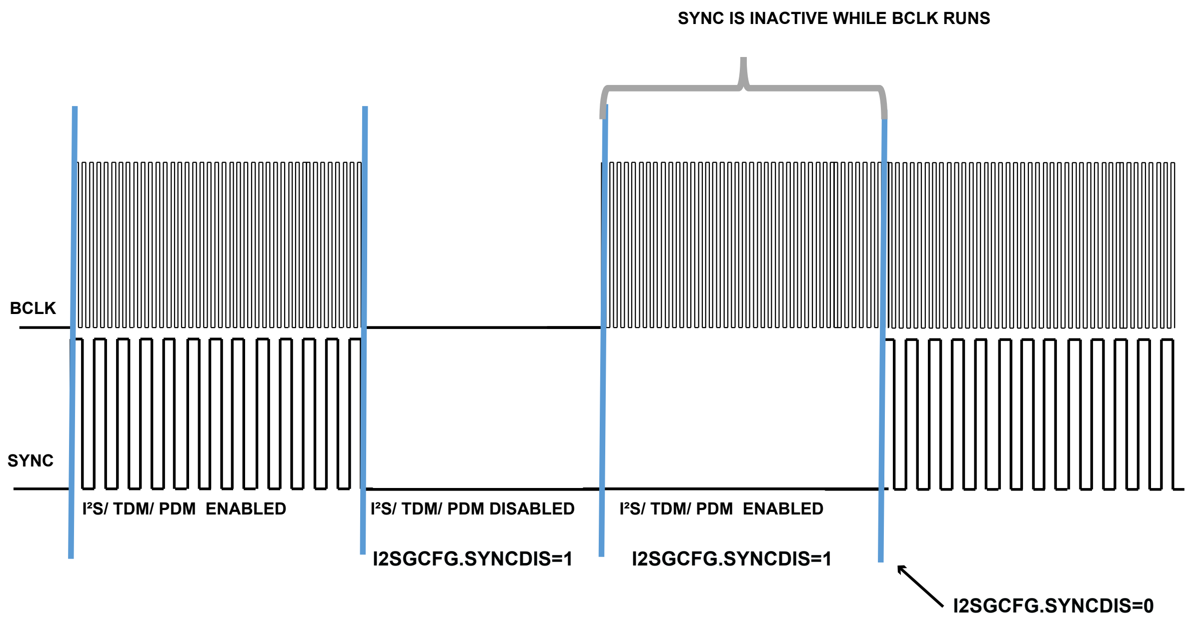


Figure 5-22: SYNC Waveform

Pulse-Density Modulation Interface (PDM)

Pulse-density modulation is used in sigma delta converters. The PDM format represents an over-sampled 1-bit sigma delta ADC signal before decimation. It is often used as the output format in digital microphones. The PDM block supports high dynamic range microphones with a high signal-to-noise ratio (SNR) and an extended maximum sound pressure level (SPL). The enhanced PDM block of the transceiver supports a lower noise floor than legacy A^{2}B transceivers. This provides for an SNR greater than 120 dB.

The PDM bit clock output frequency from the transceiver is 64x faster than the PDM audio sampling rate (typically, 3.072 MHz for 48 kHz PDM audio sampling).

The PDM block is configured using the PDM control (`A2B_PDMCTL`) register:

- When `A2B_PDMCTL.PDM0EN` = 1, the SIO0 pin is enabled to receive PDM data, and the BCLK pin is an output, typically producing a 3.072 MHz clock for the TDM2 setting. In this mode, the SIO0 pin data is not passed to the I^{2}S/TDM port. Similarly, the `A2B_PDMCTL.PDM1EN` bit controls PDM data reception on the SIO1 pin.
- Each PDM-enabled receive pin can receive up to two channels of audio data (stereo). The `A2B_PDMCTL.PDMxSLOTS` bits select whether the PDM signals on the DRX pins use one (mono) or two (stereo) channels. One of the channels is associated with the rising edge of the clock, and the other with the falling edge of the clock. Refer to PDM Sampling Edge of a Connected Microphone.

- The `A2B_PDMCTL.HPFEN` bit enables high pass filtering. The `A2B_PDMCTL2.HPFCORNER` bit selects the filter corner. The cutoff frequency of the high pass filter in the PDM block on the transceiver is programmable to 1 Hz, 60 Hz, 120 Hz, and 240 Hz. The high pass filter is a first order IIR filter.
- The `A2B_PDMCTL.PDMRATE` field controls the output rate of the PDM modulators. The transceiver is programmable for 1x, 1/2x, or 1/4x PDM sampling (48 kHz, 24 kHz, or 12 kHz typical) relative to the superframe rate (48 kHz typical). For 1/2x or 1/4x PDM sampling, synchronous data in an A^{2}B slot is duplicated in order to match the superframe rate.

  For example, if `A2B_I2SRATE` is set to the superframe frequency (SFF= 48 kHz), the ***PDM Output Rate*** table describes the PDM modulator behavior in terms of when samples are driven:

Table 5-9: PDM Output Rate

PDMRATE Field	PDM Output Rate
SFF (0b00)	Unique sample driven every frame
SFF/2 (0b01)	Each sample driven in two consecutive frames
SFF/4 (0b10)	Each sample driven in four consecutive frames

The PDM rate is independent on the `A2B_I2SRATE` setting. If `A2B_I2SRATE` = SFF/2 (24 KHz), the PDM modulator produces the samples at 24 KHz, 12 KHz, and 6 KHz for the respective setting of SFF, SFF/2, and SFF/4 in `A2B_PDMCTL.PDMRATE` . Even lower PDM sampling rates are possible when the reduced rate feature of the transceiver is used in combination with PDM output rate (for example, down to 375 Hz).

PDM Sampling Edge of a Connected Microphone

The pulse-density modulation (PDM) interface allows PDM input from two microphones to be time-multiplexed on a single data line using a single clock.

A PDM microphone encodes data such that the left channel is valid on the falling edge of the clock (CLK) signal and the right channel is valid on the rising edge of the CLK signal. After the DATA signal is driven during the appropriate half phase of the CLK signal, the microphone output is tristated. As such, two microphones (one set to the left channel and the other set to the right channel) can share a single DATA line (see the ***Stereo PDM Format*** figure).

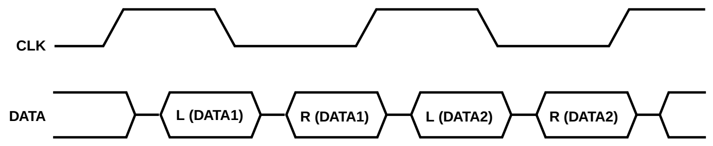


Figure 5-23: Stereo PDM Format

In the transceiver, the PDM block samples the microphone data on all 64 clock edges. When providing BCLK as PDMCLK, the transceiver must be programmed to a TDM mode that produces 64 BCLKs per frame (either the default TDM2/32 or TDM4/16 mode). The TDM settings do not affect the PDM block.

In the transceiver, the data sampled on the rising edge of BCLK is always the first channel. If `A2B_PDMCTL.PDM0SLOTS` = 1 or `A2B_PDMCTL.PDM1SLOTS` = 1, the first slot is associated with the rising edges of BCLK, and the second slot is associated with the falling edges of BCLK.

For example, two microphones are connected to each of the SIO0 and SIO1 pins of a sub node with the PDM0 and PDM1 slots configured as 2 slots (stereo). In this case, the PDM block samples 64-bit data each frame, converts it to 24-bit PCM data, and drives the converted output as follows:

- Right microphone data is sampled on the SIO0 pin on rising clock edges and driven in the first* transmit slot on the A^2B bus.
- Left microphone data is sampled on the SIO0 pin on falling clock edges and driven in the second* transmit slot on the A^2B bus.
- Right microphone data is sampled on the SIO1pin on rising clock edges and driven in the third* transmit slot on the A^2B bus.
- Left microphone data is sampled on the SIO1 pin on falling clock edges and driven in the fourth* transmit slot on the A^2B bus.

  Note that * is the actual slot number, based on the system slot configuration.

NOTE: PDM pins are always sampled with rising edge data first; therefore, the `A2B_I2SCFG.RXBCLKINV` and `A2B_I2SCFG.TXBCLKINV` clock inversion settings are ignored when the transceiver is configured in PDM mode.

PDM Enhancements

The default PDM functionality is fully backward-compatible with previous transceiver generations; however, there are several additional features which make the PDM interface more flexible.

PDM Clocking Options

The SIO0 and SIO1 input pins can be configured individually as PDM inputs. A PDMCLK signal running at 64 × f_{SYNCM} (3.072 MHz at 48 kHz f_{SYNCM}) is required to clock the PDM device. The transceivers allow either the PDMCLK/GPIO7 or BCLK pin to produce the required PDMCLK. PDMCLK on GPIO7 can be enabled by setting the `A2B_PDMCTL2.PDMALTCLK` bit.

If PDMCLK/GPIO7 is used instead of BCLK, the restriction limiting TDM mode operating to TDM2/32 or TDM4/16 is removed. The BCLK frequency can be set to a different frequency using the I^2S/TDM registers. In this case, PDMCLK/GPIO7 is used to capture PDM input on SIO0/SIO1.

NOTE: In a main node, BCLK is always an input; therefore, the clock output to PDM microphones connected to a main transceiver typically comes from PDMCLK/GPIO7.

BCLK and PDMCLK/GPIO7 can also be used concurrently to clock the PDM microphones at the same frequency and phase alignment, but with opposite polarity. This feature is accomplished by setting the `A2B_PDMCTL2.PDMALTCLK` bit.

The A2B_PDMCTL2 register controls whether the rising edge data or falling edge data is sampled first:

- When A2B_PDMCTL2.PDM0FFRST = 0 (default), the PDM0 data on SIO0 is sampled rising edge first. When A2B_PDMCTL2.PDM0FFRST = 1, it is sampled falling edge first.
- When A2B_PDMCTL2.PDM1FFRST = 0 (default), the PDM1 data on SIO1 is sampled rising edge first. When A2B_PDMCTL2.PDM1FFRST = 1, it is sampled falling edge first.

PDM Data Routing Options

The PDM interface can be used on main or subordinate transceivers. The PDM data received by the transceiver can then be sent to any node on the A^{2}B bus, sent out to the local I^{2}S port, or both. This option is configured using the A2B_PDMCTL2.PDMDEST field.

SPI Interface

The SPI interface is used to directly access the transceiver register space from a locally connected host and to remotely exchange SPI data over the A^{2}B bus between any nodes in the system. This protocol is referred to as ***SPI over distance***, where the exchanged SPI data is embedded within the data tunnel slots on the bus.

The SPI interface has the following features:

- Master/slave configurable
- Up to three slave selects
- Up to 12.288 MHz operation
- Programmable SPI clock polarity and SPI clock phase

The SPI block in the transceiver can use *data tunneling* to communicate SPI data in audio slots on the A^{2}B bus to remote A^{2}B subordinate nodes. Data tunnels are compatible with A^{2}B transceivers that do not feature an SPI interface, as these transceivers can be configured to simply pass the channels associated with the SPI data tunnels downstream or upstream (not to consume them). The SPI interface on A^{2}B transceivers drives and receives data in MSB first format.

The SPI interface supports four fundamental types of transactions: SPI local register access, SPI remote register access, SPI-to-SPI using the data tunnel, and remote I^{2}C peripheral access via SPI.

- SPI local register access
  - Local register read/write access
  - Data tunnel FIFO read
  - SPI bus FIFO read
  - SPI abort
  - SPI status read

- SPI remote register access
    - Remote register read/writes
- SPI over distance using the data tunnel
    - Full duplex read/writes
        - Up to 256 bytes of pipelined reads
    - Atomic SPI read/write
    - Bulk SPI-to-SPI writes
    - Extended full duplex
    - Extended bulk
    - A^2B subordinate-to-subordinate node SPI communication
- Remote I^2C peripheral access via SPI
    - SPI to I^2C read/writes

The SPI interface defaults to slave mode, but it can be programmed to act as the SPI master. This mode is required when the transceiver is the SPI data tunnel target. Each SPI port has 256 bytes of read data FIFO, 256 bytes of write data FIFO, and 32 bytes of bus data FIFO.

SPI Configuration

The SPI interface is configured using the `A2B_SPICFG` and `A2B_SPICKDIV` registers. Use the `A2B_SPICFG` register to configure a transceiver as an SPI master or slave and to enable or disable the SPI interface. Use the `A2B_SPICKDIV` register to configure the frequency of the clock generated on SCK when the A^2B transceiver is operating as an SPI master. Calculate the SPICLK frequency using either one of the following two formulas:

- SYSCLK / (CKDIV+1)
- PLLCLK / 2 x (CKDIV+1)

NOTE: SYSCLK = SYNC x 1024 and PLLCLK = SYNC x 2048

Pin functions for the SPI interface depend on whether the SPI is a master or a slave. For the SPI master configuration, the *SPI Master Connections* figure shows the signal connections and the *SPI Master Pin Functions* table shows the pin functions. For the SPI slave configuration, the *SPI Slave Connection* figure shows the signal connections and the *SPI Slave Pin Functions* table shows the pin functions.

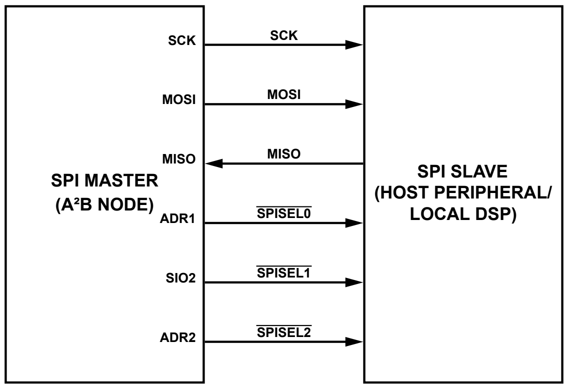


Figure 5-24: SPI Master Connections

Table 5-10: SPI Master Pin Functions

Pin	Direction	SPI Function	Description
SCK	Output	SCK	SPI Clock
MOSI	Output	MOSI	Master Output Data
MISO	Input	MISO	Master Input Data
ADR1	Output	$\overline{\text{SPISSEL0}}$	SPI Slave Select Output 0
SIO2	Output	$\overline{\text{SPISSEL1}}$	SPI Slave Select Output 1
ADR2	Output	$\overline{\text{SPISSEL2}}$	SPI Slave Select Output 2

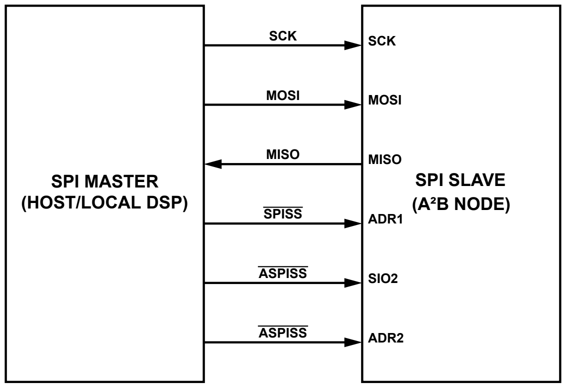


Figure 5-25: SPI Slave Connection

Table 5-11: SPI Slave Pin Functions

Pin	Direction	SPI Function	Description
SCK	Input	SCK	SPI Clock
MOSI	Input	MOSI	Slave Input Data
MISO	Output	MISO	Slave Output Data
ADR1	Input	$\overline{\text{SPISS}}$	Default SPI Slave Select Input

Table 5-11: SPI Slave Pin Functions (Continued)

Pin	Direction	SPI Function	Description
SIO2	Input	$\overline{\text{ASPISS}}$	Alternate SPI Slave Select Input
ADR2	Input	$\overline{\text{ASPISS}}$	Alternate SPI Slave Select Input

CAUTION: In SPI slave mode, ADR1 is the primary slave select line. ADR2 or SIO2 can also be used as an alternate slave select line, but it is available only for one type of SPI data tunnel transaction i.e. the register based full duplex transaction. See Full Duplex for details.

When the $\overline{\text{SPISS}}$ or $\overline{\text{ASPISS}}$ input signal transitions from high to low, an SPI transaction is initiated (the transition from 0 to 1 after reset is ignored). If an SPI interface is not required, the SPI port can be disabled by writing 0b10 to the `A2B_SPICFG.SPIMODE` field. SPI pins can be repurposed as GPIO/PWM pins when not used.

NOTE: See the *AD2437 A^2 B Transceiver Data Sheet* for more information.

The SPI interface supports four combinations of serial clock phase and polarity. These combinations are selected using the `A2B_SPICFG.SPI_CPOL` and `A2B_SPICFG.SPI_CPHA` bits. Clock transitions govern the shifting and sampling of data. Bits that are sampled on the rising edge of the clock cycle are shifted out on the falling edge of the clock cycle, and vice versa.

The clock polarity and the clock phase must be the same for the SPI master device and the SPI slave device involved in the communication link. Match the settings of the SPI master and SPI slave using the `A2B_SPICFG.SPI_CPOL` and `A2B_SPICFG.SPI_CPHA` bits.

The *SPI Transfer Protocol* figures demonstrate the two basic transfer formats as defined by the `A2B_SPICFG.SPI_CPHA` bit.

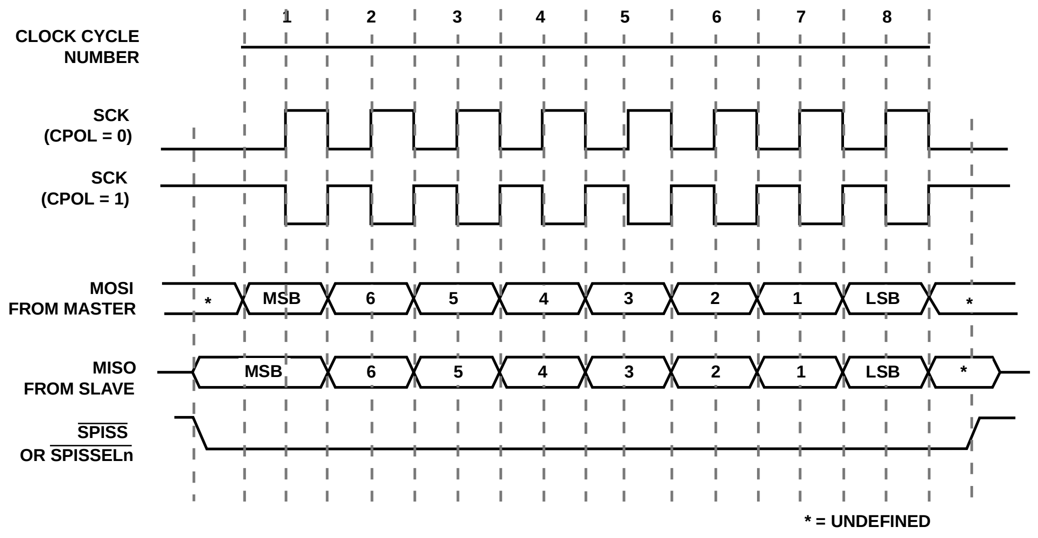


Figure 5-26: SPI Transfer Protocol for CPHA=0

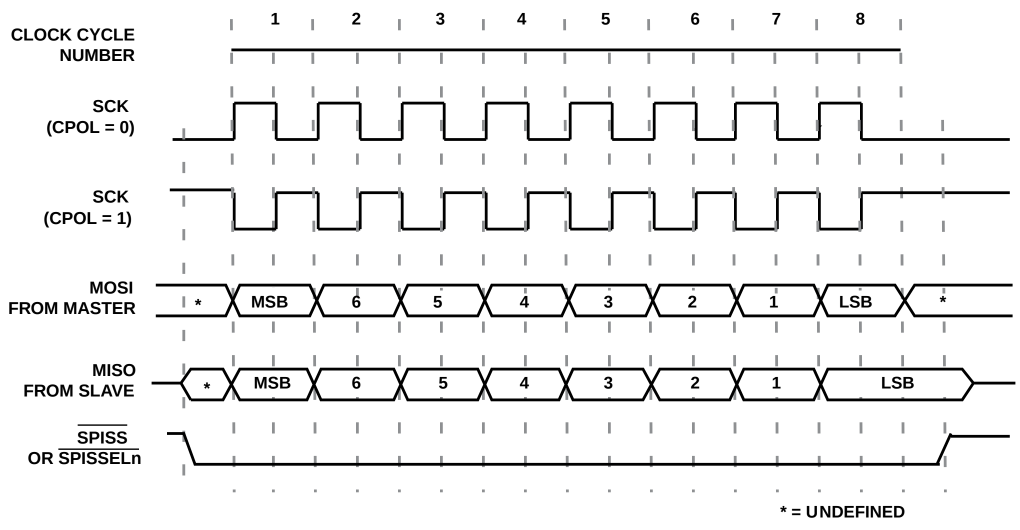


Figure 5-27: SPI Transfer Protocol for CPHA=1

SPI Programming Concepts

The SPI interface can be configured as an SPI slave (default) or tunnel responder (remote SPI master) by programming the `A2B_SPICFG.SPIMODE` bit. When configured as an SPI slave, the first byte of a transfer is the command byte which indicates the transaction type. Bytes that follow the command byte are decoded uniquely for each transaction type. The ***SPI Transaction Types*** table shows the supported transaction types and the associated command bytes used by the SPI module.

Table 5-12: SPI Transaction Types

Transaction Type	Transaction	Command Byte
Local	SPI Local Register Write	0x00
Local	SPI Local Register Read	0x01
Local	SPI Abort	0x0A
Local	SPI Bus FIFO Read	0x05
Local	SPI Status Read	0x04
Local	SPI Data Tunnel FIFO Read	0x0B
Remote (A^{2}B sub node access)	SPI Remote Register Write	0x02
Remote (A^{2}B sub node access)	SPI Remote Register Read Request	0xC0-0xDF
Remote Peripheral	SPI to I^{2}C Write	0x07
Remote Peripheral	SPI to I^{2}C Read Request	0x08
SPI Data Tunnel	SPI Atomic Write	0x0C
SPI Data Tunnel	SPI Atomic Read Request	0x0D

Table 5-12: SPI Transaction Types (Continued)

Transaction Type	Transaction	Command Byte
SPI Data Tunnel	Full Duplex	0x09
SPI Data Tunnel	Bulk SPI	0x06
SPI Data Tunnel	Extended Full Duplex	0x0E
SPI Data Tunnel	Extended Bulk	0x0F

The ***SPI Communication Block Diagram*** figure shows a system with SPI communication between the main and subordinate transceivers and SPI peripherals. Any node can be configured as the SPI master or an SPI slave in the system.

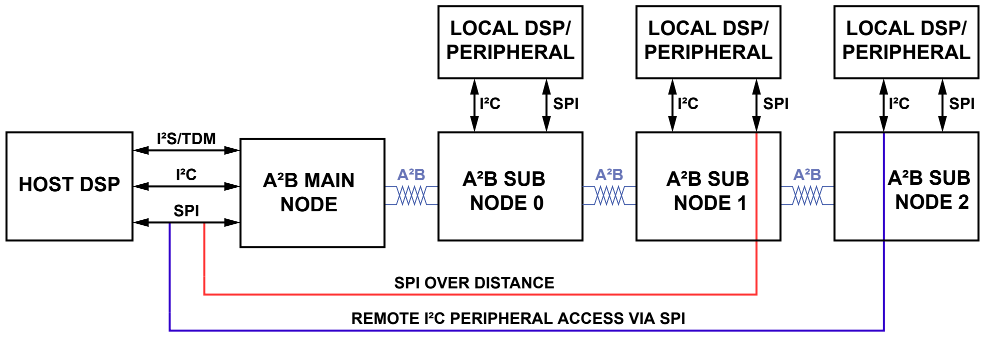


Figure 5-28: SPI Communication Block Diagram

- When configured as a tunnel responder, the SPI interface acts as the SPI master in SPI over distance transactions. Before being used as an SPI master, the `A2B_SPICKDIV` register must be programmed to generate the clock frequency for SCK. The `A2B_SPIPINCFG.SPIMSS0EN` - `A2B_SPIPINCFG.SPIMSS2EN` bits must be configured to enable the SPI slave select ($\overline{\text{SPISSEL}}$) output pins.
- When configured as an SPI slave, invalid command bytes are ignored and flagged (`A2B_SPIINT.BADCMD`=1).

SPI Register Access

Similar to the I^{2}C port, the SPI port can be used to configure and manage the A^{2}B bus. The A^{2}B host can use SPI accesses on the main node to access both the main transceiver register space and the register space of any discovered subordinate transceivers. In subordinate nodes, a locally-connected SPI host can access the sub transceiver register space, but it cannot access the register space of any other nodes in the system. SPI accesses to subordinate registers from the main node use the I^{2}C protocol hooks of the A^{2}B transceiver. This protocol supports the transfer of one byte in each direction per superframe in the synchronization control (downstream) and synchronization response (upstream) frames. The protocol also supports multiple attempts when bus conditions prevent the completion of the access. The SPI port must be configured in slave mode to support register accesses.

SPI Local Register Read

An SPI host can read the local A^{2}B node registers using a SPI local register read transaction.

As shown in the ***SPI Local Register Read Transaction*** figure, the SPI host drives the command byte (0x01), an 8-bit start address for the access (ADDR), and the 8-bit number of bytes to be read (N) on the MOSI line. The bytes of data read (N BYTES) are driven on the MISO line. Burst writes of up to 32 bytes are supported; N can be any value from 1 to 32.

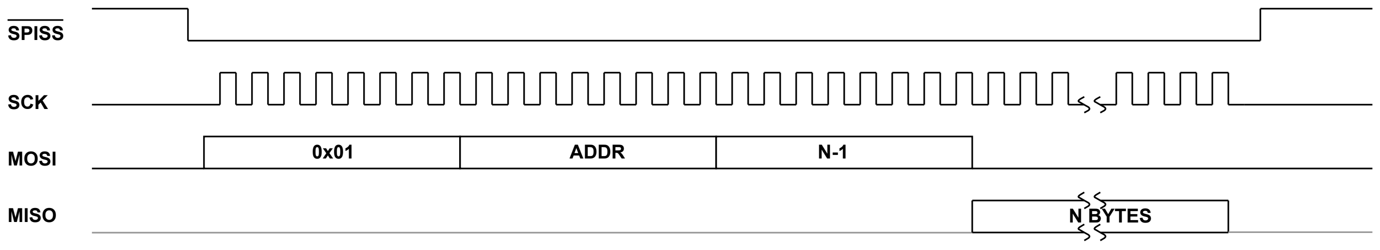


Figure 5-29: SPI Local Register Read Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a local register read.

A local register read occurs in the following sequence:

1. The host selects the local A^{2}B node as a SPI slave by transitioning the SPI slave select (ADR1/$\overline{\text{SPISS}}$) from high to low, initiating the transaction.
2. The host drives the MOSI pin with a continuous stream of data starting with the command byte (0x01), followed directly by the ADDR byte, and then by the N-1 byte (where N is the number of bytes to read).
3. The A^{2}B node drives N BYTES onto the MISO pin. The host must drive the clock for these N bytes.
4. The host transitions the SPI slave select from low to high, completing the transaction.

NOTE: Before starting a local register read, ensure that the SPI port is not busy (`A2B_SPISTAT.SPIBUSY` = 0). If the local register read is initiated while the SPI port is busy (`A2B_SPISTAT.SPIBUSY` = 1), an interrupt is generated and the `A2B_SPIINT.BADCMD` bit is set. Once initiated, local register reads are guaranteed to complete.

SPI Local Register Write

An SPI host can write the local A^{2}B transceiver registers using an SPI local register write transaction.

As shown in the ***SPI Local Register Write Transaction*** figure, the host drives the command byte (0x00), an 8-bit start address for the access (ADDR), and the N bytes of data (N BYTES) on the MOSI line. Burst writes of up to 32 bytes are supported; N can be any value from 1 to 32.

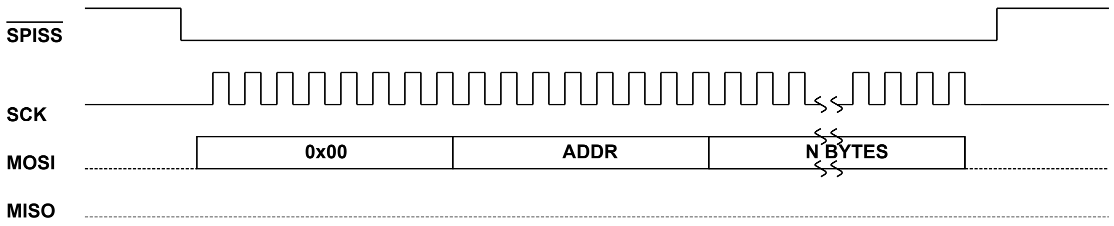


Figure 5-30: SPI Local Register Write Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a local register write.

A local register write occurs in the following sequence:

1. The host selects the local A^{2}B node as a SPI slave by transitioning the SPI slave select (ADR1/$\overline{\text{SPISS}}$) from high to low, initiating the transaction.
2. The host drives the MOSI pin with a continuous stream of data starting with the command byte (0x00), followed directly by the ADDR byte, and then N BYTES.
3. The host transitions the SPI slave select from low to high, completing the transaction.

NOTE: Before starting a local register write, ensure that the local A^{2}B transceiver is available by checking that the `A2B_SPISTAT.SPIBUSY` bit is cleared. If the local register write is initiated while the `A2B_SPISTAT.SPIBUSY` bit is set, an interrupt is generated and the `A2B_SPIINT.BADCMD` bit is set. Once initiated, local register writes are guaranteed to complete.

SPI Remote Register Write

The main node uses the SPI remote register write transaction to write registers in the A^{2}B sub node. It uses SCF/SRF frames; no A^{2}B slots need to be reserved for this SPI access. This transaction supports burst writes of up to 32 bytes.

As shown in the ***SPI Remote Register Write Transaction*** figure, the host drives the command byte (0x02), a node byte (NODE), an 8-bit start address for the access (ADDR), and N bytes of data (N BYTES) on the MOSI line. Burst writes of up to 32 bytes are supported; N can be any value from 1 to 32. The N BYTES byte consists of the data written to consecutive addresses starting from ADDR. The NODE byte includes a 4-bit field to indicate the target sub node (NSEL) and a broadcast bit (BCST) to set for broadcast writes to all nodes.

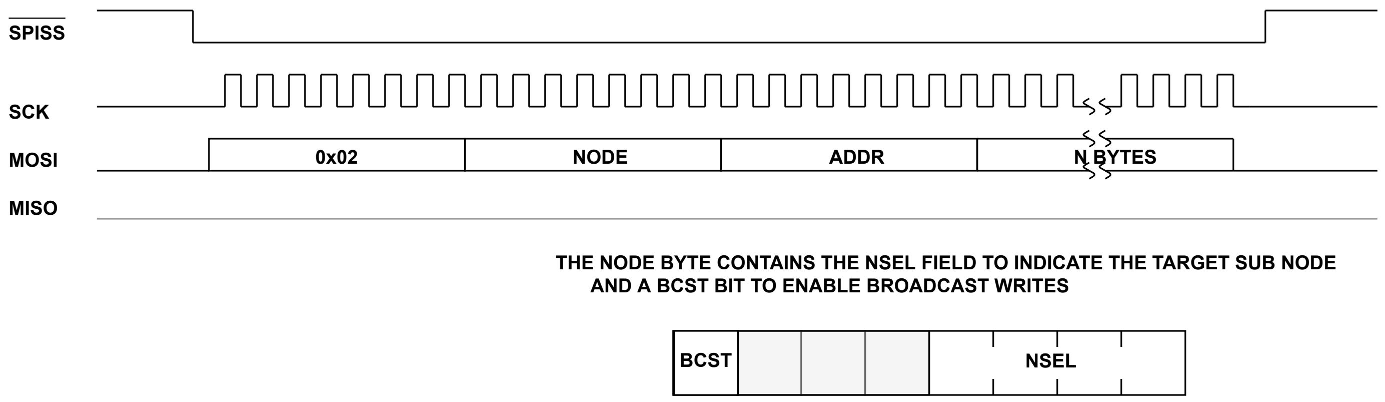

SPISS
SCK
MOSI
MISO
0x02
NODE
ADDR
N BYTES
THE NODE BYTE CONTAINS THE NSEL FIELD TO INDICATE THE TARGET SUB NODE AND A BCST BIT TO ENABLE BROADCAST WRITES
BCST
NSEL

Figure 5-31: SPI Remote Register Write Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a remote register write.

Use the following sequence to program an SPI remote register write:

- Select the A^{2}B node as the SPI slave by transitioning ADR1/$\overline{\text{SPISS}}$ from high to low
- Send the command, address, and length/data from the host to the SPI slave (local A^{2}B node)

  Read the `A2B_SPIINT` register to get the status of the transaction. If a remote register write fails to complete, the `A2B_SPIINT.SPIREGERR` bit is set.

NOTE: Before starting a remote register write request, make sure the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, a `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPISTAT.SPIBUSY` bit is set (=1) at the start of the write and remains asserted until the write completes. The `A2B_SPIINT.SPIREGERR` bit is set (=1) when a SPI remote register write fails to complete. In all the transactions, if the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the command is ignored.

SPI Remote Register Read Request

The A^{2}B main node uses the SPI remote register read request transaction to read registers in the A^{2}B sub node. It uses SCF/SRF frames; no A^{2}B slots need to be reserved for this SPI access. This transaction supports burst reads of up to 32 bytes.

The SPI remote register read request requires two transactions with a wait between them. The read request initiates the transfer of bytes from the sub node to the main node. The SPI status busy command can be used to poll the `A2B_SPISTAT.SPIBUSY` bit to determine when all the data is returned. The SPI Bus FIFO Read data command returns the read data.

Use the following sequence to program the SPI remote register read command:

1. Issue an SPI remote register read request
2. Wait for `A2B_SPISTAT.SPIBUSY`=0
3. Issue an SPI remote register read data (SPI Bus FIFO Read)

The A2B_SPIINT.FIFOUNF bit is set (=1) and zeros are returned when the read command tries to read more data from the FIFO than the preceding command.

As shown in the ***SPI Remote Register Read Request Transaction*** figure, the SPI host sends the command byte with the number of bytes to be read (LEN-1), a node byte (NODE), and an 8-bit start address for the access (ADDR) on the MOSI line. LEN bytes of data from the remote node are read at the local A^2B main transceiver.

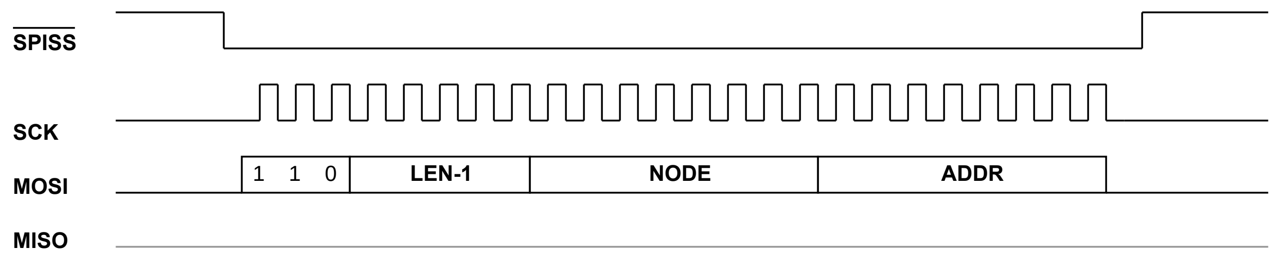


Figure 5-32: SPI Remote Register Read Request Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a remote register read request.

Use the following sequence to program an SPI remote register read request:

- Select the A^2B node as the SPI slave by transitioning ADR1/$\overline{\text{SPISS}}$ from high to low
- Send the command, address, and length/data from the host to the SPI slave (local A^2B node)

NOTE: Before starting a remote register read request, ensure that the A2B_SPISTAT.SPIBUSY bit is cleared. Otherwise, an A2B_SPIINT.BADCMD interrupt occurs. The A2B_SPIINT.SPIREGERR bit is set when an SPI remote register read request fails to complete.

SPI Bus FIFO Read

The SPI bus FIFO read transaction returns the read data requested by the subordinate register read request transaction and the SPI to I^2C read request transaction. The A2B_SPIINT.FIFOUNF bit is set and zeros are returned if the read transaction tries to read more data from the FIFO than the preceding read transaction fetched. The ***SPI Bus FIFO Read Transaction*** shows an SPI bus FIFO read transaction.

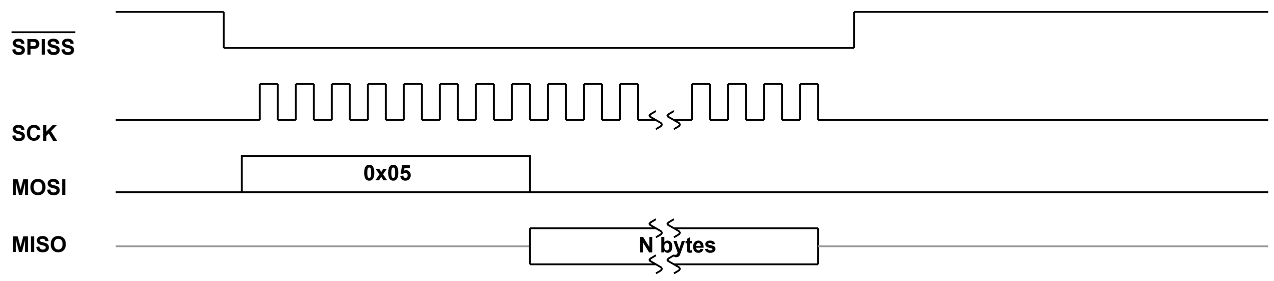


Figure 5-33: SPI Bus FIFO Read Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a bus FIFO read.

Use the following sequence to program an SPI bus FIFO read:

- Select the A^{2}B node as the SPI slave by transitioning ADR1/$\overline{\text{SPISS}}$ from high to low
- Send the command (0x05) from the host to the SPI slave (local A^{2}B node)

The A^{2}B node drives N BYTES on the MISO pin. The host must drive the clock for these N bytes. The host transitions the SPI slave select from low to high, completing the transaction.

SPI Status Read

The SPI status read transaction is used to read the A2B_SPISTAT register and to determine the status of ongoing transactions. This transaction is guaranteed to complete and never generates an interrupt.

There are several ways to obtain the state of the A2B_SPISTAT.SPIBUSY bit:

- Read the A2B_SPISTAT register using a normal SPI/I^{2}C register read.
  - With a normal register read, the A2B_SPISTAT register cannot be read with SPI when the A2B_SPISTAT.SPIBUSY bit is set.
  - Read the A2B_SPISTAT register using an SPI status read transaction as shown in the *SPI Status Read Transaction* figure.
- Enable a GPIO pin as an SPIBUSY signal using the A2B_SPIPINCFG.SPIGPIOSEL and A2B_SPIPINCFG.SPIGPIOEN fields.
- Enable the A2B_SPIINT.SPIDONE interrupt. An interrupt is generated when A2B_SPISTAT.SPIBUSY is cleared (=0).

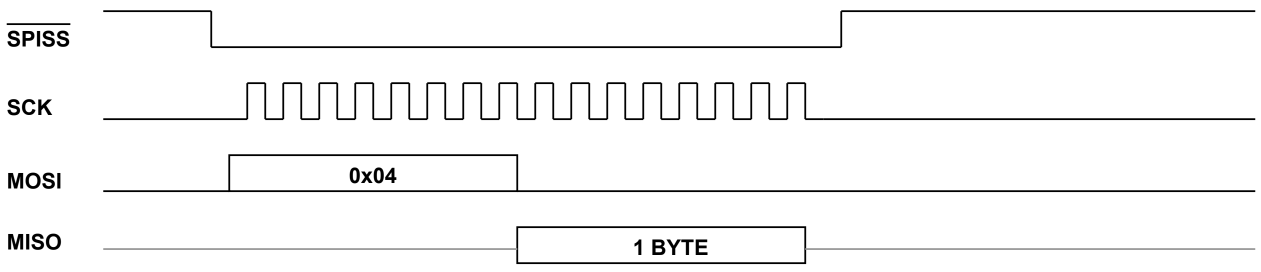


Figure 5-34: SPI Status Read Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate an SPI status read.

Remote I^{2}C Peripheral Access via SPI

An SPI to remote I^{2}C transaction allows the SPI host on the A^{2}B main node to read and write the I^{2}C peripherals on A^{2}B subordinate nodes. This transaction uses the remote I^{2}C protocol hooks built into the A^{2}B protocol to move the read and write data through the system as shown in the ***SPI to I^{2}C Transaction Block Diagram***.

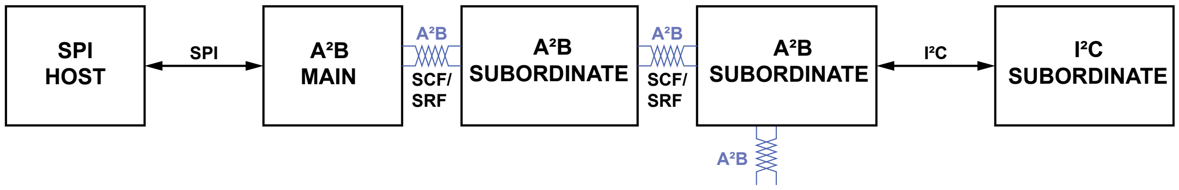


Figure 5-35: SPI to I^{2}C Transaction Block Diagram

The SPI to I^{2}C transaction uses A^{2}B I^{2}C protocol hooks to communicate with the I^{2}C subordinate node connected to the remote A^{2}B node. It does not use any A^{2}B data slots. The `A2B_SPIINT.SPII2CERR` interrupt is generated when an I^{2}C write or read operation to or from a remote I^{2}C subordinate fails. The protocol writes or reads one byte of data per superframe. A maximum of 32 bytes of data can be read/write to or from the I^{2}C subordinate node.

NOTE: 1. Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate SPI to I^{2}C transactions from the A^{2}B main node.

2. An SPI to I^{2}C remote peripheral access can only be initiated from a main node.

The SPI to I^{2}C communication sequence is:

1. The SPI host performs an SPI write to the main transceiver.
2. Data is transferred to the A^{2}B remote node through the SCF frames.
3. The remote A^{2}B node performs the I^{2}C write to the remote I^{2}C subordinate node.
4. If it is a read request, the I^{2}C subordinate node responds with the read data.
5. The data is transferred to the A^{2}B main transceiver node through the SRF frames. It is stored in the bus data FIFO.
6. The host waits until `A2B_SPISTAT.SPIBUSY`=0.
7. The SPI host initiates a bus FIFO read command at the main A^{2}B node to read the data in the bus data FIFO.

SPI to I^{2}C Write

The A^{2}B main node uses the SPI remote I^{2}C write transaction to program an I^{2}C peripheral connected to an A^{2}B subordinate node. It uses SCF/SRF frames; no A^{2}B slots need to be reserved for this SPI access. This transaction supports burst writes of up to 32 bytes.

As shown in the ***SPI Remote I^{2}C Write Transaction*** figure, the master node drives the command byte that identifies the transaction, one byte that selects the remote node (NODE), and bytes of data (N BYTES) on the MOSI line. The bytes of data are written to the master node only. N has a value from 1 to 32.

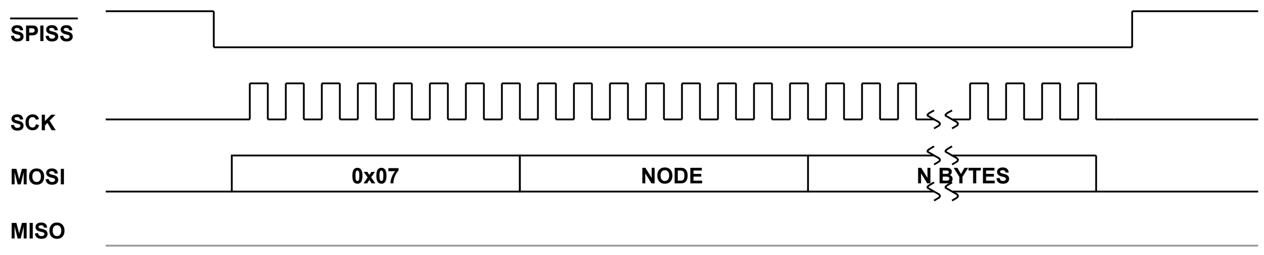


Figure 5-36: SPI Remote I2C Write Transaction

NOTE: Before starting an SPI remote I2C write transaction, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPISTAT.SPIBUSY` bit is set (=1) at the start of the write and remains asserted until the write completes. The `A2B_SPIINT.SPII2CERR` bit is set (=1) when a SPI remote I2C write fails to complete. In all the transactions, if the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the command is ignored.

SPI to I2C Read Request

The SPI remote I2C read request requires two transactions with a wait between them. The read request initiates the transfer of bytes from the subordinate node to the main node. The SPI status busy command is for polling the `A2B_SPISTAT.SPIBUSY` bit to determine when all data has been returned. The SPI bus FIFO read data command returns the read data.

Use the following sequence to program the SPI remote I2C read request:

1. Issue an SPI remote I2C read request.
2. Wait for `A2B_SPISTAT.SPIBUSY`=0.
3. Issue an SPI remote register read data (SPI bus FIFO read).

The `A2B_SPIINT.FIFOUNF` bit is set (=1) and zeros are returned if the read command tries to read more data from the FIFO than the preceding command. If the SPI remote I2C read request fails to complete, the `A2B_SPIINT.SPII2CERR` bit is set.

As shown in the ***SPI Remote I2C Read Request Transaction*** figure, the main node drives the command byte that identifies the transaction, a byte that selects the remote node (NODE), and a byte with LEN-1 on the MOSI line. The bytes of data (LEN) are only read from the subordinate peripheral. LEN has a value from 1 to 32.

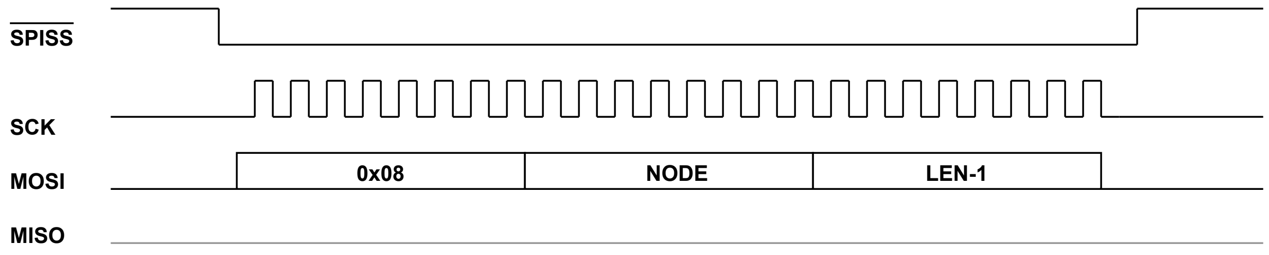


Figure 5-37: SPI Remote I^{2}C Read Request Transaction

NOTE: Before starting an SPI remote I^{2}C read request, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPII2CERR` bit is set when an SPI remote I^{2}C read request fails to complete.

Programming Sequence for SPI to I^{2}C Transactions

Use the following sequence to program an SPI to I^{2}C transaction:

1. Configure the A^{2}B main transceiver as the SPI slave and the A^{2}B subordinate transceiver as the I^{2}C main node (the I^{2}C peripheral is connected to the I^{2}C main node).
2. Program the I^{2}C subordinate device address in the chip register (`A2B_CHIP`) of the subordinate node.
3. Send the command bytes followed by N bytes of data or length for an SPI to I^{2}C transaction from the SPI host. The host initiates the data transfer through SCF/SRF frames.
4. If it is a read request, wait until `A2B_SPISTAT.SPIBUSY`=0 to initiate a bus data FIFO read from the host.
5. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored. (N must be less than 32 bytes).

SPI Over Distance

SPI over distance uses the A^{2}B protocol to allow an SPI transaction from any A^{2}B node in the system to target an SPI peripheral connected to any other transceiver in the system. An SPI host connected to a tunnel owner initiates SPI transactions that target the slave SPI interface on an A^{2}B transceiver. The A^{2}B transceiver relays the transfer to a remote node through the A^{2}B protocol to the node where the target peripheral is located.

The *SPI Over Distance Protocol* figure shows a block diagram of this transaction. The *SPI Over Distance Terminology* table provides more details on the elements in the figure.

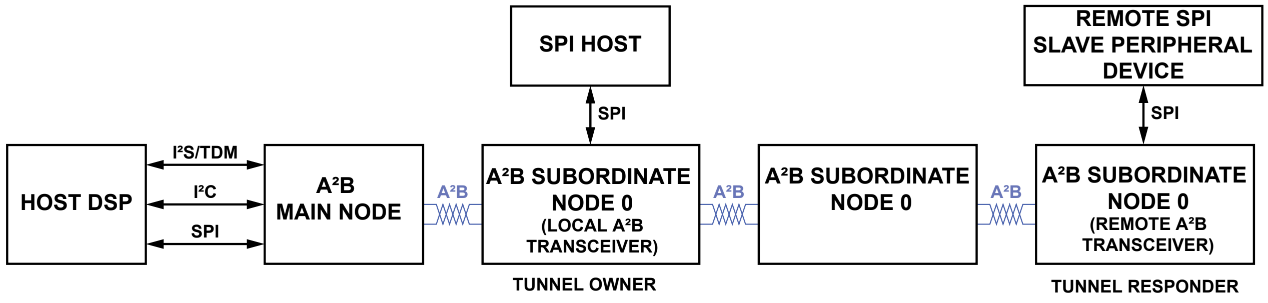

SPI HOST
REMOTE SPI SLAVE PERIPHERAL DEVICE
SPI
SPI
HOST DSP
I²S/TDM
I²C
SPI
A²B MAIN NODE
A²B
A²B SUBORDINATE NODE 0 (LOCAL A²B TRANSCEIVER)
A²B
A²B SUBORDINATE NODE 0
A²B
A²B SUBORDINATE NODE 0 (REMOTE A²B TRANSCEIVER)
TUNNEL OWNER
TUNNEL RESPONDER

Figure 5-38: SPI Over Distance Protocol

Table 5-13: SPI Over Distance Terminology

Device	Description	SPI Role
SPI Host	Initiator of SPI transaction	Master
Local A^{2}B Transceiver	Receiver of SPI transaction (tunnel owner) from locally-connected SPI host and initiator of SPI over distance transaction to a targeted subordinate node	Slave
Remote A^{2}B Transceiver	Receiver of SPI transaction over the A^{2}B bus (tunnel responder/target) and initiator of SPI transaction to locally-connected processor or peripheral	Master
Remote SPI Slave Peripheral Device	Receiver (processor or peripheral) of SPI transaction from locally-connected A^{2}B transceiver	Slave

SPI Data Tunnels

Data tunnels are dedicated upstream and downstream A^{2}B bus data slots used to transfer SPI data in SPI over distance communication. There are 2-12 slots for upstream transfer and 2-12 slots for downstream transfer. The transceiver initiating the SPI over distance transaction is the *tunnel owner*, and the transceiver receiving the SPI over distance transaction is the *tunnel responder*.

A data tunnel consists of the furthest upstream node, the furthest downstream node and the nodes in between. Tunnel nodes must enable tunnel capability by setting the tunnel enable bit (`A2B_DTCFG.DTEN=1`). The furthest upstream node always populates the downstream tunnel slots with either an empty packet (when idle) or the appropriate protocol packet (when active). The furthest downstream node similarly populates the upstream tunnel slots with an empty packet (when idle) or the appropriate protocol packet (when active). Nodes that do not have data tunnels enabled pass the downstream tunnel packets from the A-port to the B-port and the upstream tunnel packets from the B-port to the A-port without modifying content (the packets are treated like other bus slots that are not locally consumed).

RESTRICTION: A^{2}B transceivers that do not feature an SPI port must be configured to pass (and not consume) all upstream and downstream data slots configured as SPI data tunnels.

Data tunnels are configured using the following registers:

- A2B_DTCFG – enables or disables the data tunnels across the nodes. The furthest upstream node is indicated by A2B_DTCFG.DTFRST=1. The furthest downstream node is indicated by A2B_DTCFG.DTLAST=1.
- A2B_DTSLOTS – configures the number of SPI data tunnel upslots (A2B_DTSLOTS.DTUPSLOTS) and downslots (A2B_DTSLOTS.DTDNSLOTS).
- A2B_DTUPOFFS and A2B_DTDNOFFS – configures the location of data tunnels within a superframe. The upstream tunnel uses the number of slots as configured in the A2B_DTSLOTS.DTUPSLOTS bit field and starts at the slot selected by the A2B_DTUPOFFS.DTUPOFFS bit field. The downstream tunnel uses the number of slots as configured in the A2B_DTSLOTS.DTDNSLOTS bit field and starts at the slot selected by the A2B_DTSLOTS.DTUPSLOTS bit field. All nodes with a data tunnel enabled must use the same configuration for the A2B_DTUPOFFS.DTUPOFFS and A2B_DTSLOTS.DTUPSLOTS bit fields.

  The configurations in the slot and offset registers are ignored when data tunneling is disabled (A2B_DTCFG.DTEN=0).

Configuring SPI Data Tunnels

Use the following sequence before starting the first SPI data tunnel transaction:

1. Program the A2B_SPICFG register to configure the A^2B nodes as an SPI master (tunnel target) and an SPI slave (tunnel owner).
2. Program the A2B_SPICKDIV register in the SPI master (tunnel responder) to set the frequency of the SPI clock.
3. Enable the subordinate node select output (A2B_SPIPINCFG.SPIMSS0EN – A2B_SPIPINCFG.SPIMSS2EN).
4. Configure the data tunnels using registers: A2B_DTCFG, A2B_DTSLOTS, A2B_DTUPOFFS and A2B_DTDNOFFS. The A2B_DTCFG register determines the location of the nodes in the tunnel. The A2B_DTSLOTS, A2B_DTUPOFFS, and A2B_DTDNOFFS registers determine which A^2B data slots are allocated to the tunnel.

NOTE: 1. A minimum of two upstream and two downstream slots are required for SPI tunnels (even when the data is being written only). Setting one data tunnel slot is not valid.

2. A minimum of three slots are required when the data slot size (A2B_SLOTFMT register) is configured as 16 or 20 bits.

Tunnel Types

Based on the number of bytes in a tunnel, tunnels have two formats: standard and jumbo. The standard tunnel protocol supports up to 16 bytes of payload data with a 12-bit CRC and a 4-bit length field in a single superframe. The jumbo tunnel protocol supports from 17 to 48 bytes of payload data with a 16-bit CRC and an 8-bit length field in a single superframe.

The number of bytes in the tunnel is based on the number of slots configured in the `A2B_DTSLOTS` register. The data slot size is configured with the `A2B_SLOTFMT.UPSIZE` or `A2B_SLOTFMT.DNSIZE` bit fields. Tunnels must be a minimum of 6 bytes in size.

The *Standard Tunnel Size* table shows the number of bytes in a standard tunnel as a function of data slot size and the number of slots.

Table 5-14: Standard Tunnel Size

Number of Slots (DSLOTS)	Tunnel Size with Data Slot Size 16	Tunnel Size with Data Slot Size 20	Tunnel Size with Data Slot Size 24	Tunnel Size with Data Slot Size 32
2	N/A	N/A	6	8
3	6	7	9	12
4	8	10	12	16
5	10	12	16	N/A
6	12	16	N/A	N/A
7	14	N/A	N/A	N/A
8	16	N/A	N/A	N/A

The *Jumbo Tunnel Size* table shows the number of bytes in a jumbo tunnel as a function of data slot size and the number of slots.

Table 5-15: Jumbo Tunnel Size

Number of Slots (DSLOTS)	Tunnel Size with Data Slot Size 16	Tunnel Size with Data Slot Size 20	Tunnel Size with Data Slot Size 24	Tunnel Size with Data Slot Size 32
5	N/A	N/A	N/A	20
6	N/A	N/A	18	24
7	N/A	17	21	28
8	N/A	20	24	32
9	18	22	27	36
10	20	25	30	40
11	22	27	33	44
12	24	30	36	48

Tunnel Owner Packets

The *Tunnel Owner Packet* figure shows the structure of the tunnel slots for the tunnel owner on the A^{2}B bus.

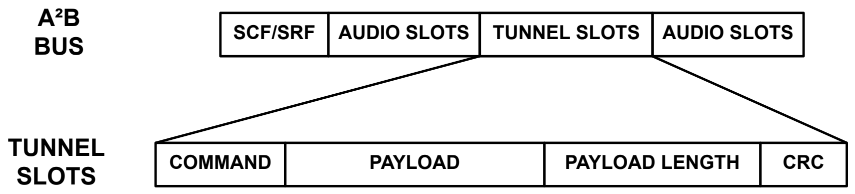


Figure 5-39: Tunnel Owner Packet

Tunnel owner packets consist of a command field that differentiates the type of transaction that occurs.

Jumbo Tunnel

The first superframe of data tunnels has an overhead of 6 bytes (3 command bytes + 1 byte payload length + 2 bytes of CRC) and the remaining superframes have an overhead of 4 bytes (1 command bytes + 1 byte payload length + 2 bytes of CRC) for all the data tunnel transactions except the atomic read request. For an atomic read request, there are 7 bytes of overhead (3 command bytes + 1 byte payload length + 2 bytes of CRC) in the first superframe and the subsequent superframes have an overhead of 4 bytes (1 command bytes + 1 byte payload length + 2 bytes of CRC).

Standard Tunnel

The first superframe of data tunnels has an overhead of 5 bytes (3 command bytes + 4 bits payload length + 12 bits of CRC) and the remaining superframes have an overhead of 3 bytes (1 command bytes + 4 bits payload length + 12 bits of CRC) for all the data tunnel transactions except atomic read request. For an atomic read request, there are 6 bytes of overhead (4 command bytes 4 bits payload length + 12 bits of CRC) in the first superframe and the subsequent superframes have an overhead of 3 bytes (1 command bytes + 4 bits payload length + 12 bits of CRC).

Tunnel Responder Packets

The *Tunnel Responder Packet* figure shows the structure of the tunnel slots for the tunnel owner on the A^{2}B bus.

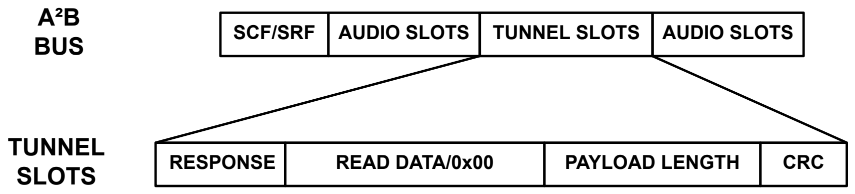


Figure 5-40: Tunnel Responder Packet

A tunnel responder packet consist of a response byte, read data, payload length, and a CRC. The response byte indicates whether the data is the read data from the last command or an acknowledgement of the last frame. When it is not read data, all tunnel data is populated with zeros and a payload length of 1. The payload length and CRC vary in size depending on whether the transaction protocol is standard or jumbo.

Jumbo Tunnel

Every superframe has an overhead of 4 bytes (1 response byte+1 byte payload length + 2 bytes of CRC)

Standard Tunnel

Every superframe has an overhead of 3 bytes (1 response byte+4 bits payload length + 12 bits of CRC)

Atomic SPI Transactions

Atomic SPI transactions allow a write or read initiated on any node in an A^{2}B system to occur at a peripheral on a different node. The `A2B_SPISTAT.SPIBUSY` bit must be cleared (=0) before starting an atomic SPI transaction. Otherwise, a BADCMD interrupt is generated.

Atomic SPI transactions include: SPI Atomic Write and SPI Atomic Read Request. The read or write transactions support a maximum of 256 bytes of data. For all atomic transaction types the `A2B_SPISTAT.SPIBUSY` is set when the first byte of the SPI transaction is received and cleared when the last byte of the SPI transaction is placed on the data tunnel. If a transaction is corrupted before reaching the remote A^{2}B node, the remote SPI transaction does not occur and an interrupt is issued.

The ***Atomic SPI Transaction Block Diagram*** shows an atomic SPI transaction. WD_FIFO refers to a write data FIFO and RD_FIFO refers to a read data FIFO.

The data received by the local A^{2}B node (tunnel owner) from SPI host is stored in the WD_FIFO and transferred to the WD_FIFO of the remote A^{2}B node (tunnel responder) though data tunnels.

The data received by the remote A^{2}B node (tunnel responder) from the SPI peripheral is stored in the RD_FIFO and transferred to the RD_FIFO of the local A^{2}B node (tunnel owner).

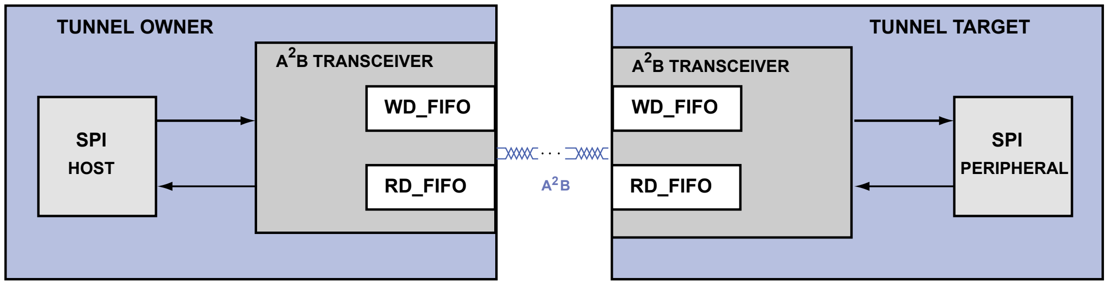


Figure 5-41: Atomic SPI Transaction Block Diagram

The sequence of an atomic SPI transaction is:

1. The bus host (or an externally-connected SPI host) performs an SPI write to the local A^{2}B transceiver (tunnel owner).
2. The local A^{2}B node waits until it receives the last byte of data before starting the data transfer to the remote A^{2}B node (tunnel responder).
3. The remote A^{2}B node waits until it receives the last byte of data into the WD_FIFO of the remote A^{2}B transceiver.

4. After receiving last byte of data, the remote A^{2}B transceiver starts writing data to the remote SPI peripheral.
5. The remote SPI peripheral starts responding with SPI read data to the RD_FIFO of the remote A^{2}B transceiver.
6. SPI read data is immediately transferred to the local A^{2}B node from the remote A^{2}B node through the data tunnel.
7. The SPI host should wait until the `A2B_SPISTAT.SPIBUSY` bit clears.
8. An SPI data FIFO read command should be issued at the SPI host to read the data from the RD_FIFO of the local A^{2}B transceiver.

NOTE: 1. The atomic SPI write transactions only performs the first 4 steps. The atomic SPI read transaction completes all the steps.

2. Only ADR1 can be used as $\overline{\text{SPISS}}$ from the SPI Host to local A^{2}B node for initiating atomic SPI transactions.

SPI Atomic Write

The atomic SPI write transaction writes the maximum of 256 bytes of data to a remote peripheral. As shown in the ***SPI Atomic Write*** figure, the local node transceiver drives the command byte that identifies the transaction, one byte that selects the remote node and has the sub node select information (NODE/SLAVE SELECT), and the bytes of data (N BYTES) on the MOSI line. N has a value from 1 to 256.

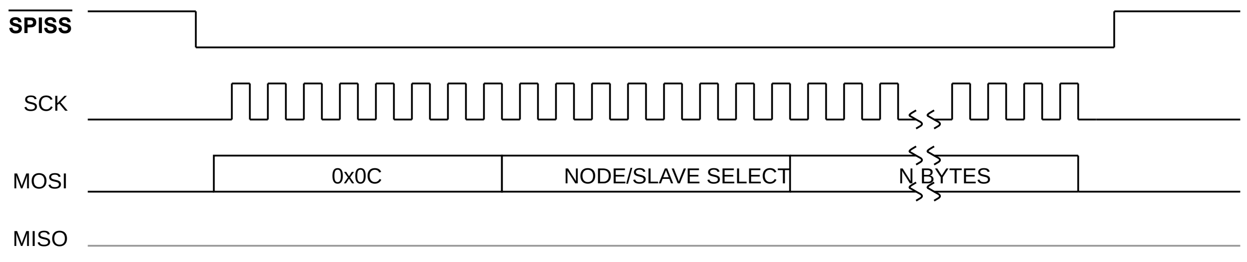


Figure 5-42: SPI Atomic Write

Table 5-16: SPI Data Tunnel Atomic Write

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	**MOSI**	**MISO**	**Byte**	**MOSI**	**MISO**
0	Atomic Write	0		Tunnel Delay Wait for the entire write command to reach the remote node	
1	Node/SS	0			
2	WR Byte [0]	0			
3	WR Byte [1]	0			

Table 5-16: SPI Data Tunnel Atomic Write (Continued)

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	**MOSI**	**MISO**	**Byte**	**MOSI**	**MISO**
...	...	0			
N+1	WR Byte [N-1]	0			
	Tunnel Delay Wait for the SPI write transaction to occur on the remote node and for the read data transport to the local node		0	WR Byte [0]	Ignored
			1	WR Byte [1]	Ignored
			...	...	Ignored
			N-1	WR Byte [N-1]	Ignored

NOTE: Before starting an SPI atomic write, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when an SPI atomic write fails to complete.

Slave Select/Node Byte

For all SPI over distance communication types, the command must be followed by node/SS byte. The ***SPI Slave Select/Node Byte*** figure shows the slave select/node byte.

The bit descriptions are:

- SSEL – indicates the slave select to target s shown below

SSEL	Target
2'b00	ADR1
2'b01	S102
2'b10	ADR2

- M/S – indicates if the target is the main node or sub node. If set (=1), it is the main node. If cleared (=0), it is the sub node.
- NODEID – the sub node id of the target when the target is a sub node (M/S = 0)

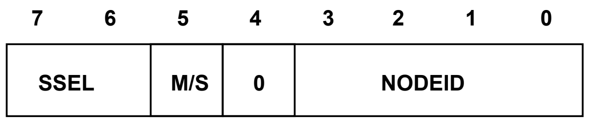


Figure 5-43: SPI Slave Select/Node Byte

SPI Atomic Read Request

The SPI atomic read request transaction reads a maximum of 256 bytes from a remote SPI slave. This transaction must be followed by a SPI Data Tunnel FIFO Read transaction to obtain the read data from the FIFO (RD_FIFO) at the local A^{2}B node.

Use the following sequence to program the SPI atomic read request:

1. Issue an SPI atomic read request.
2. Wait for `A2B_SPISTAT.SPIBUSY` = 0.
3. Issue an SPI data tunnel FIFO read.

The `A2B_SPIINT.FIFOUNF` bit is set (=1) and zeros are returned if the read command tries to read more data from the FIFO than the preceding command. If an SPI atomic read request fails to complete, the `A2B_SPIINT.SPIDTERR` bit is set.

As shown in the ***SPI Data Tunnel Atomic Read*** figure, the SPI host connected to the local A^{2}B node (tunnel owner) drives the command byte that identifies the transaction, one byte that selects the remote node and has the subordinate node select information (NODE/SLAVE SELECT), one byte with LEN-1, and the N bytes of write data (N BYTES) on the MOSI line. LEN has a value from 1 to 256. LEN read bytes are captured through the data tunnel. These N bytes include the commands sent to the remote peripheral to request the read data. N has a value from 0 to 255.

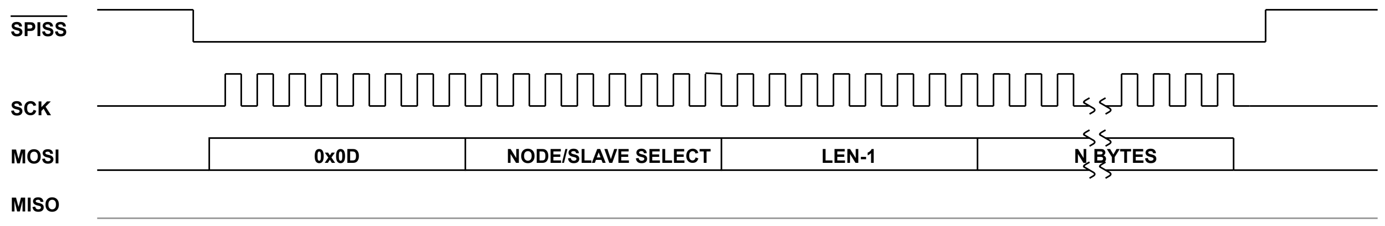


Figure 5-44: SPI Data Tunnel Atomic Read

The ***Tunnel Atomic SPI Read Request*** table details data flowing between the tunnel owner and the tunnel target. The table also identifies the specific tunnel delays associated with each transaction.

Table 5-17: Tunnel Atomic SPI Read Request

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	**MOSI**	**MISO**	**Byte**	**MOSI**	**MISO**
0	Atomic Read	0		Tunnel Delay Wait for the entire read command to reach the remote node	
1	Node/SS				
2	LEN-1	0			
3	WR Byte [0]	0			

Table 5-17: Tunnel Atomic SPI Read Request (Continued)

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	MOSI	MISO	Byte	MOSI	MISO
…	…	0			
N+1	WR Byte [N-1]	0			
	Tunnel Delay		0	WR Byte [0]	Ignored
	Wait for the SPI read transaction to occur on the remote node and read data transport to the local node		1	WR Byte [1]	Ignored
			2	…	Ignored
			N-1	WR Byte [N-1]	Ignored
			…	0	RD Byte[0]
			…	0	RD Byte[1]
			…	0	…
			N+LEN-1	0	RD Byte[LEN-1]
0	Data FIFO Read	0			
1	Ignored	RD Byte[0]			
2	Ignored	RD Byte[1]			
…	Ignored	…			
LEN	Ignored	RD Byte[LEN-1]			

NOTE: Before starting an SPI atomic read request, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when an SPI atomic read request fails to complete.

Programming Sequence for Atomic SPI Transactions

Use the following sequence to program an atomic SPI transaction:

1. Configure the tunnels. See SPI Data Tunnels.
2. Send the command bytes followed by N bytes of data or length for an atomic SPI transaction from the host to initiate the data transfer.
3. If it is a read request, wait until `A2B_SPISTAT.SPIBUSY`=0 to initiate a data tunnel FIFO read command from the host to get the read data.
4. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored.

SPI Data Tunnel FIFO Read

The SPI data tunnel FIFO read transaction returns the read data requested by atomic read transactions or the last full duplex transaction. If this read transaction is used to read more data from the FIFO than the preceding read transaction fetched, then 0x00 is returned. An interrupt is generated and the `A2B_SPIINT.FIFOUNF` bit is set.

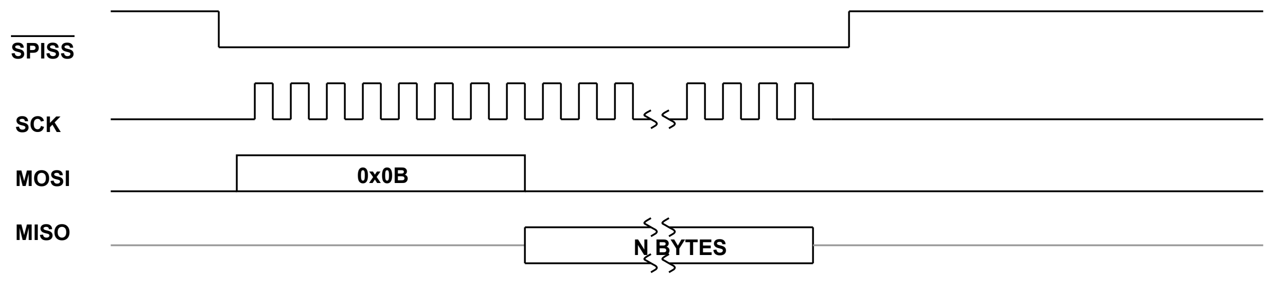


Figure 5-45: SPI Data Tunnel FIFO Read Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ to initiate a data tunnel FIFO read.

Use the following sequence to program an SPI data tunnel FIFO read:

1. Select the A^{2}B node as the SPI slave by transitioning the ADR1/$\overline{\text{SPISS}}$ signal from high to low.
2. Send the command bytes (0x0B) from the host to the SPI slave (local A^{2}B node). The A^{2}B node drives N BYTES onto the MISO pin. The host must drive the clock for N bytes.

Full Duplex

The full duplex SPI transaction has two packet flow structures.

1. Full duplex SPI transaction – slave select 0 (ADR1) selects the SPI slave from the host. The host sends the command bytes followed by the data.
2. Full duplex register-based SPI transaction – the A^{2}B node is selected by either the ADR2 or SIO2 pin. If `A2B_SPICFG.SPIFDSS` =1 or 2, the host use ADR2/SIO2, respectively, to initiate a full duplex command without any command bytes. The `A2B_SPIFDTARG` and `A2B_SPIFDSIZE` registers are used as the arguments for the full duplex transaction over the bus when `A2B_SPICFG.SPIFDSS` is non-zero. The host sends only write data without any commands.

The SPI host initiates the full duplex SPI transaction by sending the command bytes and data as shown in the *Full Duplex Transaction* figure. The delay introduced by the A^{2}B protocol introduces a one SPI transaction delay in the return of read data to the SPI master on MISO (the read data returned during the current transaction is the result of the previous SPI transaction).

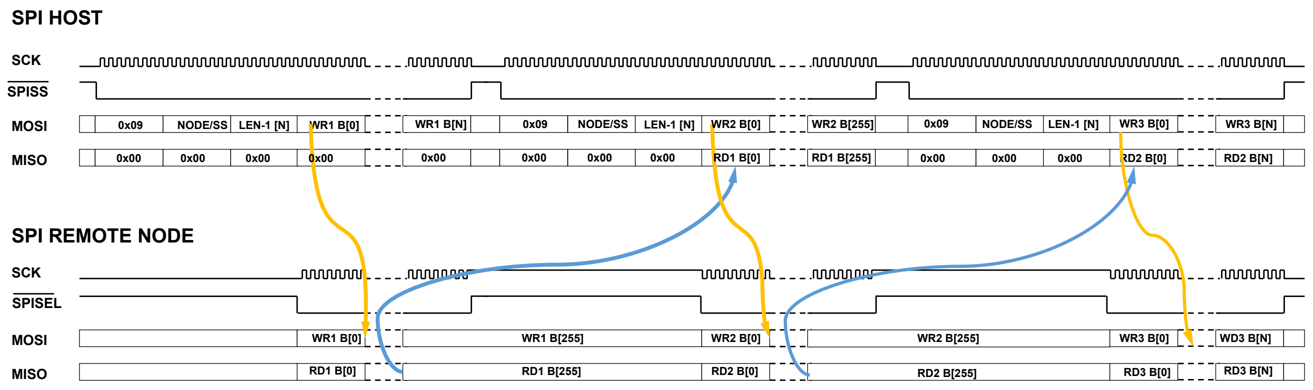


Figure 5-46: Full Duplex Transaction

If the A2B_SPICFG.ENFDCS bit is set (=1) in the remote node, full duplex SPI transactions do not wait for all the payload data to reach the remote node before starting the remote transaction. This configuration reduces the system latency of the transaction. If the A2B_SPICFG.ENFDCS bit is not set in the remote node, full duplex SPI transactions wait for all the payload data to reach the remote node before starting the remote transaction. The A2B_SPIFDSIZE register configures the transaction size for a register-based full duplex SPI transaction.

The A2B_SPISTAT.SPIBUSY bit must be cleared (=0) before starting a full duplex SPI transaction, or the A2B_SPIINT.BADCMD bit is set and an interrupt is generated. The first full duplex transaction does not have read data available from the previous transaction and returns zeros for all bytes. The first full duplex transaction is defined as any full duplex transaction that was not preceded by a full duplex transaction (for example, the first transaction on an SPI port or when the prior transaction was a non-full duplex SPI read transaction).

The sequence of a full duplex SPI transaction is:

1. The SPI host performs an SPI write to the local A^{2}B node.
2. The SPI write data is transferred immediately to a A^{2}B remote node through the data tunnels.
3. The remote A^{2}B node waits to receive the last SPI write data byte or does not wait (based on the A2B_SPICFG.SPIFDSS bit).
4. The remote A^{2}B node sends the SPI write data to the remote SPI slave.
5. The remote SPI slave instantly responds with SPI read data.
6. The remote A^{2}B node immediately transfers the SPI read data to the local A^{2}B node through data tunnels.
7. The SPI host receives this previous SPI read data when it performs the next full duplex SPI write.

NOTE: 1. A data FIFO read can be issued at the local A^{2}B node to receive the last full duplex transaction read data.

2. If the sub-sequent full duplex transactions are of different transaction lengths, use a data FIFO read in between to receive data from the previous transaction at the local A^{2}B node.

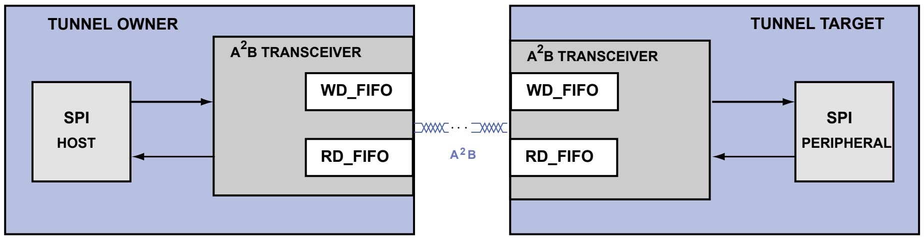


Figure 5-47: SPI Full Duplex Block Diagram

As shown in the ***SPI Full Duplex Access Transaction*** figure, the main node drives the command byte that identifies the transaction, one byte that selects the remote node and has the subordinate node select information (NODE/ SLAVE SELECT), one byte with LEN-1, and bytes of data (LEN BYTES) on the MOSI line. LEN has a value from 1 to 256.

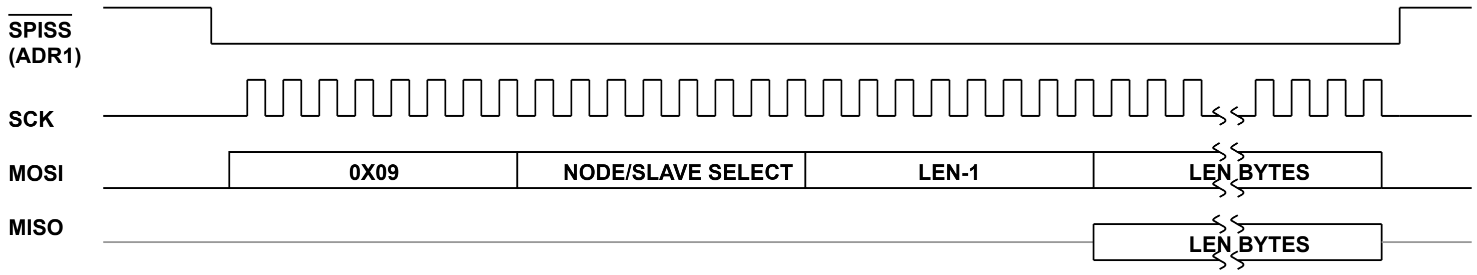


Figure 5-48: SPI Full Duplex Access Transaction

Table 5-18: SPI Full Duplex

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	MOSI	MISO	Byte	MOSI	MISO
0	Full Duplex	0		Tunnel Delay Remove transfer starts when the first bytes are received at the remote node	
1	Node/SS	0			
2	LEN-1	0			
3	WR2 Byte [0]	RD1 Byte [0]			
4	WR2 Byte [1]	RD1 Byte [1]	0	WR2 Byte [0]	RD2 Byte [0]
…	…	…	1	WR2 Byte [1]	RD2 Byte [1]
LEN+1	WR2 Byte [LEN-1]	RD1 Byte [LEN-1]	2	WR2 Byte [2]	RD2 Byte [2]
			…	…	…
			LEN-1	WR2 Byte [LEN-1]	RD2 Byte [LEN-1]

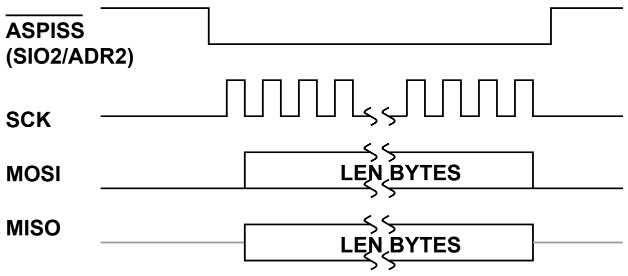


Figure 5-49: SPI Full Duplex Register Based

Table 5-19: SPI Full Duplex - Register Based

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	MOSI	MISO	Byte	MOSI	MISO
0	WR2 Byte [0]	RD1 Byte [0]		Tunnel Delay Remove transfer starts when the first bytes are received at the remote node	
1	WR2 Byte [1]	RD1 Byte [1]			
...	...	...			
FDLEN+1	WR2 Byte [FDLEN-1]	RD1 Byte [FDLEN-1]			
			0	WR2 Byte [0]	RD2 Byte [0]
			1	WR2 Byte [1]	RD2 Byte [1]
			2	WR2 Byte [2]	RD2 Byte [2]
			...	...	...
			FDLEN-1	WR2 Byte [FDLEN-1]	RD2 Byte [FDLEN-1]

There is no restriction on the SPI clock frequency at the tunnel owner and tunnel responder. They are not required to be the same speed. If the speeds differ, there are no resulting data overflow or underflow errors.

NOTE: Before starting a full duplex transaction, make sure the `A2B_SPISTAT.SPIBUSY` bit is clear or an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when a full duplex transaction fails to complete.

Programming Sequence for Full Duplex SPI Transactions

Use the following sequence to program a full duplex SPI transaction:

1. Configure the tunnels. See SPI Data Tunnels.
2. To initiate the data transfer, send the command bytes followed by N bytes of data for a full duplex SPI transaction from the host. For register-based full duplex SPI transaction, sending data is enough.
3. For the full duplexed register-based transaction, program the `A2B_SPICFG.SPIFDSS` bit to select between ADR2 and SIO2.
4. For full duplex register-based transactions, program the `A2B_SPIFDSIZE` and `A2B_SPIFDTARG` registers .

5. Read data of the previous full duplex transaction is received without issuing any command.
6. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored. N must be less than 256 bytes.
7. To start the next full duplex transaction check for the `A2B_SPISTAT.SPIBUSY` bit.

Extended Full Duplex

The A^{2}B SPI protocol limits full duplex SPI transactions to 256 bytes. Some SPI peripherals require larger transactions. The full duplex extended command makes multiple SPI transactions between the SPI host and local A^{2}B node behave as a single transaction with the clock stretching at byte boundaries to the remote SPI slave. The full duplex extended command behaves the same as the standard full duplex command, except that the SPI slave select remains asserted on the remote node. A standard full duplex command is issued to end the transfer. The *Extended Full Duplex Transaction* figure shows three SPI host transfers being translated into a single remote SPI transfer.

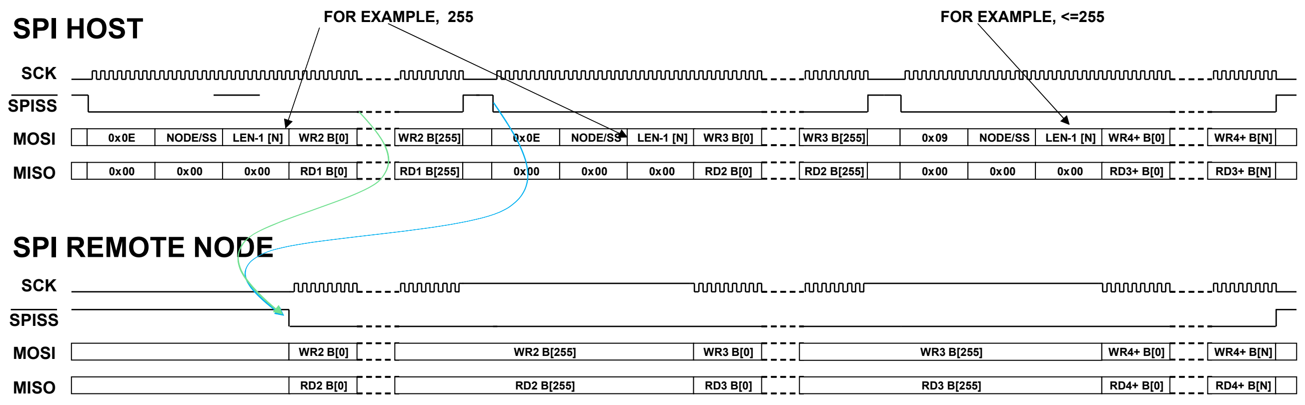


Figure 5-50: Extended Full Duplex Transaction

Tunnel owners with open extended transactions can interleave other transactions with the ongoing extended transaction. The following is a list of transactions that are legal to interleave:

- Extended full duplex to the same tunnel responder
- Legacy full duplex to the same tunnel responder
- Local register access
- Subordinate node register access when the tunnel owner is the A^{2}B main node
- Remote I^{2}C access when the tunnel owner is the A^{2}B main node

If the host initiates any other data tunnel transactions (except full duplex) to the same tunnel responder, the transaction is aborted on the tunnel owner. The extended full duplex transaction on the tunnel responder closes ($\overline{\text{SPISELn}}$ signal high).

If the host initiates any data tunnel transactions to a different tunnel responder, the transaction is aborted on the tunnel owner. The extended full duplex transaction on the tunnel responder stays open ($\overline{\text{SPISELn}}$ signal low).

NOTE: Before starting an extended full duplex transaction, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when an extended full duplex transaction fails to complete.

Programming Sequence for Extended Full Duplex SPI Transactions

Use the following sequence to program an extended full duplex SPI transaction:

1. Configure the tunnels. See SPI Data Tunnels.
2. To initiate the data transfer, send the command bytes followed by N (≤256) bytes of data for extended full duplex SPI transaction from the host.
3. Read data of the previous extended full duplex transaction is received without issuing any command.
4. Start a normal full duplex transaction to end the extended full duplex transaction.
5. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored.

NOTE: A data FIFO read can be issued at the local A^{2}B node to receive the last full duplex transaction read data.

Bulk SPI

Bulk SPI transactions are similar to full duplex transactions without the read. The SPI host initiates the bulk SPI transaction by sending the command bytes and data. The `A2B_SPISTAT.SPIBUSY` must be cleared (=0) before starting a bulk SPI write. Otherwise, an `A2B_SPIINT.BADCMD` interrupt is generated. Bulk SPI transactions do not wait for all of the payload data to reach the remote node before starting the remote transaction. This configuration reduces the system latency of the transaction.

The sequence of a bulk SPI transaction is:

1. The SPI host performs an SPI write to a local A^{2}B node (256 bytes maximum).
2. SPI write data is immediately transferred to the A^{2}B remote node through the data tunnels.
3. The remote A^{2}B node immediately sends SPI write data to a remote SPI slave.

As shown in the ***SPI Data Tunnel Bulk Write Transaction*** figure, the main node drives the command byte that identifies the transaction, one byte that selects the remote node and has the subordinate node select information (NODE/SLAVE SELECT), one byte with LEN-1, and bytes of data (LEN BYTES) on the MOSI line. LEN has a value from 1 to 256.

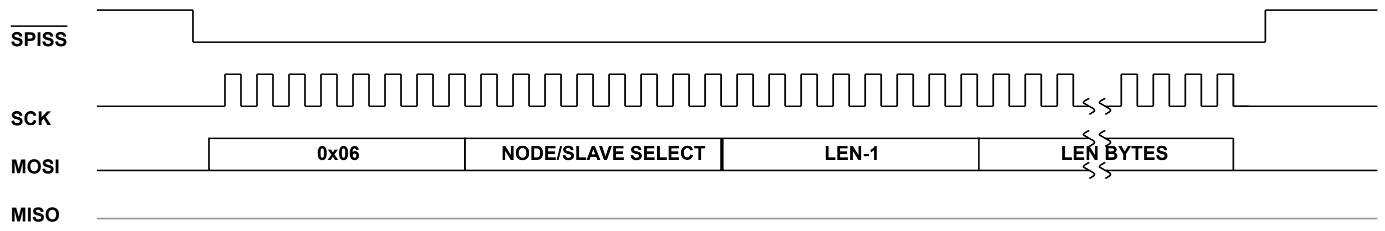


Figure 5-51: SPI Data Tunnel Bulk Write Transaction

NOTE: Only ADR1 can be used as $\overline{\text{SPISS}}$ from the SPI host to the local A^{2}B node for initiating a bulk SPI write.

Table 5-20: SPI Data Tunnel Bulk Write

	Local Node (SPI Host to Tunnel Owner)			Remote Node (Tunnel Target to Remote Peripheral)	
Byte	MOSI	MISO	Byte	MOSI	MISO
0	Bulk	0		Tunnel Delay Remove transfer starts when the first bytes are received at the remote node	
1	Node/SS	0			
2	LEN-1	0			
3	WR Byte [0]	0			
4	WR Byte [1]	0	0	WR Byte [0]	Ignored
...	...	0	1	WR Byte [1]	Ignored
LEN+2	WR Byte [LEN-1]	0	2	WR Byte [2]	Ignored
			...	...	Ignored
			LEN-1	WR2 Byte [LEN-1]	Ignored

NOTE: Before starting a bulk SPI transaction, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when a bulk SPI transaction fails to complete.

Programming Sequence for Bulk SPI Transactions

Use the following sequence to program a bulk SPI transaction:

1. Configure the tunnels. See SPI Data Tunnels.
2. To initiate the data transfer, send the command bytes followed by N bytes of data for a bulk SPI transaction from the host.
3. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored.

   N must be less than 256 bytes.

Extended Bulk

The A^2B SPI protocol limits bulk write SPI transactions to 256 bytes. Some SPI peripherals require larger transactions. The bulk write extended command makes multiple SPI transaction between the SPI host and local A^2B node behave as a single transaction with the clock stretching at byte boundaries to the remote SPI slave. The bulk write extended command behaves the same as the standard bulk write command, except the SPI slave select remains asserted on the remote node. A standard bulk write command is issued to end the transfer. The *Extended Bulk Transaction* figure shows three SPI host transfers being translated into a single remote SPI transfer.

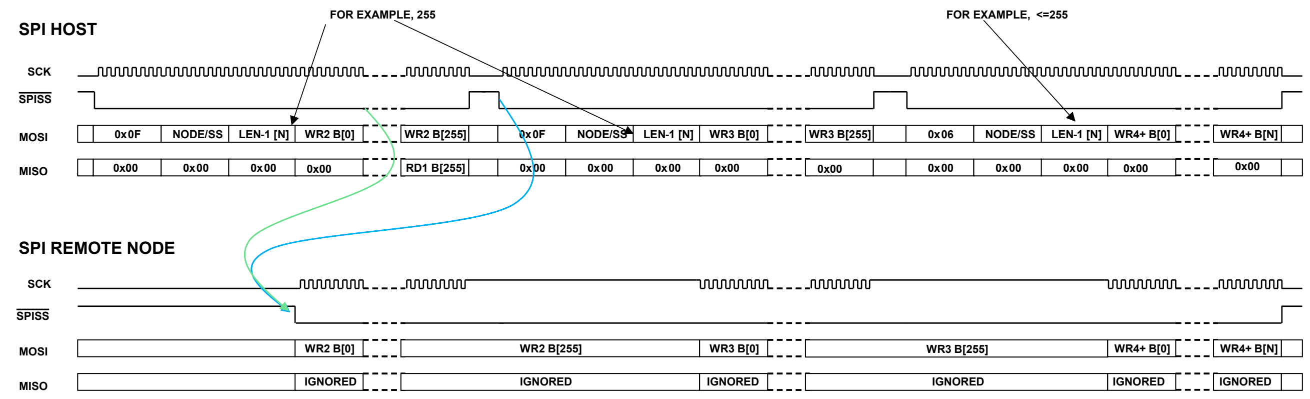


Figure 5-52: Extended Bulk Transaction

Tunnel owners with open extended transactions can interleave other transactions with the ongoing extended transaction. The following is a list of transactions that are legal to interleave:

- Extended bulk write to the same tunnel responder
- Legacy bulk to the same tunnel responder
- Local register access
- Subordinate node register access when the tunnel owner is the main node
- Remote I^2C access when the tunnel owner is the main node

If the host initiates any other data tunnel transactions (except bulk SPI write or extended bulk) to the same tunnel responder, the transaction is aborted on the tunnel owner. The extended bulk transaction on the tunnel responder closes (SPISELn signal high).

If the host initiates any data tunnel transactions to a different tunnel responder, then the transaction is aborted on the tunnel owner. The extended bulk transaction on the tunnel responder stays open (SPISELn signal low).

NOTE: Before starting an extended bulk transaction, ensure that the `A2B_SPISTAT.SPIBUSY` bit is cleared. Otherwise, an `A2B_SPIINT.BADCMD` interrupt occurs. The `A2B_SPIINT.SPIDTERR` bit is set when an extended bulk transaction fails to complete.

Programming Sequence for Extended Bulk SPI Transactions

Use the following sequence to program an extended bulk SPI transaction:

1. Configure the tunnels. See SPI Data Tunnels.
2. To initiate the data transfer, send the command bytes followed by N ($\leq$ 256) bytes of data for extended bulk SPI transaction from the host.
3. Start a standard bulk transaction to end the extended bulk transaction.
4. If the command byte is not valid, the `A2B_SPIINT.BADCMD` bit is set and the transaction is ignored.

Data Tunnel Restrictions

In SPI over distance communication, the A^2B transceiver connected to the SPI host is the tunnel owner. The A^2B transceiver connected to an SPI slave device is the tunnel responder.

Multiple tunnel configurations can exist in the system. For example, the main node and sub node 0 can have one tunnel transaction, while sub node 1 and sub node 2 can have another tunnel transaction. Tunnels can be dynamically added, removed, or changed in size. Its preferable to configure more data tunnels (2-12) during initialization (while programming the remote SPI peripheral) and few or no tunnels during audio configuration.

The tunnel must be idle when:

- changing the tunnel configuration
- changing the tunnel owner
- changing the tunnel responder

A tunnel is idle when the `A2B_SPISTAT.SPIBUSY` and `A2B_SPISTAT.DTACTIVE` bits are not set on the current data tunnel owner transceiver.

NOTE: It is possible to have overlapping data tunnels as shown in the ***Overlapping Data Tunnels*** figure. However, the overlapping data tunnels must use different audio slots as data tunnel slots.

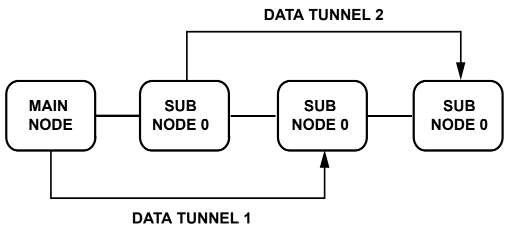


Figure 5-53: Overlapping Data Tunnels

Data Tunnel Configuration Examples

Tunnel configuration examples are included in the following sections. The definitions and restrictions apply to all tunnel configurations. For details regarding general slot management and terms, see Mapping Between TDM Channels and A^2B Slots . Each main and subordinate transceiver in a full A^2B system must be properly configured for the desired slot management scheme and format for both upstream and downstream traffic on the bus between any two transceivers.

Main Node Tunnel Configuration

The *Main Node Tunnel Configuration* figure shows examples of valid data tunnel slot configuration on a main node with the data tunnel slots before, within, and after audio data slots, as defined by the `A2B_DNSLOTS` and `A2B_UPSLOTS` registers.

Restrictions include:

- Setting the data tunnel first node bit (`A2B_DTCFG.DTFRST` = 1)
- Clearing the data tunnel last node bit (`A2B_DTCFG.DTLAST`= 0)
- Properly configuring the data tunnel slots window within the downstream A^{2}B data slots (`A2B_DTDNOFFS` ≤ `A2B_DNSLOTS`)
- Properly configuring the data tunnel slots window within the upstream A^{2}B data slots (`A2B_DTUPOFFS` ≤ `A2B_UPSLOTS`)

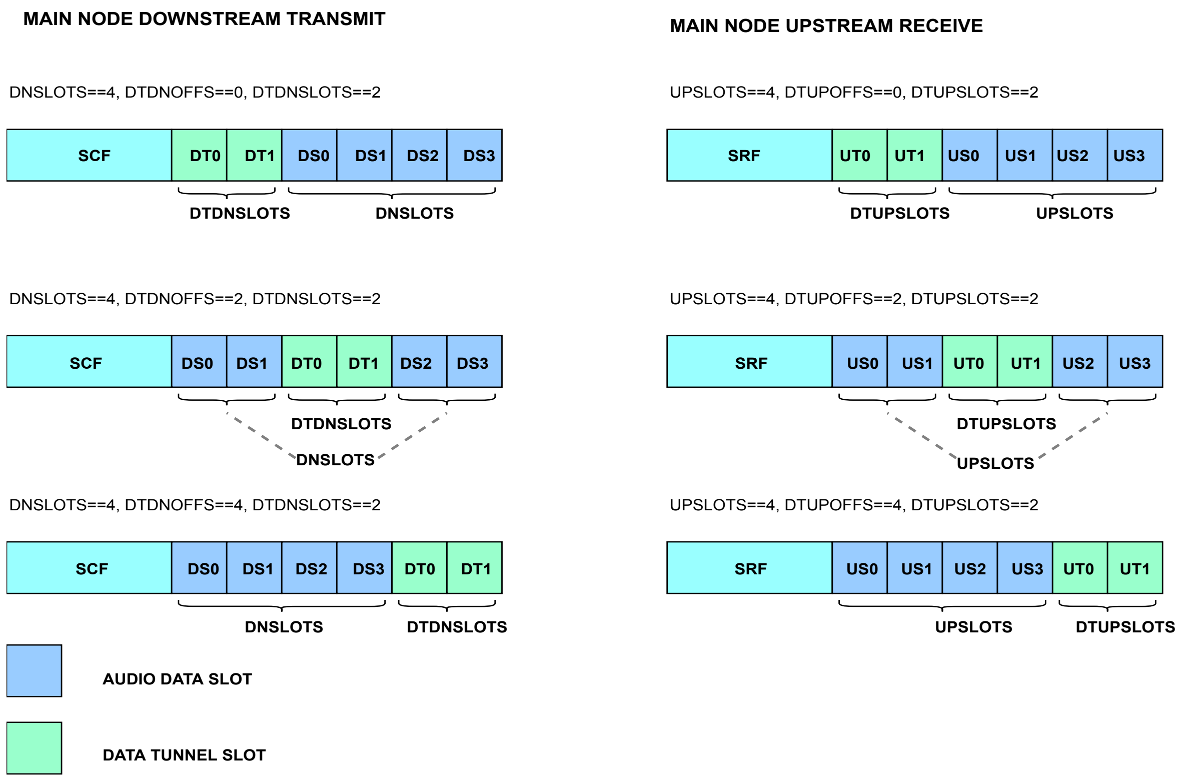

MAIN NODE DOWNSTREAM TRANSMIT
MAIN NODE UPSTREAM RECEIVE
DNSLOTS==4, DTDNOFFS==0, DTDNSLOTS==2
SCF
DT0
DT1
DS0
DS1
DS2
DS3
DTDNSLOTS
DNSLOTS
UPSLOTS==4, DTUPOFFS==0, DTUPSLOTS==2
SRF
UT0
UT1
US0
US1
US2
US3
DTUPSLOTS
UPSLOTS
DNSLOTS==4, DTDNOFFS==2, DTDNSLOTS==2
UPSLOTS==4, DTUPOFFS==2, DTUPSLOTS==2
DNSLOTS==4, DTDNOFFS==4, DTDNSLOTS==2
UPSLOTS==4, DTUPOFFS==4, DTUPSLOTS==2
AUDIO DATA SLOT
DATA TUNNEL SLOT

Figure 5-54: Main Node Tunnel Configuration

Last Subordinate Node Data Tunnel Configuration

The *Last Subordinate Node Tunnel Configuration* figure shows examples of valid data tunnel slot configurations on a last subordinate node with the data tunnel slots before, within, and after audio data slots, as defined by the `A2B_DNSLOTS` and `A2B_UPSLOTS` registers.

Restrictions include:

- Clearing the data tunnel first node bit (A2B_DTCFG.DTFRST = 0)
- Setting the data tunnel last node bit (A2B_DTCFG.DTLAST= 1)
- Disabling downstream data slot masks (A2B_LDNSLOTS.DNMASKEN = 0)
- Properly configuring the data tunnel slots window within the downstream A^{2}B data slots (A2B_DTDNOFFS ≤ A2B_LDNSLOTS)
- Properly configuring the data tunnel slots window within the upstream A^{2}B data slots (A2B_DTUPOFFS ≤ A2B_LUPSLOTS)

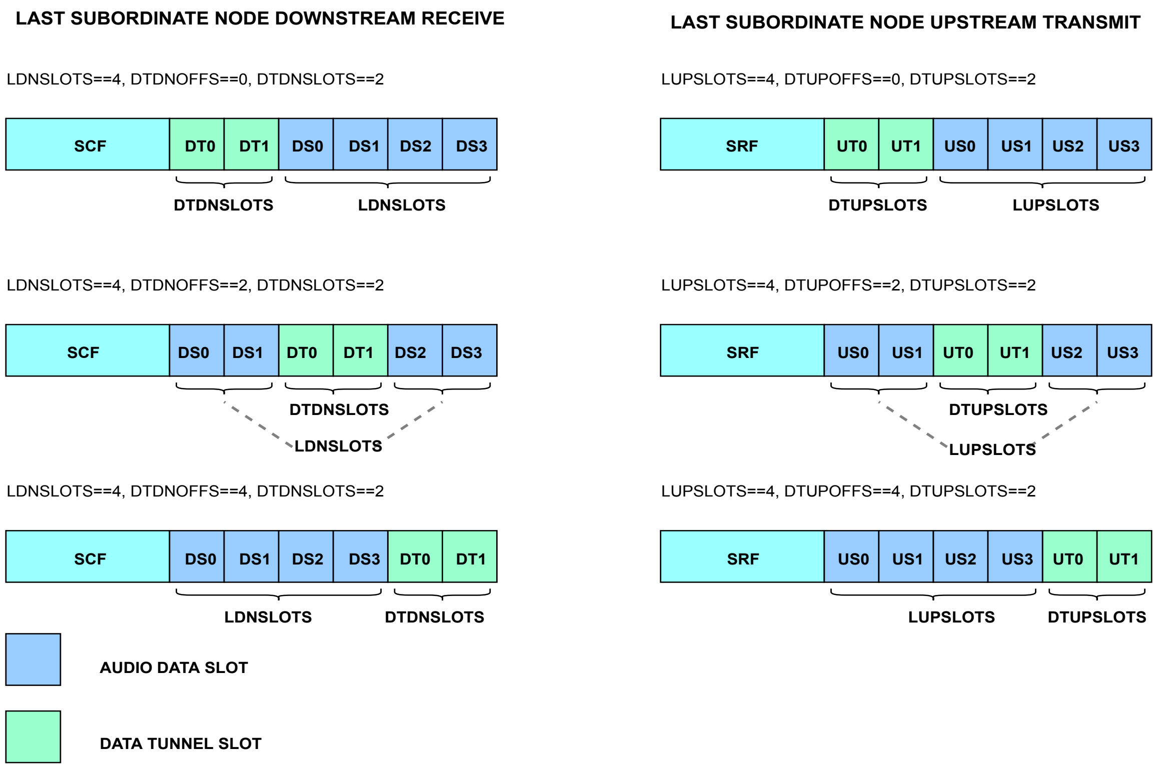

LAST SUBORDINATE NODE DOWNSTREAM RECEIVE
LAST SUBORDINATE NODE UPSTREAM TRANSMIT
LDNSLOTS==4, DTDNOFFS==0, DTDNSLOTS==2
SCF DT0 DT1 DS0 DS1 DS2 DS3
DTDNSLOTS
LDNSLOTS
LUPSLOTS==4, DTUPOFFS==0, DTUPSLOTS==2
SRF UT0 UT1 US0 US1 US2 US3
DTUPSLOTS
LUPSLOTS
LDNSLOTS==4, DTDNOFFS==2, DTDNSLOTS==2
SCF DS0 DS1 DT0 DT1 DS2 DS3
DTDNSLOTS
LDNSLOTS
LUPSLOTS==4, DTUPOFFS==2, DTUPSLOTS==2
SRF US0 US1 UT0 UT1 US2 US3
DTUPSLOTS
LUPSLOTS
LDNSLOTS==4, DTDNOFFS==4, DTDNSLOTS==2
SCF DS0 DS1 DS2 DS3 DT0 DT1
LDNSLOTS
DTDNSLOTS
LUPSLOTS==4, DTUPOFFS==4, DTUPSLOTS==2
SRF US0 US1 US2 US3 UT0 UT1
LUPSLOTS
DTUPSLOTS
AUDIO DATA SLOT
DATA TUNNEL SLOT

Figure 5-55: Last Subordinate Node Data Tunnel Configuration

Non-Last Subordinate Node - First Node of Tunnel Configuration

The ***Non-Last Subordinate Node - First Node Tunnel Configuration*** shows examples of valid data tunnel slot configurations for all subordinate nodes between the main node and the last subordinate node. The node is the first (most upstream) node of a data tunnel.

Restrictions include:

- Setting the data tunnel first node bit (A2B_DTCFG.DTFRST = 1)
- Clearing the data tunnel last node bit (A2B_DTCFG.DTLAST= 0)

- Enabling downstream data slot masks (`A2B_LDNSLOTS.DNMASKEN` = 1)
- Properly configuring the data tunnel slots window within the downstream A^{2}B data slots (`A2B_DTDNOFFS` ≥ `A2B_DNSLOTS` AND `A2B_DTDNOFFS` ≤ (`A2B_DNSLOTS` +`A2B_LDNSLOTS`))
- Properly configuring the data tunnel slots window within the upstream A^{2}B data slots (`A2B_DTUPOFFS` ≥ `A2B_UPSLOTS` AND `A2B_DTUPOFFS` ≤ MAX (`A2B_UPSLOTS`, upmaskrx)).

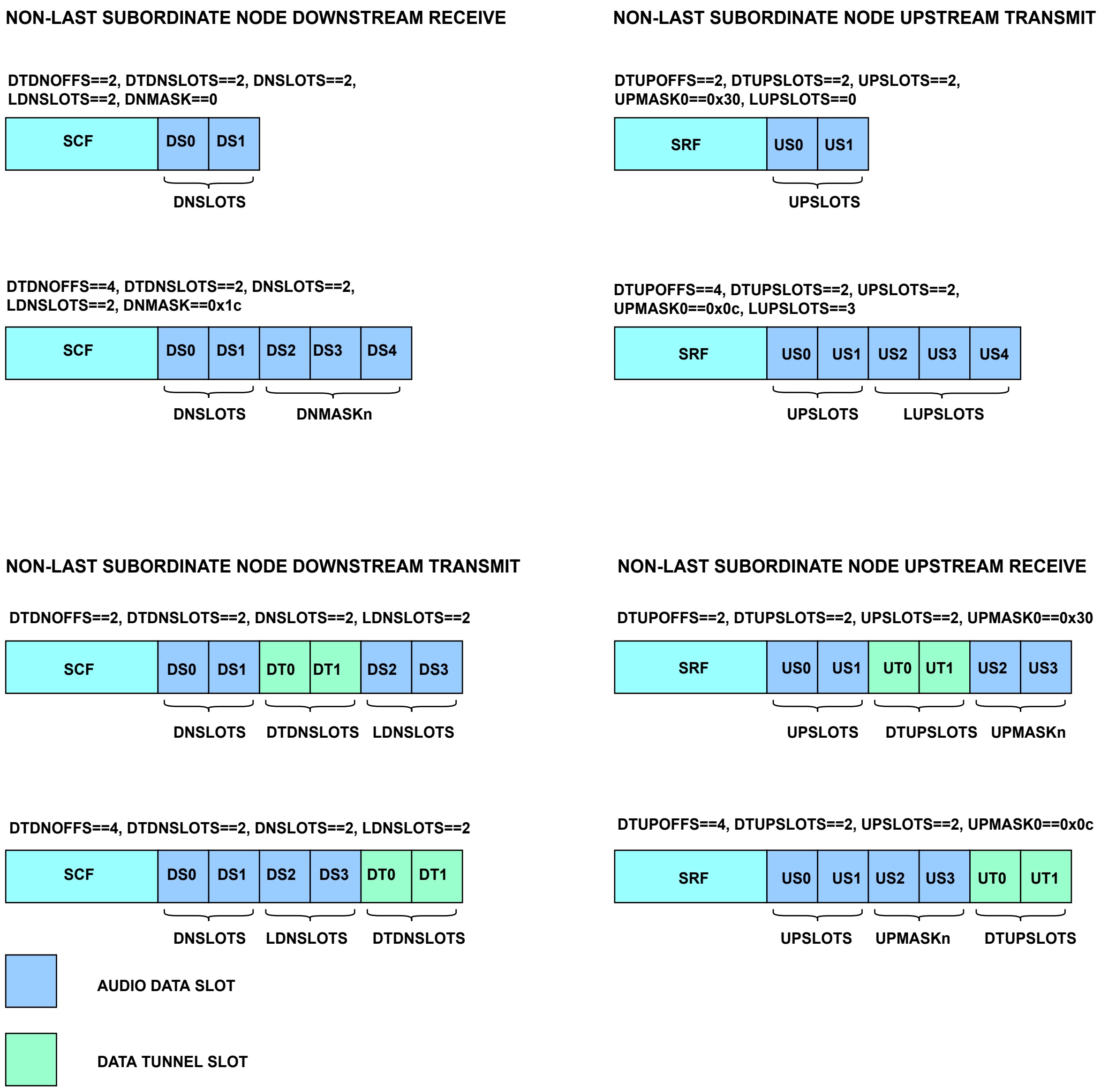

NON-LAST SUBORDINATE NODE DOWNSTREAM RECEIVE
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==2, LDNSLOTS==2, DNMASK==0
SCF
DS0
DS1
DNSLOTS
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==2, LDNSLOTS==2, DNMASK==0x1c
SCF
DS0
DS1
DS2
DS3
DS4
DNSLOTS
DNMASKn
NON-LAST SUBORDINATE NODE UPSTREAM TRANSMIT
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==2, UPMASK0==0x30, LUPSLOTS==0
SRF
US0
US1
UPSLOTS
DTUPOFFS==4, DTUPSLOTS==2, UPSLOTS==2, UPMASK0==0x0c, LUPSLOTS==3
SRF
US0
US1
US2
US3
US4
UPSLOTS
LUPSLOTS
NON-LAST SUBORDINATE NODE DOWNSTREAM TRANSMIT
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==2, LDNSLOTS==2
SCF
DS0
DS1
DT0
DT1
DS2
DS3
DNSLOTS
DTDNSLOTS
LDNSLOTS
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==2, LDNSLOTS==2
SCF
DS0
DS1
DS2
DS3
DT0
DT1
DNSLOTS
LDNSLOTS
DTDNSLOTS
NON-LAST SUBORDINATE NODE UPSTREAM RECEIVE
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==2, UPMASK0==0x30
SRF
US0
US1
UT0
UT1
US2
US3
UPSLOTS
DTUPSLOTS
UPMASKn
DTUPOFFS==4, DTUPSLOTS==2, UPSLOTS==2, UPMASK0==0x0c
SRF
US0
US1
US2
US3
UT0
UT1
UPSLOTS
UPMASKn
DTUPSLOTS
AUDIO DATA SLOT
DATA TUNNEL SLOT

Figure 5-56: Non-Last Subordinate Node - First Node Tunnel Configuration

Non-Last Subordinate Node - Middle Node of Tunnel Configuration

The ***Non-Last Subordinate Node - Middle Node Tunnel Configuration*** shows the data tunnel configuration of a middle node in a tunnel. This node is not the first (most upstream) or last (most downstream) node of the data tunnel.

Restrictions include:

- Clearing the data tunnel first node bit (`A2B_DTCFG.DTFRST` = 0)
- Clearing the data tunnel last node bit (`A2B_DTCFG.DTLAST`= 0)
- Properly configuring the data tunnel window within the downstream A^{2}B data slots (`A2B_DTDNOFFS` ≤ `A2B_DNSLOTS`)
- Properly configuring the data tunnel window within the upstream A^{2}B data slots (`A2B_DTUPOFFS` ≤ `A2B_UPSLOTS`)

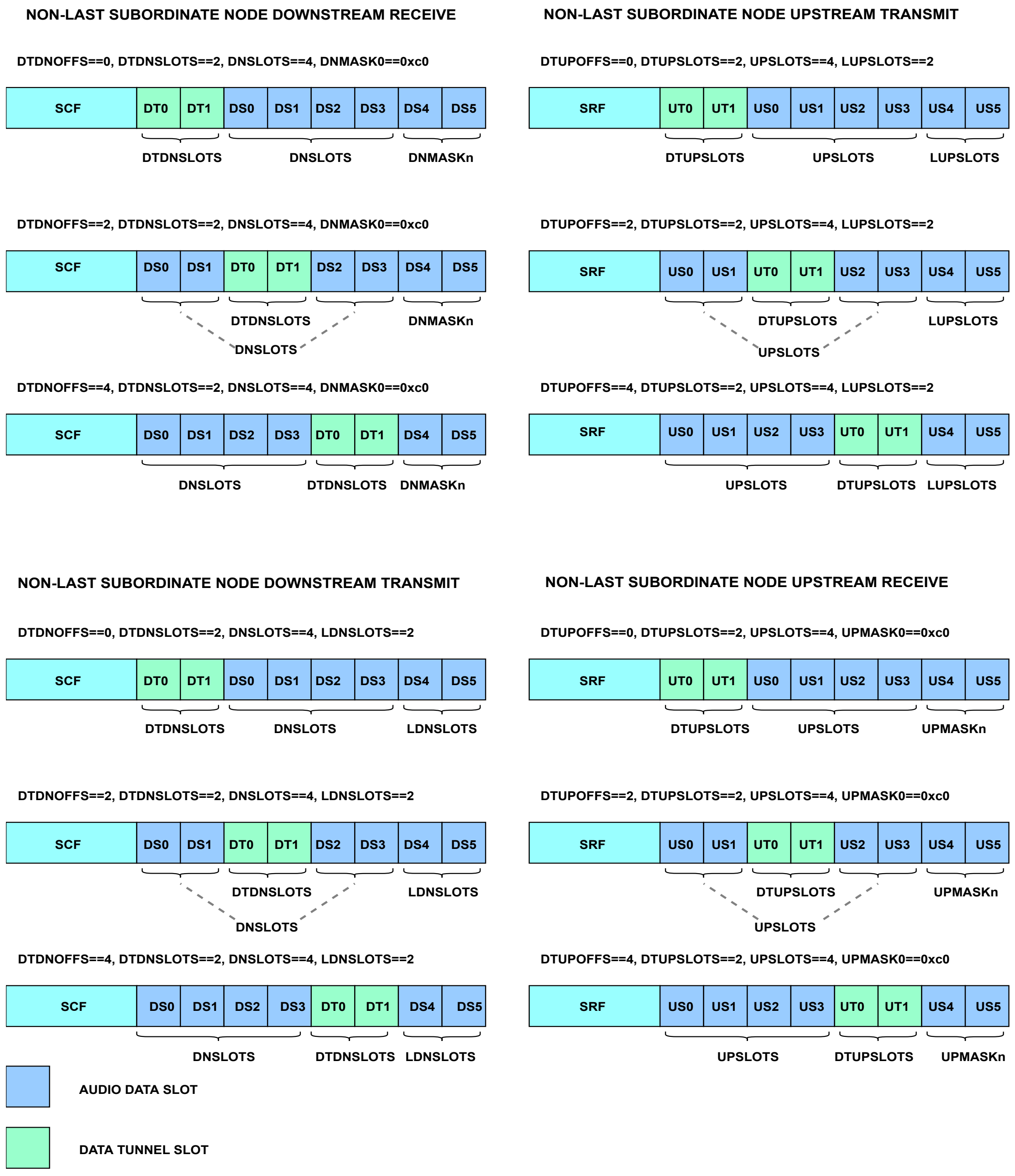

NON-LAST SUBORDINATE NODE DOWNSTREAM RECEIVE
DTDNOFFS==0, DTDNSLOTS==2, DNSLOTS==4, DNMASK0==0xc0
SCF DT0 DT1 DS0 DS1 DS2 DS3 DS4 DS5
DTDNSLOTS DNSLOTS DNMASKn
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==4, DNMASK0==0xc0
SCF DS0 DS1 DT0 DT1 DS2 DS3 DS4 DS5
DTDNSLOTS DNMASKn DNSLOTS
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==4, DNMASK0==0xc0
SCF DS0 DS1 DS2 DS3 DT0 DT1 DS4 DS5
DNSLOTS DTDNSLOTS DNMASKn
NON-LAST SUBORDINATE NODE UPSTREAM TRANSMIT
DTUPOFFS==0, DTUPSLOTS==2, UPSLOTS==4, LUPSLOTS==2
SRF UT0 UT1 US0 US1 US2 US3 US4 US5
DTUPSLOTS UPSLOTS LUPSLOTS
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==4, LUPSLOTS==2
SRF US0 US1 UT0 UT1 US2 US3 US4 US5
DTUPSLOTS LUPSLOTS UPSLOTS
DTUPOFFS==4, DTUPSLOTS==2, UPSLOTS==4, LUPSLOTS==2
SRF US0 US1 US2 US3 UT0 UT1 US4 US5
UPSLOTS DTUPSLOTS LUPSLOTS
NON-LAST SUBORDINATE NODE DOWNSTREAM TRANSMIT
DTDNOFFS==0, DTDNSLOTS==2, DNSLOTS==4, LDNSLOTS==2
SCF DT0 DT1 DS0 DS1 DS2 DS3 DS4 DS5
DTDNSLOTS DNSLOTS LDNSLOTS
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==4, LDNSLOTS==2
SCF DS0 DS1 DT0 DT1 DS2 DS3 DS4 DS5
DTDNSLOTS LDNSLOTS DNSLOTS
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==4, LDNSLOTS==2
SCF DS0 DS1 DS2 DS3 DT0 DT1 DS4 DS5
DNSLOTS DTDNSLOTS LDNSLOTS
NON-LAST SUBORDINATE NODE UPSTREAM RECEIVE
DTUPOFFS==0, DTUPSLOTS==2, UPSLOTS==4, UPMASK0==0xc0
SRF UT0 UT1 US0 US1 US2 US3 US4 US5
DTUPSLOTS UPSLOTS UPMASKn
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==4, UPMASK0==0xc0
SRF US0 US1 UT0 UT1 US2 US3 US4 US5
DTUPSLOTS UPMASKn UPSLOTS
DTUPOFFS==4, DTUPSLOTS==2, UPSLOTS==4, UPMASK0==0xc0
SRF US0 US1 US2 US3 UT0 UT1 US4 US5
UPSLOTS DTUPSLOTS UPMASKn
AUDIO DATA SLOT
DATA TUNNEL SLOT

Figure 5-57: Non-Last Subordinate Node - Middle Node of Tunnel Configuration

Non-Last Subordinate Node - Last Node of Tunnel Configuration

The *Non-Last Subordinate Node - Middle Node Tunnel Configuration* shows the data tunnel configuration of the non-last subordinate node that is the last (most downstream) node of a data tunnel. It shows examples of valid data tunnel slots configuration on a non-last subordinate node with the data tunnel slots before, within, and after audio data slots, as defined by the A2B_DNSLOTS and A2B_UPSLOTS registers.

Restrictions include:

- Clearing the data tunnel first node bit (`A2B_DTCFG.DTFRST` = 0)
- Setting the data tunnel last node bit (`A2B_DTCFG.DTLAST`= 1)
- Enabling downstream data slot masks (`A2B_LDNSLOTS.DNMASKEN` = 1)
- Properly configuring the data tunnel slots window within the downstream A^{2}B data slots (`A2B_DTDNOFFS` ≥ `A2B_DNSLOTS` AND `A2B_DTDNOFFS` ≤ MAX (`A2B_DNSLOTS`, dnmaskrx)).
- Properly configuring the data tunnel slots window within the upstream A^{2}B data slots (`A2B_DTUPOFFS` ≥ `A2B_UPSLOTS` AND `A2B_DTUPOFFS` ≤ (`A2B_UPSLOTS`+`A2B_LUPSLOTS`))

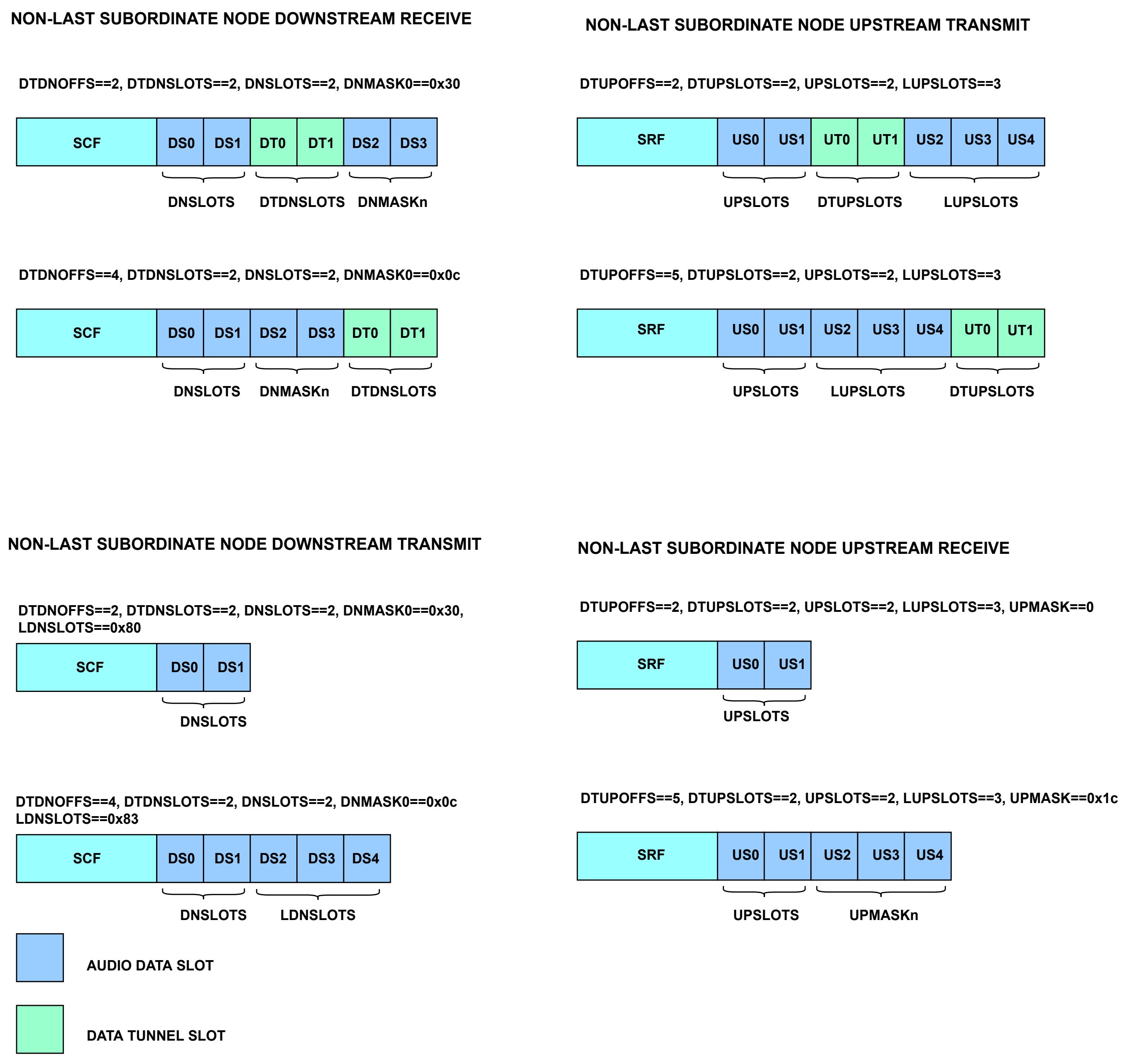

NON-LAST SUBORDINATE NODE DOWNSTREAM RECEIVE
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==2, DNMASK0==0x30
SCF DS0 DS1 DT0 DT1 DS2 DS3
DNSLOTS DTDNSLOTS DNMASKn
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==2, DNMASK0==0x0c
SCF DS0 DS1 DS2 DS3 DT0 DT1
DNSLOTS DNMASKn DTDNSLOTS
NON-LAST SUBORDINATE NODE UPSTREAM TRANSMIT
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==2, LUPSLOTS==3
SRF US0 US1 UT0 UT1 US2 US3 US4
UPSLOTS DTUPSLOTS LUPSLOTS
DTUPOFFS==5, DTUPSLOTS==2, UPSLOTS==2, LUPSLOTS==3
SRF US0 US1 US2 US3 US4 UT0 UT1
UPSLOTS LUPSLOTS DTUPSLOTS
NON-LAST SUBORDINATE NODE DOWNSTREAM TRANSMIT
DTDNOFFS==2, DTDNSLOTS==2, DNSLOTS==2, DNMASK0==0x30, LDNSLOTS==0x80
SCF DS0 DS1
DNSLOTS
DTDNOFFS==4, DTDNSLOTS==2, DNSLOTS==2, DNMASK0==0x0c LDNSLOTS==0x83
SCF DS0 DS1 DS2 DS3 DS4
DNSLOTS LDNSLOTS
NON-LAST SUBORDINATE NODE UPSTREAM RECEIVE
DTUPOFFS==2, DTUPSLOTS==2, UPSLOTS==2, LUPSLOTS==3, UPMASK==0
SRF US0 US1
UPSLOTS
DTUPOFFS==5, DTUPSLOTS==2, UPSLOTS==2, LUPSLOTS==3, UPMASK==0x1c
SRF US0 US1 US2 US3 US4
UPSLOTS UPMASKn
AUDIO DATA SLOT
DATA TUNNEL SLOT

Figure 5-58: Non-Last Subordinate Node - Last Node of Tunnel Configuration

Loosening Data Tunnel Restrictions

Typically, all the data tunnel restrictions should be followed to set offset values (see Data Tunnel Restrictions). However, if a system cannot strictly follow the offset restrictions, there are two cases where restrictions can be safely loosened at the cost of bandwidth.

In the *Data Tunnel Restriction Workaround* figure, the first node in the tunnel is the main node and the last node is A^2B sub node 1.

The data tunnel up offset restrictions are as follows:

1. Main node: A2B_DTUPOFFS ≤ A2B_UPSLOTS (A2B_DTUPOFFS.DTUPOFFS ≤ 4)
2. Sub node 1 (non-last sub node acting as last node of tunnel): A2B_DTUPOFFS ≤ A2B_UPSLOTS (A2B_DTUPOFFS.DTUPOFFS ≤ 4)
3. Sub node 1 (non-last sub node acting as last node of tunnel): A2B_DTUPOFFS ≤ (A2B_UPSLOTS + A2B_LUPSLOTS) (A2B_DTUPOFFS.DTUPOFFS ≤ 4 + 0)

   Given these three conditions, A2B_DTUPOFFS = 4 will satisfy all restrictions and is a valid data tunnel up offset.

The data tunnel down offset restrictions are follows:

1. Main node: A2B_DTDNOFFS ≤ A2B_DTSLOTS.DTDNSLOTS(A2B_DTDNOFFS ≥ 2)
2. Sub node 1 (non-last sub node acting as last node of tunnel): A2B_DTDNOFFS ≥ A2B_DNSLOTS (A2B_DTDNOFFS.DTDNOFFS ≥ 4)
3. Sub node 1 (non-last sub node acting as last node of tunnel): A2B_DTDNOFFS ≤ MAX(A2B_DNSLOTS, dnmaskrx) (A2B_DTDNOFFS.DTDNOFFS ≤ MAX (4,0))

   There is no data tunnel down offset that can satisfy all of these three conditions. For example, a number cannot be both less than 2 and greater than 4. If a system cannot follow these restrictions, then modify the offset value of sub node 1 (non-last sub node acting as last node of tunnel) to A2B_DTDNOFFS ≤ A2B_DNSLOTS. Ensure this value is the same as the main node (first node of tunnel). When using this workaround, pass A2B_DTSLOTS.DTDNSLOTS + A2B_DNSLOTS (for example, A2B_DNSLOTS = A2B_DTSLOTS.DTDNSLOTS + A2B_DNSLOTS) downstream from this node.

   Similarly, when sub node 1 (non-last sub node acting as first node of tunnel) A2B_DTUPOFFS may not be set by following all the restrictions. If a system cannot follow these restrictions, then modify the offset value of sub node 1 (non-last sub node acting as first node of tunnel) configuration to A2B_DTUPOFFS ≤ A2B_UPSLOTS. Make sure this value is the same as the last node of tunnel configuration. When using this workaround, pass A2B_DTSLOTS.DTUPSLOTS+A2B_UPSLOTS (for example, A2B_UPSLOTS = A2B_DTSLOTS.DTUPSLOTS + A2B_UPSLOTS) upstream from this node.

   Using this workaround, tunnel slots are not removed on the bus. They are passed upstream/downstream limiting bandwidth.

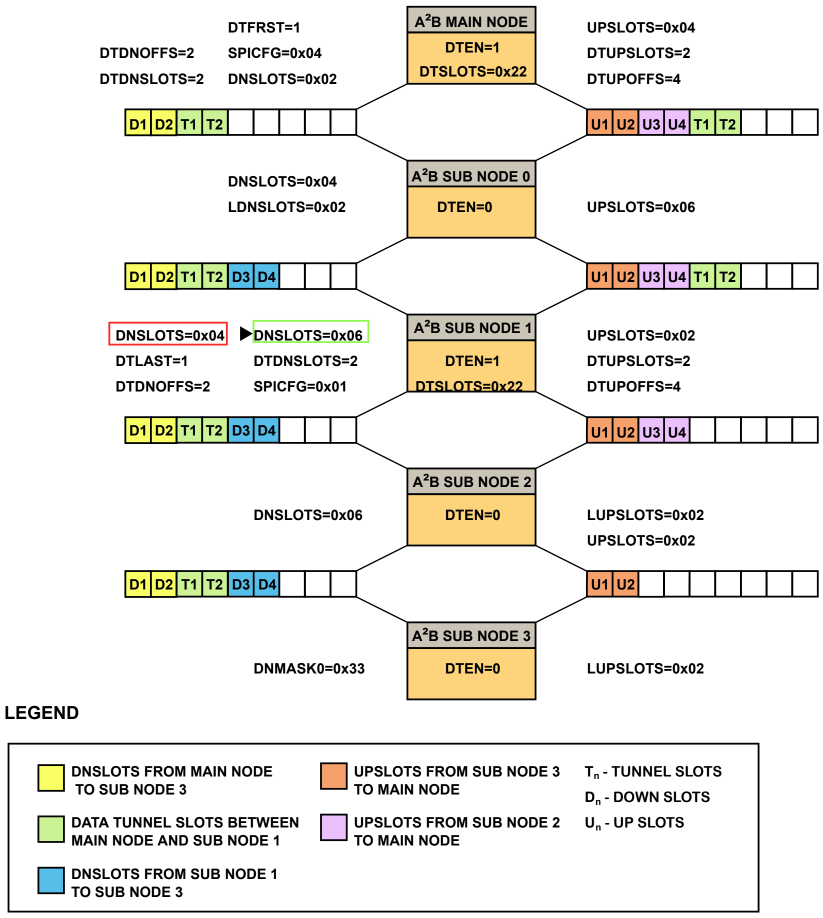


Figure 5-59: Workaround for Data Tunnel Restrictions

SPI Abort

The SPI host terminates the ongoing SPI transaction and flushes all the data in FIFO. It can be initiated when `A2B_SPISTAT.SPIBUSY=1`. If any other transaction is initiated when `A2B_SPISTAT.SPIBUSY=1`, an interrupt is generated and the `A2B_SPIINT.BADCMD` bit is set.

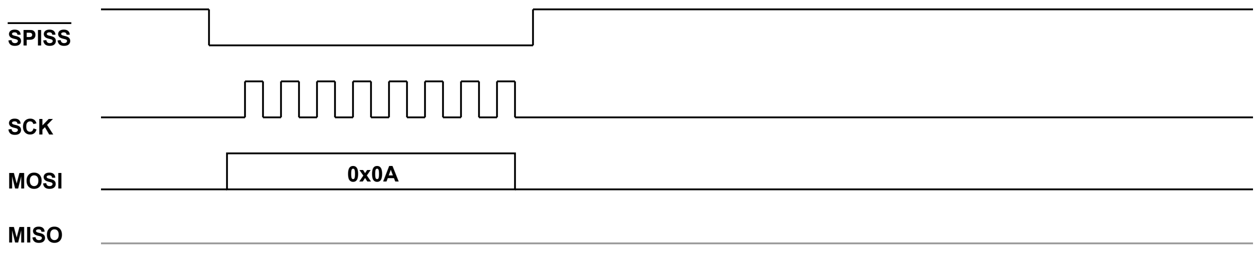


Figure 5-60: SPI Abort Transaction

NOTE: The SPI abort can be initiated by an SPI host connected to an A^{2}B transceiver acting as the SPI slave. The A^{2}B transceiver may or may not be tunnel owner.

SPI Interrupts and Errors

The ***SPI Interrupts*** table describes the available interrupts, when they are issued, and the transactions that can cause the interrupt.

Table 5-21: SPI Interrupts

Interrupt	Description of Occurrence	Transactions
`A2B_SPIINT.BADCMD`	1. When starting a SPI transfer while `A2B_SPISTAT.SPIBUSY`= 1 results in a BADCMD interrupt. 2. If the first byte of a transaction is not a valid command (for example, 0x10 to 0xB9) 3. When $\overline{\text{SPISS}}$ is deasserted before sending a specified number of bytes (for example, sending 10 bytes of data but sent the write data length as FF) 4. When $\overline{\text{SPISS}}$ is not deasserted after sending the specified number of bytes (sending write data length as 10, but sent more than 10 bytes of data without deassertion of the slave select signal)	SPI Local Register Write SPI Local Register Read SPI Remote Register Read Request SPI Remote Register Write SPI BUS FIFO Read SPI Data Tunnel FIFO Read SPI Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write SPI to Remote I^{2}C Read SPI to Remote I^{2}C Write SPI Status Read SPI Abort
`A2B_SPIINT.FIFOUNF`	1. When accessing more bytes than the remote register read request command received. 2. When accessing more bytes than the SPI to I^{2}C read request command received 3. When accessing more bytes than the atomic read request command received 4. Initiating a bus FIFO read, or data FIFO read prior to the slave register read request, SPI to I^{2}C read request, atomic read request, and full duplex	SPI BUS FIFO Read SPI Data Tunnel FIFO Read
`A2B_SPIINT.FIFOOVF`	When any SPI command attempts to write more than 32 bytes to the bus FIFO or more than 256 bytes to the data FIFO.	No existing command sets this bit

Table 5-21: SPI Interrupts (Continued)

Interrupt	Description of Occurrence	Transactions
A2B_SPIINT.SPIDTERR	When any one of the following bits are set: • A2B_SPISTAT.DTINVALID • A2B_SPISTAT.DTBADPKT • A2B_SPISTAT.DTABORT	SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write
A2B_SPIINT.SPII2CERR	When the read/write fails due to bus conditions or I^2C peripheral issues. (If the A2B_CHIP register is not programmed with a valid I^2C peripheral address in the subordinate transceiver or if the transceiver node being accessed is dropped out of the bus).	SPI to Remote I^2C Read SPI Remote I^2C Write
A2B_SPIINT.SPIREGERR	When the remote register (an A^2B sub transceiver register) read/write operation fails to complete.	SPI Remote Register Read Request SPI Remote Register Write
A2B_SPIINT.SPIDONE	After the completion of each access (after A2B_SPISTAT.SPIBUSY is cleared (=0)).	SPI Local Register Write SPI Remote Register Read Request SPI Remote Register Write SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write SPI to Remote I^2C Read SPI Remote I^2C Write

Table 5-21: SPI Interrupts (Continued)

Interrupt	Description of Occurrence	Transactions
`A2B_SPISTAT.SPIBUSY`	While SPI transactions are in progress	SPI Local Register Write SPI Remote Register Read Request SPI Remote Register Write SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write SPI to Remote I^2C Read SPI Remote I^2C Write
`A2B_SPISTAT.DTACTIVE`	When the SPI port is actively using the data tunnels (for example, the transaction is on-going)	SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write
`A2B_SPISTAT.DTINVALID`	1. When the tunnels are improperly configured (for example, the `A2B_DTCFG.DTLAST` bit is not set in the last node of the tunnel) 2. When there is an illegal/inconsistent number of tunnel slots (audio slot sizes of 8, 12, 28 and less than two tunnel slots) 3. Tunnel owner and/or responder not configured	SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write
`A2B_SPISTAT.DTBADPKT`	1. When the data is corrupted and the CRC check fails 2. If a node in between the responder and owner replaces the SPI data with audio or does not pass the data downstream/ upstream 3. If the `A2B_DTCFG.DTLAST`bit is not set in the non-last sub node acting as last tunnel node	SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write

Table 5-21: SPI Interrupts (Continued)

Interrupt	Description of Occurrence	Transactions
A2B_SPISTAT.DTABORT	1. When a corrupted packet is received 2. When there is a missing response packet (for example, the tunnel responder does not send the response to the current data tunnel transaction) 3. When the middle nodes in between the tunnel owner and responder does not pass the tunnel slots upstream or downstream 4. When the SPI host initiates an extended FD/bulk or other legacy data tunnel transaction to the tunnel responder or to some other node while an extended bulk/FD is open on the tunnel responder	SPI Data Tunnel Atomic Write SPI Atomic Read Request Full Duplex Bulk SPI Write Extended Full Duplex Extended Bulk Write

SPI Data Tunnel Error Handling

Because multiple bits in the A2B_SPISTAT register contribute to the SPIDTERR interrupt, these bits are not automatically cleared when SPIDTERR is cleared. The recommended interrupt handling flow for SPIDTERR is:

- Read the A2B_INTSRC and A2B_INTTYPE registers (or the A2B_LINTTYPE register for a sub node tunnel owner) and see type 0x43.
- Read the A2B_SPISTAT register from the appropriate node to see which bit is set (A2B_SPISTAT.DTINVALID /A2B_SPISTAT.DTABORT / A2B_SPISTAT.DTBADPKT)
- Write to the A2B_SPISTAT register to clear these bits. A new SPIDTERR interrupt is not generated until these bits are cleared in the A2B_SPISTAT register.

The tunnel owner aborts the transaction when there is a corrupted or missed response from a tunnel responder. The tunnel responder aborts the transaction if there is corrupted data from the tunnel owner. This information is updated in A2B_SPISTAT and A2B_SPIINT registers of the tunnel owner. When the transaction is aborted, the tunnel owner (main node/sub node) is not responsible for retrying the transaction.

Once the transaction is aborted, the A2B_SPISTAT register and A2B_SPISTAT.DTINVALID / A2B_SPISTAT.DTABORT / A2B_SPISTAT.DTBADPKT bits of the tunnel owner are updated. The host processor connected to tunnel owner (main node/sub node) is responsible for reading these registers and retrying the transaction.

For atomic transactions, since the tunnel responder waits until it gets all bytes of data, all successfully received data are discarded without the remote SPI peripheral seeing the transaction errors. Therefore, the host must restart the transaction.

For full duplex transactions, if the `A2B_SPICFG.ENFDCS` bit is not set, no transaction occurs. If set, an unspecified portion of the remote transaction may occur.

For bulk transactions, an unspecified portion of remote transaction may occur when there is a data tunnel error. The number of bytes of data transferred to remote the SPI peripheral connected to the tunnel responder may be unknown (unless the remote SPI peripheral gives this information to the host). Therefore, it is recommended to start the transaction again.

RTM Use Case

The *RTM Sample Demo Configuration* figure shows the configuration of a remote tuner module (RTM). The host processor uses SPI over distance to program the remote peripheral (RTM connected to sub node 1) to receive the audio information from the RTM using the I^2S audio slots. The configuration uses two downstream data tunnel slots and two upstream data tunnel slots. The number of these slots can be increased based on the requirement. See the ***RTM Requirements*** table. The A^2B main node acts as the first node of the tunnel; it is the tunnel owner. The A^2B sub node 1 is the last node of tunnel; it is the tunnel responder. The data tunnels in sub node 0 are disabled. Subordinate node 0 treats the tunnel data as audio information and passes the data downstream/upstream. Streams used in the sample demo are described in the ***Demo Streams*** table.

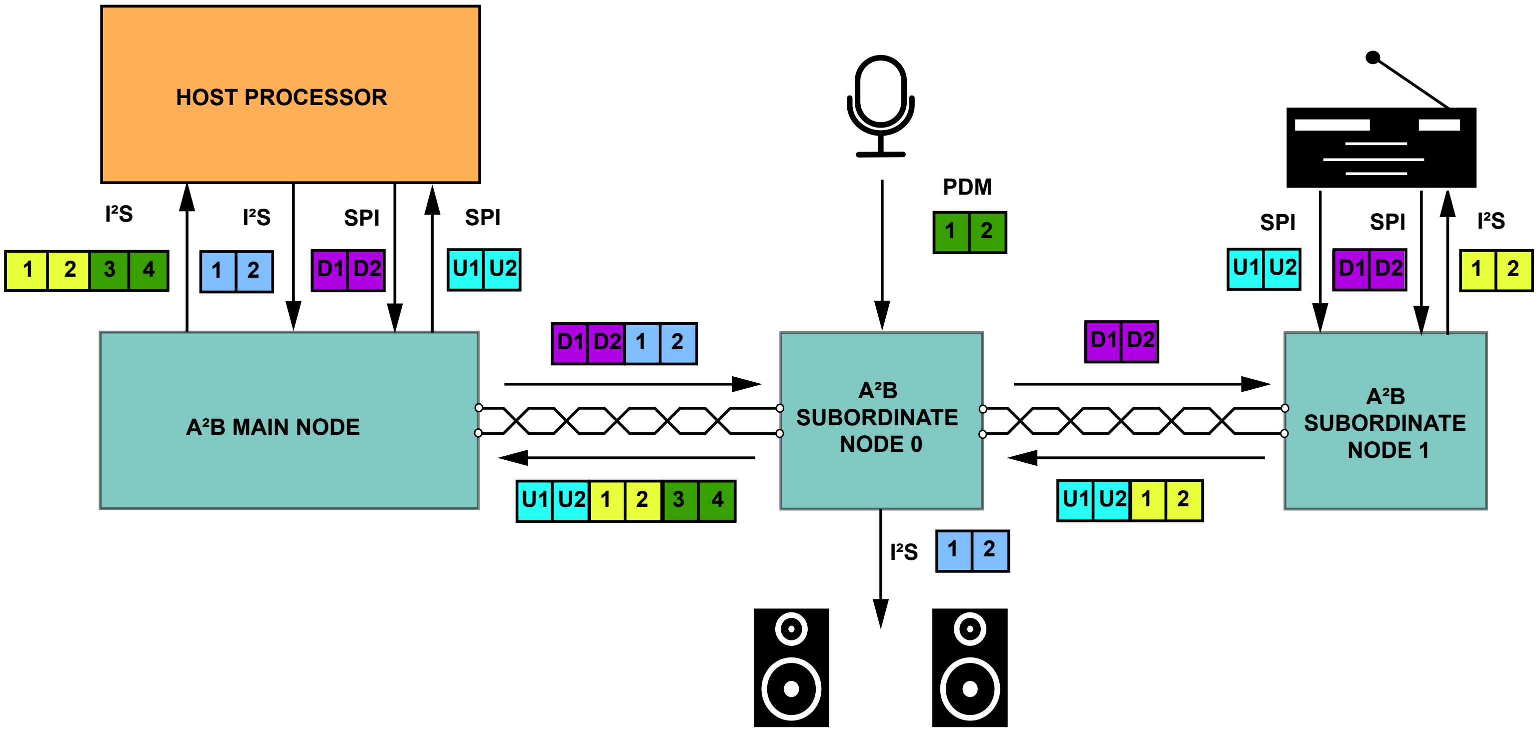


Figure 5-61: RTM Sample Demo Configuration

Table 5-22: RTM Requirements

	A^2B Main Node Transceiver	A^2B Subordinate Node 1 Transceiver
Functionality	**Tunnel owner**	**Tunnel responder**
Data tunnel downslots	2	2
Data tunnel upslots	2	2
SPI clock	-	9.8304 MHz
Subordinate Node select	ADR1	ADR1
SPI CPOL and SPI CPHA	Based on host	Based on RTM

Table 5-23: Demo Streams

Streams	From	To	Number of Slots
Data tunnel downslots	Main Node	Subordinate Node 1	2
Data tunnel upslots	Subordinate Node 1	Main Node	2
I^2S RTM data	Subordinate Node 1	Main Node	2
Mic data	Subordinate Node 0	Main Node	2
Speaker data	Main Node	Subordinate Node 0	2

Using the following procedure to program the RTM using SPI over distance:

```
 A2B_SPI_Over_Distance ()
{
 Discover_A2B ()
 Init_Tunnels_AudioSlots ()
Bulk_SPI ()//If the payload size is more than 256 bytes, reinitiate based
   on your payload size
Check_Status ()
 }
Discover_A2B ()
 {
  //Follow Simple discovery flow explained in Chapter7
 }
Init_Tunnels_AudioSlots ()
  {
```

The *RTM Use Case* figure shows registers used to configure the slots for RTM.

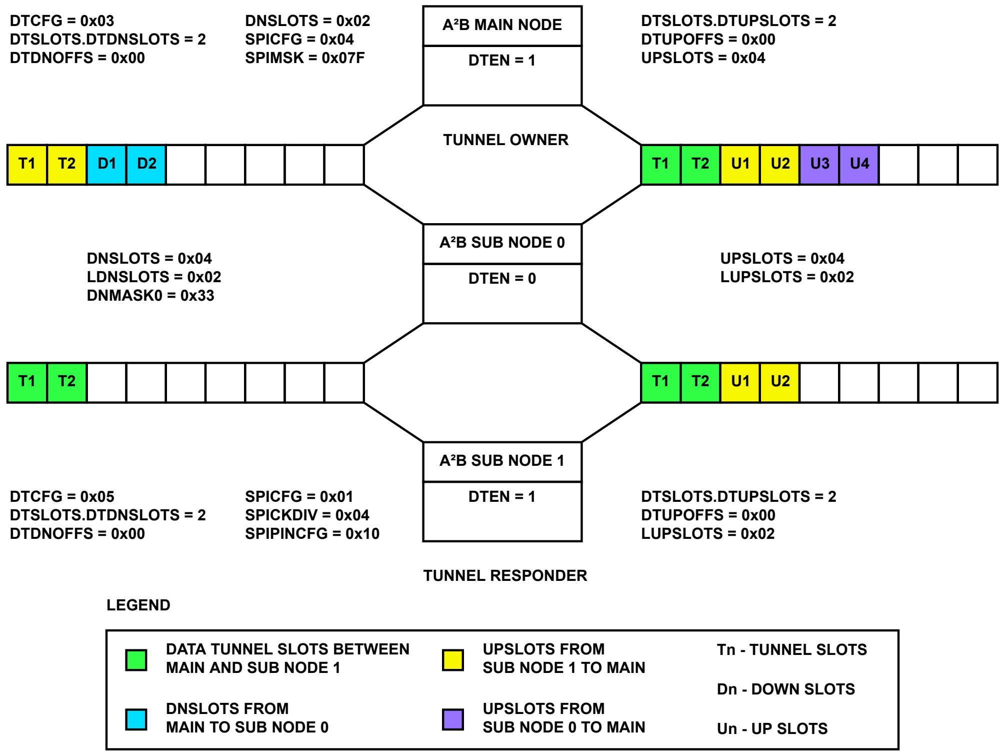


Figure 5-62: RTM Use Case

Program the registers shown in the *RTM Use Case* figure using SPI/I^{2}C. While programming through the SPI, use the local register access and remote register access.

Since sub node 0 does not have tunnels enabled, it treats the tunnel slots as indicated by the `A2B_DNSLOTS` or `A2B_UPSLOTS` register. When programming a node transceiver without the tunnels enabled, program the `A2B_DNSLOTS`register with slot information (Downslots + Data Tunnel Downslots) from the previous node transceiver. Similarly, program the `A2B_UPSLOTS`. register with slot information (Upslots + Data Tunnel Upslots) from the previous node transceiver.

```
}
      Bulk_SPI ()
      {
      SPI_Write (0x06,0x01, LEN-1, DATA0, DATA1..............., DATALEN-1) // Bulk SPI
Write (0X06 is the Bulk SPI Command,0x01 is NODES/SS byte and LEN is the length
is the write data length)
      //SPI_Write: Drive data on MOSI after asserting Subordinate Node Select
      }
      Check_Status ()
      {
      //Read Status of SPI using SPI_Status Read Transaction or through the
normal register read
      //SPIBUSY should be cleared before starting a new transaction
      //Read SPIINT to check for interrupts and clear by writing one to the bit
if there is any
```

```
        }
```

Pulse Width Modulation

LEDs can be dimmed in two ways: analog dimming or pulse-width modulation dimming. Most LED implementations use Pulse Width Modulation (PWM) for controlling brightness. Analog dimming changes LED light output by adjusting the DC current in the string, while PWM dimming achieves the same effect by varying the duty cycle of a constant current in the string to effectively change the average current in the string (more high time= brighter, less high time= dimmer).

Analog dimming is inappropriate for many applications because it loses dimming accuracy and it skews the color of the LEDs. In contrast, PWM dimming can produce higher dimming ratios without any significant loss of accuracy, and no change in LED color.

The PWM module is a programmable waveform generator. PWM signals from the transceiver can be connected to some external LED drivers to enable digital PWM dimming. The PWM generation unit features:

- Up to three PWM output channels
- One output enable (OE) main dimmer
- Two frequency generators
- 11 programmable frequencies
- LED blinking
- Frequency hopping

Pin Assignment

The PWM outputs are shared with the SPI pins. SPI functionality must be disabled when any PWM channels are enabled.

The ***PWM Pin Assignment*** table shows how the PWM channels map to the external pins of the transceiver.

Table 5-24: PWM Pin Assignment

PWM Function	**Transceiver Pin**
PWM Output Enable	ADR1
PWM Channel 1	MISO
PWM Channel 2	MOSI
PWM Channel 3	SCK

Frequency Selection

Two PWM frequency generators are available. PWM channels 1-3 run from one frequency generator. The OE channel runs from the other. The frequency generators can be independently programmed to one of the following frequencies: 192 kHz, 96 kHz, 48 kHz, 24 kHz, 12 kHz, 3 kHz, 1.5 kHz, 750 Hz, 375 Hz, 187.5 Hz. To set the frequency of these generators, program the `A2B_PWMFREQ` register. For LED dimming, the frequency of the OE channel must be less than that of the PWM channels. When programmed to a same frequency, the PWM and OE channels are generated so that each output rises on a different phase to limit instantaneous current draw. See the ***PWM Phases*** table.

Table 5-25: PWM Phases

PWM Function	Phase
PWM1	90
PWM2	180
PWM3	270
Output Enable (Dim)	0

Phase alignment between OE channels and PWM is not specified when they are operating at different frequencies.

Frequency Hopping

The PWM block can be configured to operate using a frequency hopping scheme. Two LFSR-based frequency hopping engines are available: one for PWM channels 1-3 and the other for the OE channel. The PWM frequency hopper randomly selects frequencies from 187.5 Hz to 3 kHz to spread the PWM emissions over a range of frequencies. In this mode, phase alignment is fixed to 90, 180, 270 of a 3 kHz period.

Led Blinking

The PWM supports independent blink rates of 0, 0.25, 0.75, and 1 second (blink period) for the PWM and OE channels. The blink duty cycle is fixed at 50%. If the blink rate is 0.5, the LED is ON for 0.5 sec and OFF for 0.5 sec, resulting in a 1 second blink period. Program the `A2B_PWMBLINK1` and `A2B_PWMBLINK2` registers to set the blink rate.

Duty Cycle

The PWM module controls the brightness (or color) of the LED. It controls the duty cycle of the output current; the amount of time the power is on relative to the total cycle time. For example, having the power on for only 50% of the cycle results in a reduction of brightness by 50%. More high time results in a brighter LED. Less high time results in a dimmer LED. When a 16-bit resolution is used, programming the `PWMnVAL` register with 0xFFFF results in a 100% duty cycle. Programming the register with 0x8000 results in a 50% duty cycle. Programming the register with 0x0000 results in a 0% duty cycle. The number of bits in the `PWMnVAL` register depends on the desired resolution. The resolution depends on the frequency selected. While the register is 16 bits, some frequency settings do not use one or more of the LSBs. The ***Effective Resolution*** table shows the effective resolution at different PWM frequencies.

The PWM logic runs at the sysclk rate (1 clock is 1-bit time on the bus). The PWM values are truncated at higher frequencies.

Table 5-26: Effective Resolution

PWM Frequency	Resolution of Value Used
187.5 Hz	16
375 Hz	16
750 Hz	16
1500 Hz	15
3 kHz	14
6 kHz	13
12 kHz	12
24 kHZ	11
48 kHz	10
96 kHz	9
192 kHz	8
Random	10

PWM Programming Concepts

To program the PWM interface, complete the following steps:

1. Disable the SPI interface by writing 0x02 to the `A2B_SPICFG` register.
2. Write 0x01 to the `A2B_MMRPAGE` register.
3. Enable the PWM channels by setting (=1) the appropriate bits in the `A2B_PWMCFG` register. For example, setting this register to 0b'00001010 results in enabling PWM channels 1 and 3.
4. Set the frequency of PWM channels using the `A2B_PWMFREQ` register. PWM channels 1-3 (`A2B_PWMFREQ.PWMPFREQ`) run on the same frequency whereas the PWMOE channel (`A2B_PWMFREQ.PWMOFREQ`) runs on a different frequency. By default, all PWM channels run at 192 kHz frequency when enabled.
5. Enable the frequency hopping scheme using the `A2B_PWMCFG.PWMORAND` and `A2B_PWMCFG.PWMPRAND` bits to select random frequencies between 187.5 Hz to 3 kHz.
6. Configure PWM channels with different blink rates (0, 0.5, 1.0, 1.5, 2.0 s) using the `A2B_PWMBLINK1` and `A2B_PWMBLINK2` registers.

7. Configure the duty cycle values of PWM channels using the A2B_PWM1VALL/ A2B_PWM2VALL/ A2B_PWM3VALL and A2B_PWM1VALH/A2B_PWM2VALH/A2B_PWM3VALH registers.

PWM Programming Example

Enable PWM channel 1 and the corresponding PWM functions using the following steps:

1. Write 0x01 to the A2B_MMRPAGE register to access PWM register space.
2. Write 0x0A to the A2B_PWMFREQ register to set the PWM channel 1 frequency to 187.5 Hz.
3. Write 0x04 to the A2B_PWMBLINK1 register to set the PWM channel 1 blink rate to a 2 second blink.
4. Write 0x00 to the A2B_PWM1VALL register to write the low byte of the output value first.
5. Write 0x04 to the A2B_PWM1VALH register to set the PWM channel 1 output value to 0x0400.
6. Write 0x01 to the A2B_PWMCFG to enable PWM channel 1.

The channel value and blink rate can be changed while a channel is running. All other parameters must be changed while the channel is disabled.

General Purpose Input/Output (GPIO) Pins

General Purpose Input/Output (GPIO) pins provide an interface between the A^{2}B transceiver and the local process-or or peripheral. The A^{2}B transceiver supports up to eight configurable GPIO pins. The *GPIO Pin Configuration* table shows the pins that are available for GPIO based on the mode selected in the A2B_PINCFG.GPIOSEL bit field.

Table 5-27: GPIO Pin Configuration

IO Bit	Pin Name		
	A2B_PINCFG.GPIOSEL= 0	**A2B_PINCFG.GPIOSEL** = 2	**A2B_PINCFG.GPIOSEL** = 3
IO0	SCK	SIO0	SIO4
IO1	SIO1	SIO1	SIO1
IO2	SIO2	SIO2	SIO2
IO3	SIO3	SIO3	SIO3
IO4	ADR1	ADR2	ADR2
IO5	MISO	SCL	SCL
IO6	MOSI	SDA	SDA
IO7	GPIO7	GPIO7	GPIO7

GPIO pins can be configured as input or output pins using the A2B_GPIOIEN and A2B_GPIOOEN registers. Before configuring the GPIO function on a given pin, verify whether the pin is available as GPIO. The *GPIO Pin*

Peripheral Functions table shows the conditions under which a pin can be used as GPIO when it is not used for any peripheral functions.

Table 5-28: GPIO Pin Peripheral Functions

Pin Name	Peripheral Functions		
	Peripheral Pin Name	**Pin Description**	**Pin Usage Conditions**
SIO0	DRX0	TDM data receive pin 0	When the TDM receive operation is enabled
	PDM0	PDM data receive pin 0	When PDM input is enabled on the SIO0/PDM0 pin
SIO1	DRX0	TDM data receive pin 0	When the TDM receive operation is enabled, with the SIO0 pin configured as a PDM pin
	DRX1	TDM data receive pin 1	When the TDM receive operation is enabled with at least two receive pins, and the SIO0 pin is configured as a DRX0 pin
	DTX3	TDM data transmit pin 3	When a 4-pin TDM transmit operation is enabled
	PDM1	PDM data receive pin 1	When PDM input is enabled on the SIO1/PDM1 pin
SIO2	DRX0	TDM data receive pin 0	When the TDM receive operation is enabled, with both SIO0 and SIO1 pins configured as PDM pins
	DRX1	TDM data receive pin 1	When the TDM receive operation is enabled with at least two receive pins, and either the SIO0 or SIO1 pin is configured as a PDM pin
	DRX2	TDM data receive pin 2	When the TDM receive operation is enabled with at least three receive pins, and: • the SIO0 pin is configured as DRX0 • the SIO1 pin is configured as DRX1
	DTX2	TDM data transmit pin 2	When the TDM transmit operation is enabled with at least three transmit pins and • the SIO4 pin is configured as DTX0 • the SIO3 pin is configured as DTX1
	$\overline{\text{ASPISS}}$	Alternate SPI slave select input	When the register-based full duplex SPI mode is enabled and the SIO2 pin is configured as the SPI slave select input pin (SPI is configured as the SPI slave)
	$\overline{\text{SPISSEL1}}$	SPI slave select output1	When the slave select output is enabled on the SIO2/$\overline{\text{SPISSEL1}}$ pin (SPI is configured as the SPI master)

Table 5-28: GPIO Pin Peripheral Functions (Continued)

Pin Name	Peripheral Functions		
	Peripheral Pin Name	Pin Description	Pin Usage Conditions
SIO3	DRX1	TDM data receive pin 1	When the TDM receive operation is enabled with at least two receive pins, with both SIO0 and SIO1 configured as PDM pins
	DRX2	TDM data receive pin 2	When the TDM receive operation is enabled with three receive pins, and either SIO0 or SIO1 is configured as a PDM pin
	DRX3	TDM data receive pin 3	When a 4-pin TDM receive operation is enabled
	DTX1	TDM data transmit pin 1	When TDM transmit is enabled with at least two transmit pins, and SIO4 is configured as DTX0
SIO4	DTX0	TDM data transmit pin 0	When the TDM transmit operation is enabled
SCL	SCL	I^{2}C serial clock	When I^{2}C operation is enabled
SDA	SDA	I^{2}C serial data	When I^{2}C operation is enabled
SCK	SCK	SPI serial clock	When SPI is enabled (as SPI slave/SPI master)
	PWM3	PWM3	When PWM 3 is enabled
MISO	MISO	SPI - master in slave out	When SPI is enabled (as SPI slave/SPI master)
	PWM1	PWM1	When PWM 1 is enabled
MOSI	MOSI	SPI - master out slave in	When SPI is enabled (as SPI slave/SPI master)
	PWM2	PWM2	When PWM 2 is enabled
ADR1	$\overline{\text{SPISS}}$	SPI slave select input	When SPI is enabled with the node transceiver configured as an SPI slave (except for the register-based full duplex SPI mode that is enabled with the SIO2/ADR2 pin configured as SPI slave select inputs)
	$\overline{\text{SPISSEL0}}$	SPI slave select output 0	When slave select output is enabled on ADR2/ $\overline{\text{SPISSEL0}}$ pin (SPI is configured as the SPI master)
	PWMOE	PWM output enable	When PWMOE is enabled
	CLKOUT1	Clock out pin 1	When clock out 1 functionality is enabled to source clock to peripherals

Table 5-28: GPIO Pin Peripheral Functions (Continued)

Pin Name	Peripheral Functions		
	Peripheral Pin Name	**Pin Description**	**Pin Usage Conditions**
ADR2	$\overline{\text{ASPISS}}$	Alternate SPI slave select input	When register-based full duplex SPI mode is enabled with the ADR2 pin configured as a SPI slave select input pin (SPI is SPI slave)
	$\overline{\text{SPISSEL2}}$	SPI slave select output 1	When slave select output is enabled on the ADR2/ $\overline{\text{SPISSEL2}}$ pin (SPI is configured as the SPI master)
	CLKOUT2	Clock out pin 2	When clock out 2 functionality is enabled to source clock to peripherals.
GPIO7	PDMCLK	Alternate PDM clock	When configured as an alternate PDM clock out for PDM operation
	RRSTRB	Reduced rate strobe pin	When configured as a reduced rate strobe pin in I^2S reduced rate mode

CAUTION: When using I^2C pins as GPIO outputs, these open-drain pins require an external pull-up resistor.

NOTE: The ADR1/CLKOUT1 and ADR2/CLKOUT2 pins can be used to source the clock to the peripherals. When SPI is enabled, the ADR1/CLKOUT1 or ADR2/CLKOUT2 pins do not function based on the GPIO mode.

- ADR1/CLKOUT1 is not functional when SPI is enabled with the non-default GPIO select (GPIO MODE=2/3)
- ADR2/CLKOUT2 is not functional when SPI is enabled with the default GPIO select (GPIO MODE=0)

The *GPIO Pins - ADR1/ADR2 Clock Source* table shows the pins available for GPIO when ADR1 or ADR2 are functioning as CLKOUT. The available pins are based on the SPI configuration and GPIO modes.

Table 5-29: GPIO Pins - ADR1/ADR2 Clock Source

SPI Mode	SPI Chip Select	GPIO Mode 0 (on SPI pins)		GPIO Mode 2/3 (on I^2C and RX/ TX pins)	
		CLKOUT1	**CLKOUT2**	**CLKOUT1**	**CLKOUT2**
SPI Slave	ADR1/$\overline{\text{SPISS}}$	NA	x	NA	ADR2
	SIO2/$\overline{\text{ASPISS}}$	x	x	x	ADR2
	ADR2/$\overline{\text{ASPISS}}$	x	NA	x	NA
SPI Master	ADR1/$\overline{\text{SPISSEL0}}$	NA	x	NA	ADR2
	SIO2/$\overline{\text{SPISSEL1}}$	ADR1	x	x	ADR2
	ADR2/$\overline{\text{SPISSEL2}}$	ADR1	NA	x	NA

GPIO Input

Any GPIO pin can be configured as a general-purpose input pin using the A2B_GPIOIEN register. The state of GPIO input pin is updated (low or high) in the A2B_GPIOIEN register. Setting the corresponding IO bit in the A2B_PINTEN register enables an interrupt on the GPIO input pin. All GP input interrupts (including subordinate node GP inputs) are reported on the main node IRQ signal only (conveyed through the A2B_INTSRC register). Set the corresponding IO bits in A2B_PINTEN and MASK bits in the A2B_INTMSK1 register to report the GPIO interrupts. In the main transceiver, set the corresponding IO bits to receive subordinate node GPIO interrupts using the A2B_INTMSK2.SLVIRQEN interrupt enable bit.

Upon input change on the GPIO pin and when the interrupt is enabled, this interrupt information is updated in the A2B_INTTYPE register of the main transceiver. By default, the interrupt is raised on the rising edge of the GPIO input pin. The interrupt triggering edge can be changed to falling edge by setting (=1) the corresponding IO bit in the A2B_PINTINV register.

The *GPIO Interrupt* shows how GPIO interrupts are handled on the main and subordinate nodes.

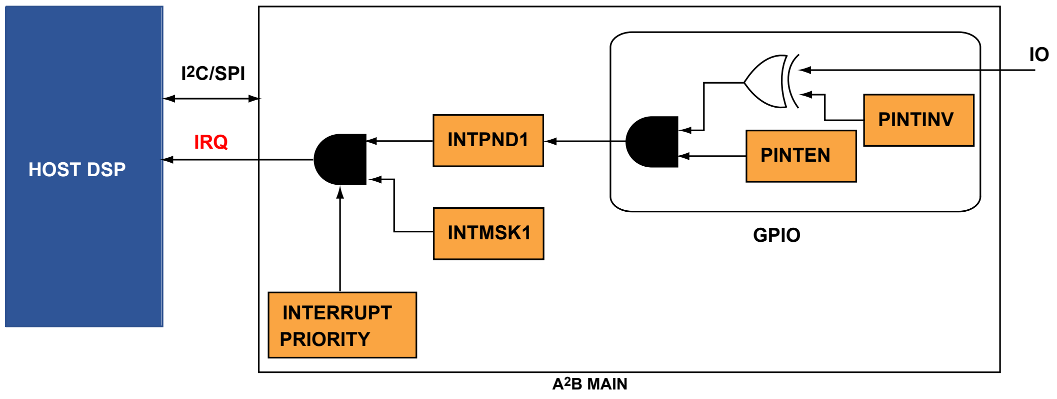


Figure 5-63: GPIO Interrupt Handling - Main Node

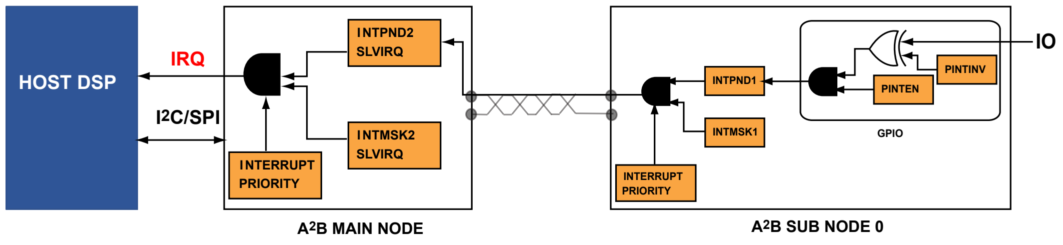


Figure 5-64: GPIO Interrupt Handling- Sub Node

GPIO Output

Any GPIO pin can be configured as a general-purpose output pin using the A2B_GPIOOEN register. The state of a GPIO output pin is driven low or high based on the corresponding IO bit enabled in the A2B_GPIODAT register.

The A2B_GPIODATCLR and A2B_GPIODATSET registers allow clearing or setting of individual GPIO output pins to 0 or 1 (write 1 to clear) without influencing the 0 or 1 value of other GPIO output pins. Read operations from these registers return the value in the GPIO output data register.

GPIO Over Distance

The GPIO over distance feature allows GPIO communication to occur over the A^{2}B bus without host intervention after initial programming. The host is only required to initialize the feature through virtual ports. GPIO over distance has the following features:

- Eight parallel 1-bit virtual ports, managed by the main node. The host processor can read the state of each virtual port in the A2B_GPIODDAT register.
- A flexible mapping scheme of GPIO pins to virtual ports 0 through 7. Any GPIO pin of any A^{2}B transceiver can be mapped to any virtual port.
- GPIO pins can be configured as inputs that update the contents of the A2B_GPIODDAT register or as outputs that reflect the contents of the A2B_GPIODDAT register.
- Output from one virtual port can be mapped to multiple GPIO output pins on different sub transceivers. When multiple virtual ports are mapped to one GPIO output pin, the values are OR'ed together.
- When multiple GPIO input pins are mapped to one virtual port, the values are OR'ed together even if they are from multiple nodes.

NOTE: If multiple nodes are updating the same virtual port, the A2B_GPIODINV register settings can be used to change the behavior from a wired OR to a wired AND. For example, to create a wired AND of multiple, active-high GPIO bits, the GPIO inputs and GPIO outputs must be inverted.

The A^{2}B transceiver internally communicates GPIO over distance information using the SCF and SRF. Subordinate node transceivers conveys the GP input info (upon input toggle) using SRFs to the A^{2}B main node. The main node transceiver evaluates the GPIO over distance virtual ports. If a GPIO output pin toggle (high to low /low to high) is required (based on the virtual port state), the main transceiver uses the SCF to communicate this data to the sub transceivers.

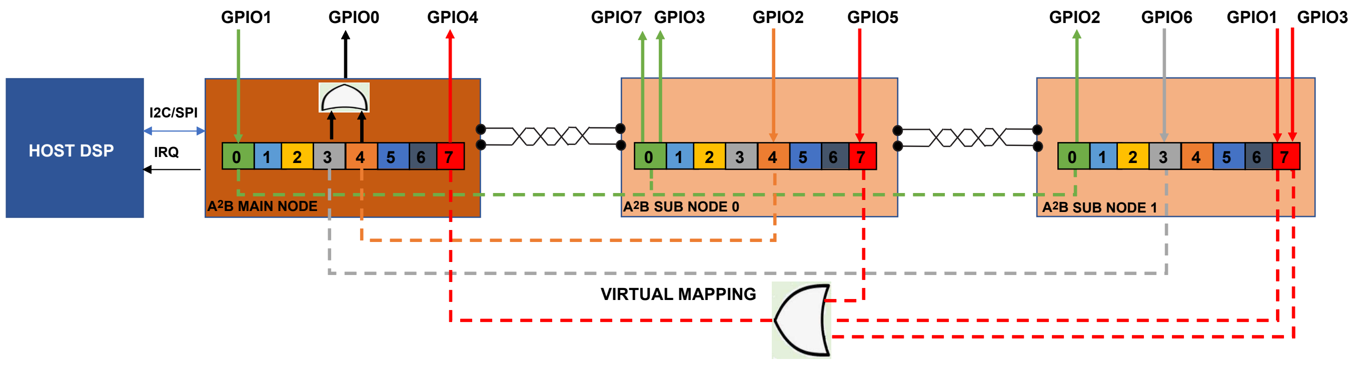


Figure 5-65: GPIO Over Distance

The *GPIO Over Distance* figure shows virtual mapping using GPIO pins. Multiple virtual ports are configured with IOs mapped to and from different transceivers. Data from an input GPIO pin connected to a virtual port is reflected to the output GPIO pin connected to same port on a different A^{2}B node.

- GPIO1 of the main node is mapped to GPIO7; GPIO3 of sub node 0 and GPIO2 of sub node 1 use virtual port 0. Therefore, the input signal on the GPIO1 pin of the main node is automatically reflected on GPIO7, GPIO3 of sub 0, and GPIO2 of sub 1.

  Use case example –This configuration can be used when the host processor must simultaneously control the mute signal of all the audio codecs connected to the different sub nodes. Traditionally, changing the GP output pins on the sub nodes requires that the host processor write to the `A2B_GPIODAT` register of respective sub nodes using an I^{2}C command. This operation can be simplified by mapping a GPIO pin of the main transceiver to the GP output pins of the sub node transceivers. Using pin mapping, the host processor only needs to toggle its GPIO pin to control the mute signal.

- Multiple GPIO input pins (GPIO1, GPIO3 of sub node 1, and GPIO5 of sub node 0) are mapped to virtual port 7. The output of the virtual port is mapped to the GPIO4 pin of the main node. The input signal on these three GPIO input pins is OR'ed and reflected on the GPIO4 pin of the main node.

  Use case example – This configuration can be used when the host processor must service interrupts from a remote peripheral device first (for example, amplitude overshoot/clipping interrupts from an audio codec or class D amplifier). Traditionally, the host processor relies on its IRQ interrupt; it must service the IRQ interrupt to understand the remote peripheral device interrupt. This sequence of operations can have significant delay. This use case can be simplified by mapping GPIO pins from the sub nodes to a GPIO pin of the main node. Therefore, whenever a sub node raises an interrupt (for example, toggles its GPIO pin), the notification is automatically reflected on GPIO pin of main node. The host processor can interpret the interrupt immediately and take action.

- Multiple virtual ports (port 3 and port 4) are mapped to a single GPIO output pin (GPIO0) of the main node. In this case, the values of both virtual ports are OR'ed and reflected on GPIO0 of the main node.

NOTE: All virtual ports are evaluated and managed by the A^{2}B main node only.

Mapping Multiple GPIO Inputs to One Virtual Port

When more than one node has a GPIO input mapped to the same virtual port, the protocol treats the input pins as a wired OR into the virtual port. When the virtual port is low (inactive), any request to set the virtual port results in a command from the main node to update all of the A2B_GPIODDAT registers across the system.

When the virtual port is high (active), any request to clear the virtual port results in a special command from the main node to notify all of the sub nodes of the request. If any of the sub nodes reject the request, the main node sees the rejection of the request, and the A2B_GPIODDAT registers retain their values. If none of the sub nodes reject the request, the main node sees an acceptance of the request and follows up with the updated A2B_GPIODDAT value. A wired AND can be used instead of wired OR by inverting all GPIO inputs and GPIO outputs using the A2B_GPIODINV register.

Programming GPIO Over Distance

When a pin is available as GPIO, the GPIO over distance feature is enabled by setting the appropriate enable bit in the GPIO over distance enable (A2B_GPIODEN) register. See the *GPIO Pin Peripheral Functions* table to check pin availability. When a bit is set (=1), the corresponding GPIO pin can then be mapped to one or more GPIO over distance virtual ports using the GPIO over distance mask registers (A2B_GPIOD0MSK through A2B_GPIOD7MSK). These registers correspond to GPIO-capable pins IO0 through IO7, respectively. Bits 0 through 7 in these registers correspond to virtual ports 0 through 7, respectively. If a bit is set in of these registers, it maps the GPIO pin associated with the register to the corresponding virtual port.

When GPIO over distance is enabled, the direction of the pin is controlled exclusively using the GPIO output enable register (A2B_GPIOOEN). The direction is not controlled by a combination of this register and the complementary GPIO input enable register (A2B_GPIOIEN). When a bit in the A2B_GPIOOEN register is set (=1), the associated GPIO pin becomes an output for GPIO over distance. If the bit is cleared (=0) in the A2B_GPIOOEN register, the associated GPIO pin becomes an input to GPIO over distance. It is not necessary to program the A2B_GPIOIEN register when using GPIO over distance for the pins.

If the GPIO pin is an input (the associated bit in A2B_GPIOOEN = 0), the local node updates the virtual ports associated with the bits in the corresponding GPIO over distance mask registers (A2B_GPIOD0MSK.IOD0MSK through A2B_GPIOD7MSK.IOD7MSK). The virtual port values can be read in the GPIO over distance data register (A2B_GPIODDAT).

If the GPIO pin is an output (the associated bit in A2B_GPIOOEN = 1), the virtual ports that are mapped to that pin, as determined by the set bits in the associated GPIO over distance mask registers (A2B_GPIOD0MSK.IOD0MSK through A2B_GPIOD7MSK.IOD7MSK) are OR'ed together to produce the GPIO output value (the logic OR of the corresponding bits in the A2B_GPIODDAT register).

NOTE: The A2B_GPIODDAT register is read only. It is recommended that the host always read this register from the main node.

The GPIO over distance inversion register (A2B_GPIODINV) allows for inversion of a GPIO pin input or output. When a bit is set in this register, the associated GPIO pin signal is inverted. The inversion is applied on the way in from the pin when the GPIO pin is an input to a virtual port (the associated bit in A2B_GPIOOEN = 0). The inversion is applied on the way out to the GPIO pin when the pin is an output from a virtual port (the associated bit in `A2B_GPIOOEN` = 1).

If multiple nodes are updating the same virtual port, the `A2B_GPIODINV` register settings can be used to change the behavior from a wired OR to a wired AND. For example, to create a wired AND of multiple, active-high GPIO bits, the GPIO inputs and GPIO outputs must be inverted.

The following sections describe pin mapping cases that use GPIO over distance.

Mapping the Main Node MOSI/GPIO6 Pin to the Subordinate Node 2 SIO1/GPIO1 Pin

The following procedure describes how to map the main node MOSI/GPIO pin to the subordinate node 2 SIO1/GPIO1 pin.

1. Write 0x04 to the main node `A2B_GPIOD6MSK` register to map the MOSI/GPIO6 pin to virtual port 2.
2. Write 0x40 to the main node `A2B_GPIODEN` register to enable GPIO over distance access on the MOSI/GPIO6 pin.
3. Write 0x02 to the subordinate node 2 `A2B_GPIOOEN` register to enable GPIO output for the SIO1/GPIO1 pin.
4. Write 0x04 to the subordinate node 2 `A2B_GPIOD1MSK` register to map virtual port 2 to the SIO1/GPIO1 pin.
5. Write 0x02 to the subordinate node 2 `A2B_GPIODEN` register to enable GPIO over distance access on the SIO1/GPIO1 pin.

Mapping the Subordinate Node 1 ADR1/GPIO4 Pin to the Main Node SIO1/GPIO1 Pin

The following procedure describes how to map the subordinate node 1 ADR1/GPIO4 pin to the main node SIO1/GPIO1 pin.

1. Write 0x10 to the subordinate node 1 `A2B_GPIOD4MSK` register to map the ADR1/GPIO4 pin to bus GPIO port 4.
2. Write 0x10 to the subordinate node 1 `A2B_GPIODEN` register to enable GPIO over distance access on the ADR1/GPIO4 pin.
3. Write 0x02 to the main node `A2B_GPIOOEN` register to enable GPIO output for the SIO1/GPIO1 pin.
4. Write 0x10 to the main node `A2B_GPIOD1MSK` register to map bus GPIO port 4 to the SIO1/GPIO1 pin.
5. Write 0x02 to the main node `A2B_GPIODEN` register to enable GPIO over distance access on the SIO1/GPIO1 pin.

Mapping the SIO1/GPIO1 Pins on Subordinate Nodes 0 Through 2 to the Main Node SIO1/GPIO1 Pin

The following procedure describes how to map the SIO1/GPIO1 pin on sub nodes 0 through 2 to the main node SIO1/GPIO1 pin.

1. For sub nodes 2, 1, and 0, write 0x01 to the `A2B_GPIOD1MSK` register to map the SIO1/GPIO1 pin of each sub node to bus GPIO port 0.
2. For sub nodes 2, 1, and 0, write 0x02 to the `A2B_GPIODEN` register to enable GPIO over distance access on the SIO1/GPIO1 pin of each sub node.
3. Write 0x02 to the main node `A2B_GPIOOEN` register to enable GPIO output for the SIO1/GPIO1 pin.
4. Write 0x01 to the main node `A2B_GPIOD1MSK` register to map bus GPIO port 0 to the SIO1/GPIO1 pin.
5. Write 0x02 to the main node `A2B_GPIODEN` register to enable GPIO over distance access on the SIO1/GPIO1 pin.

GPIO Over Distance Latency

GPIO over distance latency depends on the input state. When set, (0-1) operation, the change is latched during a superframe and data is sent to the receiving node in the next superframe. The GPIO on the receiving node is set at the end of this superframe.

For a clear (1-0) operation, the change is latched during a superframe and the signal is sent to the receiving node in the next superframe. The operation requires one extra superframe to validate the other nodes input when it is a wired OR to determine whether a clear can occur. The GPIO on the receiving node is set at the end of this superframe.

The *GPIO Over Distance Latency* figures shows the timing for set and clear operations for GPIO over distance. The main GPIO is sub node 0 GPIO or sub node 1 GPIO. Logic functions like a wired OR/AND permit set operations to be updated in the next superframe. However, clear operations require an extra superframe to allow all of the nodes to determine whether a clear operation can occur.

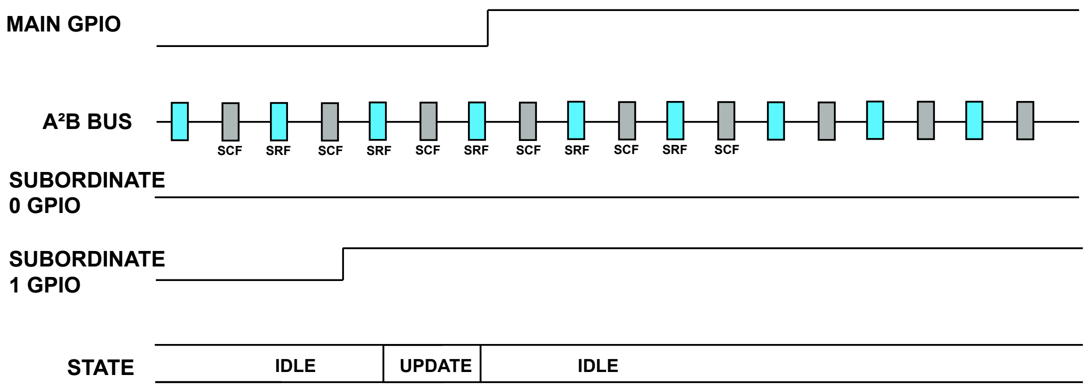


Figure 5-66: GPIO Over Distance Latency – Set Operation

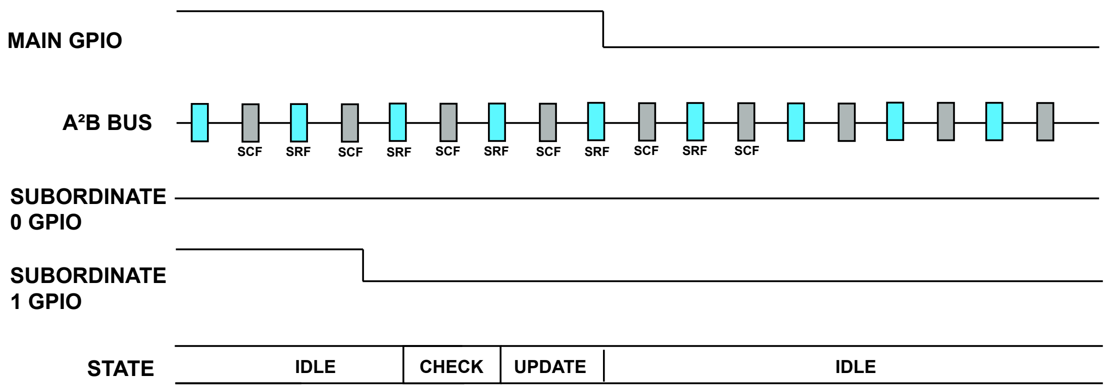


Figure 5-67: GPIO Over Distance Latency – Clear Operation

Mailboxes

The A^{2}B transceiver has two dedicated mailboxes to support interrupt driven, bidirectional message exchange between local processors at different subordinate nodes and the host connected to the main node. The mailboxes can be used to customize handshaking among numerous nodes in a system to coordinate system events, such as synchronizing audio.

There are two virtual mailboxes, MBOX0 and MBOX1, that allow for inter-processor communication between the host and a subordinate node control processor. Message exchange directly between subordinate node processors is not possible.

NOTE: Throughout this section, all specific references to MBOX0 also apply to the MBOX1 instance.

Mailbox Programming and Operation

Each mailbox is a 4-byte data buffer used to keep messages to be exchanged. These messages are accessible as mailbox data byte 0 through 3 registers (`A2B_MBOX0B0` - `A2B_MBOX0B3`). The length of the mailbox can be configured to hold 8-, 16-, 24-, or 32-bit messages, as configured in the `A2B_MBOX0CTL.MB0LEN` field. It determines which of the four mailbox data registers to use for the message exchange. The first byte is always in the mailbox data byte 0 (`A2B_MBOX0B0`) register; the final byte is in the highest data register required to accommodate the programmed data length, as shown in the following table.

Mailbox Length (Bytes)	Final Byte of Mailbox
1	Byte 0
2	Byte 1
3	Byte 2
4	Byte 3

NOTE: The mailbox data registers are present in the subordinate node only and are not present in main node.

The direction of each mailbox can be independently configured as a transmit mailbox in which a local controller can send messages to the host processor. Or, it can be configured as a receive mailbox in which the host processor can send messages to the local processor connected to the subordinate node. The mailbox control register (`A2B_MBOX0CTL`) provides bit fields to enable the mailbox and control direction, message length, and interrupt capabilities.

CAUTION: Dynamic reconfiguration of an enabled mailbox (`A2B_MBOX0CTL.MB0EN` = 1) is forbidden. The host must first disable the mailbox (`A2B_MBOX0CTL.MB0EN` = 0) and then re-enable it in two separate accesses if reconfiguration is required.

NOTE: Reading the mailbox data registers (A2B_MBOXnBn) is the only way to clear or to update new data into the mailbox data registers.

Transmit Mailbox

The processor in the subordinate node writes the message to the mailbox data registers of local subordinate transceiver using I^2C/SPI. In the main node, the host processor is informed about the new message by an interrupt on the IRQ pin of the main transceiver. The `A2B_INTTYPE` and `A2B_INTSRC` registers are updated with the corresponding interrupt. The host can read out the message from A^2B subordinate transceiver registers over I^2C/SPI.

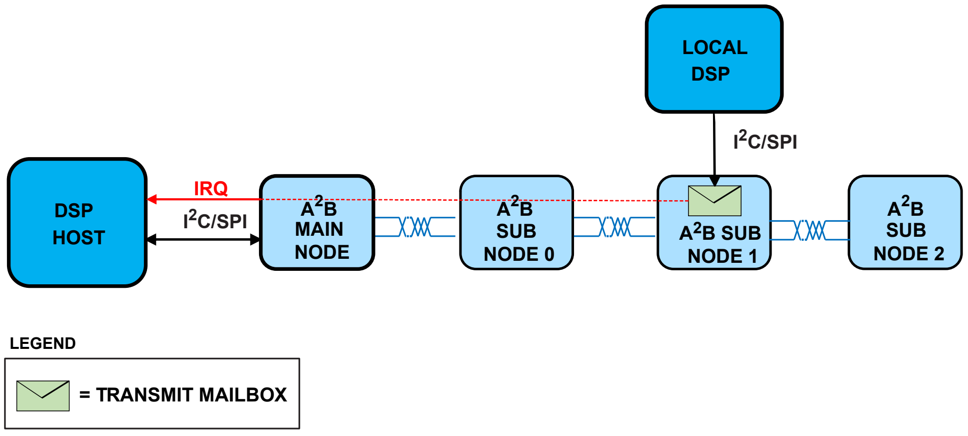


Figure 5-68: Transmit Mailbox Communication

The mailbox offers the following interrupt functionality.

- Mailbox full interrupt is enabled (`A2B_MBOX0CTL.MB0FIEN` = 1) – after the final byte of the mailbox is written by the local processor, the subordinate node internally informs the main node to generate the mailbox full interrupt to the host processor. The interrupt indicates to the host processor that a new message has been received from the local processor and is available to read.

  NOTE: The host processor should only read the mailbox data registers when the mailbox is full. Confirm the mailbox status with the mailbox full interrupt or by checking the mailbox status register.

- Mailbox empty interrupt is enabled (A2B_MBOX0CTL.MB0EIEN = 1) – after the final byte of the mailbox is read by the host processor, the subordinate node raises the interrupt on the local IRQ pin to the local processor. The interrupt indicates to the local processor that the previous message has been read by the host processor; the host processor is ready to receive a new message.

  NOTE: The local processor should only write the mailbox data registers when the mailbox is empty. Confirm the mailbox status with the mailbox empty interrupt or by checking the mailbox status register.

The A2B_MBOX0STAT.MB0EIRQ and A2B_MBOX0STAT.MB0FIRQ bits are set when the mailbox signals the mailbox empty or full interrupt to the host or local processor; the bits are cleared when the interrupt is processed by the host or local processor.

NOTE: The A2B_MBOX0STAT.MB0FULL and A2B_MBOX0STAT.MB0EMPTY bits are set upon the mailbox full and mailbox empty events even when the A2B_MBOX0CTL.MB0FIEN and A2B_MBOX0CTL.MB0EIEN bits are not set.

Receive Mailbox

If the host processor wants to send a message to a local processor of a subordinate node, it places the message in the mailbox data registers of the corresponding subordinate node. The host processor can access the subordinate node mailbox data registers over the A^{2}B bus using I^{2}C/SPI. In the subordinate node, the processor is informed of this new message by an interrupt on the IRQ pin of the subordinate transceiver. It can directly read out the message over I^{2}C/SPI from the local A^{2}B subordinate transceiver registers after checking the A2B_LINTTYPE register.

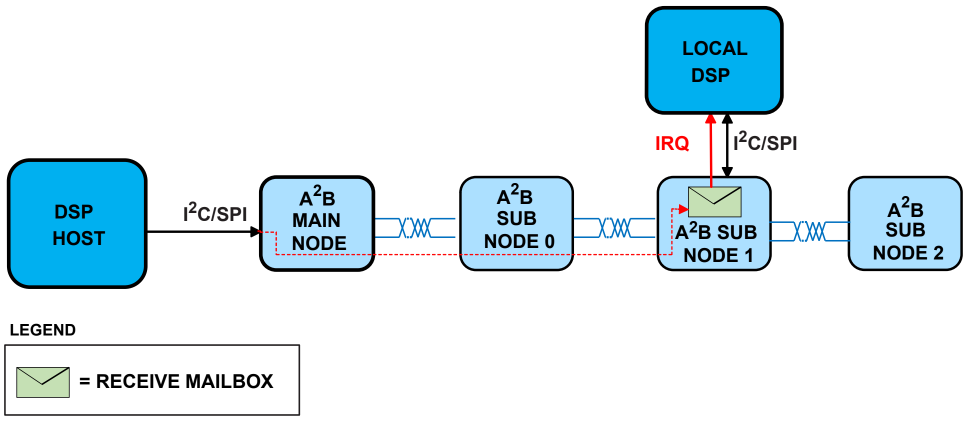


Figure 5-69: Receive Mailbox Communication

The mailbox offers the following interrupt functionality:

- Mailbox full interrupt is enabled (A2B_MBOX0CTL.MB0FIEN = 1) – after the host processor writes the final byte of the mailbox and the data is received by A^{2}B subordinate transceiver, the subordinate node raises the interrupt on the local IRQ pin to the local processor. The interrupt indicates to the local processor that a new message has been received from the host processor and is available to read.

NOTE: The local processor should only read the mailbox data registers when the mailbox is full. Confirm the mailbox status with the mailbox full interrupt or by checking the mailbox status register.

- Mailbox empty interrupt is enabled (`A2B_MBOX0CTL.MB0EIEN` = 1) – after the final byte of the mailbox is read by the local processor, the subordinate node internally informs the main node to generate the empty interrupt to the host processor. The interrupt indicates to the host processor that the previous message has been read by the local processor; the local processor is ready to receive a new message.

NOTE: The host processor should only write the mailbox data registers when the mailbox is empty. Confirm the mailbox status with the mailbox empty interrupt or by checking the mailbox status register.

Using Mailbox in Multiple Subordinate Nodes

Multiple subordinate nodes can communicate to the main node through their transmit mailboxes. In the main node, the `A2B_INTTYPE` register contains information about the pending interrupt generated by any subordinate node; the subordinate node that generated the interrupt is indicated in the `A2B_INTSRC` register.

When two subordinate nodes write to their mailboxes simultaneously, the main gets the interrupt indication from the subordinate that is closer to the main node. Upon detecting the interrupt, the host extracts the interrupt information by reading the interrupt type (`A2B_INTTYPE`) and interrupt source (`A2B_INTSRC`) registers of the main transceiver to determine which interrupt occurred and which subordinate node generated it, respectively. Upon reading the `A2B_INTTYPE` register, the interrupt request for that interrupt is cleared in the subordinate node identified by the value in the `A2B_INTSRC` register. The IRQ pin toggles to the deasserted state and then immediately back into the asserted state due to the still active interrupt from the other subordinate node. The host can again read the `A2B_INTTYPE` and `A2B_INTSRC` registers of the main transceiver to acknowledge the mailbox interrupt of the other subordinate node.

Mailbox Latency

The mailbox transactions are made up of register reads and writes over the I^2C/SPI bus. The interrupt request from a subordinate node to the main is part of the SRF packet, so the latency on the subordinate node to main node mailbox can include an extra superframe waiting for this time.

The following figures show the system timing for the mailbox transactions in both directions. The light gray slots indicate the SCF field, and the dark gray slots indicate the SRF field.

As shown in the ***I^2C Mailbox Latency (from Host to Subordinate)*** figure, when the mailbox message is from the host to a subordinate processor, the host processor writes the mailbox data to the A^2B subordinate node through the SCF field using a 2-byte burst write access to the main transceiver BUS_ADDR device address. When the writes complete, the subordinate transceiver immediately generates the interrupt to its local node processor. As a result, the subordinate node interrupt request (SUB-NODE IRQ) asserted on IRQ aligns with the SCF field. Once this interrupt is asserted, the locally-connected processor can use the I^2C/SPI local register access to interrogate the `A2B_LINTTYPE` register to determine that it is the mailbox full interrupt, after which it can then extract the data from the mailbox data registers using I^2C/SPI. Once those transactions finish, the mailbox empty interrupt is generated at the main node (MAIN-NODE IRQ), aligned with the SRF field, and the host proceeds with reading the

A2B_INTSRC and A2B_INTTYPE registers of the main transceiver (using the main transceiver BASE_ADDR device address) to determine that it is the mailbox empty interrupt originating with the indicated subordinate node.

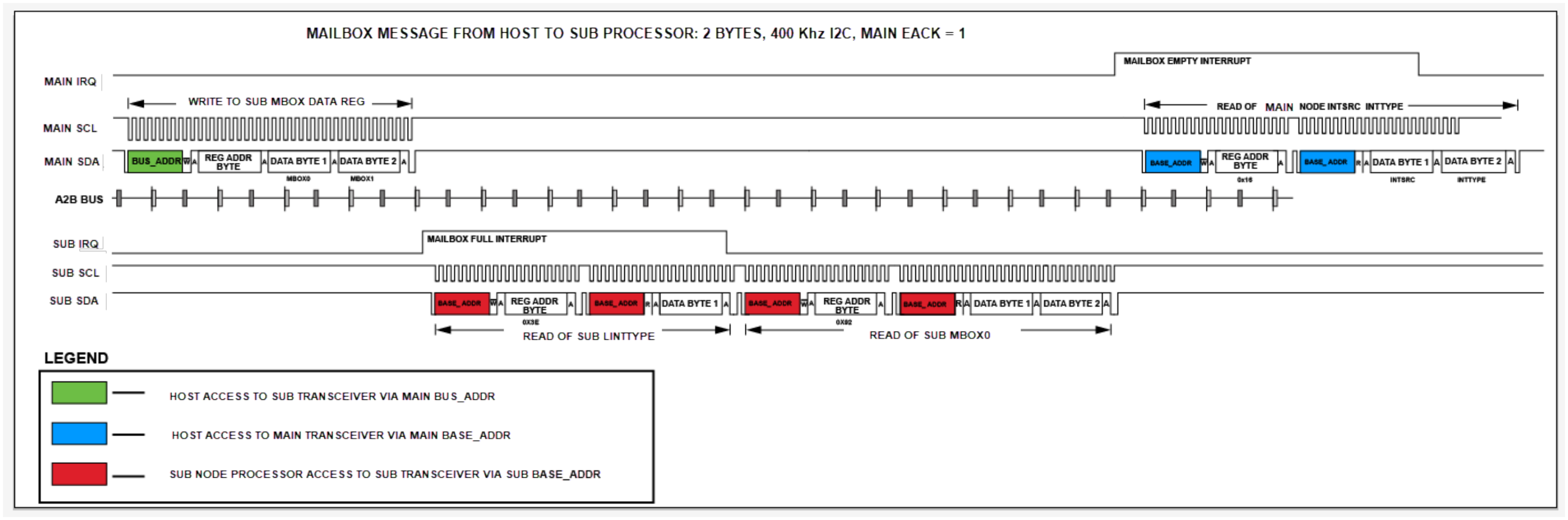

MAILBOX MESSAGE FROM HOST TO SUB PROCESSOR: 2 BYTES, 400 Khz I2C, MAIN EACK = 1
MAIN IRQ
MAILBOX EMPTY INTERRUPT
WRITE TO SUB MBOX DATA REG
READ OF MAIN NODE INTSRC INTTYPE
MAIN SCL
MAIN SDA
BUS_ADDR
REG ADDR BYTE
DATA BYTE 1
DATA BYTE 2
MBOX0
MBOX1
BASE_ADDR
INTSRC
INTTYPE
A2B BUS
SUB IRQ
MAILBOX FULL INTERRUPT
SUB SCL
SUB SDA
READ OF SUB LINTTYPE
READ OF SUB MBOX0
LEGEND
HOST ACCESS TO SUB TRANSCEIVER VIA MAIN BUS_ADDR
HOST ACCESS TO MAIN TRANSCEIVER VIA MAIN BASE_ADDR
SUB NODE PROCESSOR ACCESS TO SUB TRANSCEIVER VIA SUB BASE_ADDR

Figure 5-70: I^{2}C Mailbox Latency (from Host to Subordinate)

Similarly, as shown in the ***I^{2}C Mailbox Latency (from Subordinate to Host)*** figure, when the mailbox message is from a subordinate processor to the host, the subordinate node processor populates the mailbox data registers at any time by issuing writes to the registers using I^{2}C/SPI local register writes, and the interrupt indication to the main A^{2}B node goes through the SRF field. As a result, the main mailbox full interrupt request (MAIN-NODE IRQ) asserted on IRQ aligns with the SRF field. Once this interrupt is asserted, the host (using the main transceiver BASE_ADDR device address) interrogates the A2B_INTSRC and A2B_INTTYPE registers to determine that it is the mailbox full interrupt originating with the indicated subordinate node.

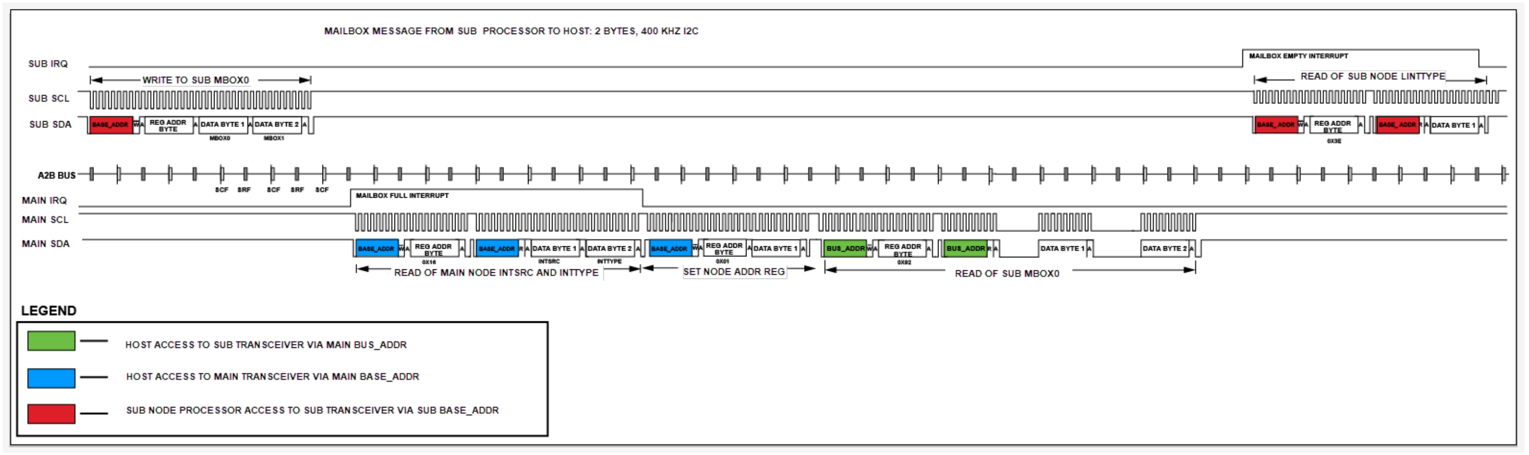

MAILBOX MESSAGE FROM SUB PROCESSOR TO HOST: 2 BYTES, 400 KHZ I2C
SUB IRQ
MAILBOX EMPTY INTERRUPT
WRITE TO SUB MBOX0
READ OF SUB NODE LINTTYPE
SUB SCL
SUB SDA
A2B BUS
SCF
SRF
MAIN IRQ
MAILBOX FULL INTERRUPT
MAIN SCL
MAIN SDA
READ OF MAIN NODE INTSRC AND INTTYPE
SET NODE ADDR REG
READ OF SUB MBOX0
LEGEND
HOST ACCESS TO SUB TRANSCEIVER VIA MAIN BUS_ADDR
HOST ACCESS TO MAIN TRANSCEIVER VIA MAIN BASE_ADDR
SUB NODE PROCESSOR ACCESS TO SUB TRANSCEIVER VIA SUB BASE_ADDR

Figure 5-71: I^{2}C Mailbox Latency (from Subordinate to Host)

To extract the data from the mailbox of the subordinate transceiver, the host must read the mailbox data byte registers of the subordinate transceiver using an I^{2}C remote access or an SPI remote register read request. Once the last byte is read by the host, the mailbox empty interrupt request of the subordinate node (SUB-NODE IRQ) is asserted in the next SCF. Then, the subordinate node processor can read the A2B_LINTTYPE register and take action after identifying that it was the mailbox empty interrupt that occurred (for example, load the mailbox data registers again to restart the process).

NOTE: When mailbox registers are accessed over SPI, the related register access time between the host and the main node transceiver and/or between the subordinate node transceiver and the local processor differs from the ***I^{2}C Mailbox Latency*** figures. The latency on the A^{2}B bus, when accessed through SPI, is the same latency as an I^{2}C access; the SPI register access passes via the SCF and SRF.

Transceiver Reset

The A^{2}B transceivers have three primary reset sources:

- Power on reset
- Hardware reset
- Soft reset

Power on Reset

Power on reset is generated by an internal reset circuit that monitors the state of VIN, DVDD, IOVDD and TRXVDD. This reset is held asserted until the VIN, DVDD, IOVDD and TRXVDD voltages transition above their reset deassertion voltages. The node goes into reset if VIN, DVDD, IOVDD, or TRXVDD transition below their reset assertion voltages.

Hardware Reset

The transceiver features a dedicated active low hardware reset pin to reset the device. The reset pin can be deasserted after all the power domains are stable, thus eliminating the need for power-up sequencing.

All A^{2}B system registers are held in reset and the transceiver state is held at reset until both the hardware reset and power on reset are deasserted.

Soft Reset

Software can reset the protocol engine of the transceiver and return every register in the A^{2}B node to their reset state (except the `A2B_CONTROL`, and `A2B_BMMCFG` registers). A soft reset can be applied by writing `A2B_CONTROL.SOFTRST` = 1. When a soft reset is applied to the main node, the external MOS switch opens, resulting in the dropping of subordinate nodes in the A^{2}B chain due to loss of PLL synchronization (SCF).

In the main node, when a soft reset is applied and `A2B_CONTROL.MSTR` = 1, there is no effect on the PLL state machine; the PLL remains locked to the incoming SYNC signal. But, if a soft reset is applied and `A2B_CONTROL.MSTR` = 0, the node is no longer the main node. This condition causes the PLL to unlock and the transceiver, subsequently, enters the SUSTAIN state.

In the subordinate node, a soft reset can be applied locally from a local processor by writing to the `A2B_CONTROL` register using I^{2}C. If the A^{2}B node is operational (PLL locked), the soft reset causes the PLL to unlock and the node enters the SUSTAIN state. The transceiver requires a wait time of 1024 x SYNC periods to come out of sustain mode and the system resets.

NOTE: There is no extra delay needed for a soft reset to take effect. It takes effect immediately upon the completion of the I^2C access in which the soft reset command is received.

CAUTION: A soft reset does not take effect when the node is in the power-up state.

NOTE: Although not a primary option, the main node transceiver can be reset by removing the SYNC signal. The subordinate transceiver can be reset by opening the switch of the upstream node which stops sending the SCF to the subordinate node. The signal goes to the PLL input of the transceiver; any break in transmission causes the PLL to unlock. When the PLL unlocks, the registers return to the RESET state (except the `A2B_CONTROL` and `A2B_BMMCFG` registers).

Voltage Monitor ADC

The Voltage Monitor (VMTR) ADC allows the monitoring of up to five different supply voltages (V_{IN}, V_{BUS}, V_{DVDD}, V_{IOVDD} , and V_{TRXVDD}), downstream high-side bus current (V_{VBUS}- $V_{ISENSEP}$) and low-side bus current ($V_{ISENSEN}$- $V_{VSENSEN}$) . When a voltage domain is enabled for monitoring, the VMTR samples the voltage level present on the associated pin, as shown in the *Voltage Monitor Sampling* figure.

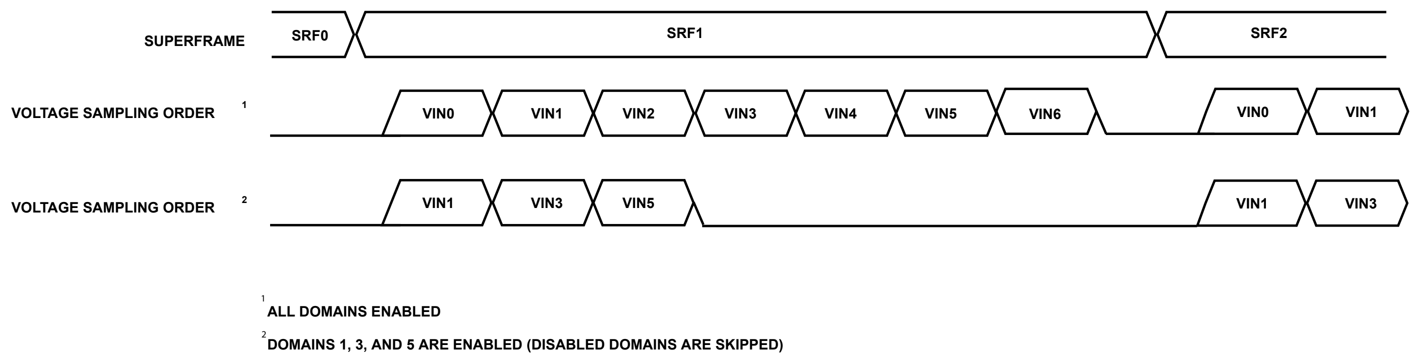


Figure 5-72: Voltage Monitor Sampling

The VMTR updates the corresponding measured voltage register (`A2B_VMTR_VLTG0` - `A2B_VMTR_VLTG6`) in each superframe.

If an interrupt is enabled, the VMTR generates an interrupt when the input voltage being monitored drops below a minimum threshold or exceeds a maximum threshold.

Programming the Voltage Monitor ADC

To program the voltage monitor ADC, complete the following steps:

1. Write 0x01 to the A2B_MMRPAGE register.
2. Enable the VMTR to monitor the desired voltage domains by setting (=1) the appropriate bits in the enable voltage measurement (A2B_VMTR_VEN) register. For example, setting this register to 0b00101010 results in the voltage domains associated with 1, 3, and 5 to be monitored, as depicted in the *Voltage Monitor Sampling* figure.
3. Configure the minimum voltage threshold registers (A2B_VMTR_VMIN0 through A2B_VMTR_VMIN6) and the maximum voltage threshold registers (A2B_VMTR_VMAX0 through A2B_VMTR_VMAX6) to configure the interrupt trip points for each enabled domain. By default, the minimum threshold values are 0x00 and the maximum threshold values are 0xFF.
4. Set (=1) the appropriate bits in the interrupt enable register (A2B_VMTR_INTEN) to enable VMTR interrupts. The A2B_VMTR_INTEN register is ignored when the corresponding domain is not enabled for monitoring (when the associated bit in the A2B_VMTR_VEN register is 0).
5. The measured voltage register (A2B_VMTR_VLTG0 - A2B_VMTR_VLTG6), minimum voltage check result status bit (A2B_VMTR_MNSTAT[n]), and maximum voltage check result status bit A2B_VMTR_MXSTAT[n]) are updated when voltage sampling completes. These registers are updated once per superframe, and reads from these registers return the most recently measured values. The A2B_VMTR_VLTG0 - A2B_VMTR_VLTG6 registers indicate the monitored voltage. A set bit in the A2B_VMTR_MNSTAT register indicates that the associated measured voltage was less than the minimum threshold. A set bit in the A2B_VMTR_MXSTAT register indicates that the associated measured voltage exceeded the maximum threshold.

   NOTE: When a given domain is not enabled for monitoring in the A2B_VMTR_VEN register, reads of the associated A2B_VMTR_VLTG0 - A2B_VMTR_VLTG6 register return 0x00.

   IMPORTANT: The A2B_VMTR_MXSTAT and A2B_VMTR_MNSTAT registers are write-1-clear registers. Once an interrupt is generated and appropriate action is taken, the host must write 1 to clear the registers.

VMTR ADC I/O Voltage Ranges

When programming and reading the VMTR registers, 8-bit binary encoding is used to represent voltage levels within the *Full Scale Voltage Range* associated with the pin being monitored:

- 0x00 represents the lower limit of the *Full Scale Voltage Range* (0V)
- 0x01 - 0xFE represent incremental voltage levels within the *Full Scale Voltage Range*
- 0xFF represents the upper limit of the *Full Scale Voltage Range*

Because there are 2^8 discrete values (ranging from 0x00 to 0xFF), an *Incremental Step Value* within the *Full Scale Voltage Range* is a function of the upper voltage limit. For example, if the upper limit is 32 V, the *Incremental Step Value* for the full voltage range is 32 V/2^8 = 125 mV. The *VMTR ADC Voltage Range* table summarizes the fixed voltage range associated with each input pin being monitored, and the resulting fixed incremental step values within the range provided.

Table 5-30: VMTR ADC Voltage Range

A2B_VMTR_VEN Bit	Input	Fixed Full Scale Range [V]	Incremental Step Value [mV]
0	VIN – GND	0 – 16	62.5
1	VBUS – GND	0 – 32	125
2	IOVDD – GND	0 – 4	15.625
3	TRXVDD – GND	0 – 4	15.625
4	DVDD – GND	0 – 4	15.625
5	ISENSEN – VSENSEN	0 – 0.24 *1	0.94
6	VBUS – ISENSEP	0 – 0.24 *2	0.94

*1 Upper limit depends on the current across the sense resistor between the ISENSEN and VSENSEN pins

*2 Upper limit depends on the current across the sense resistor between the VBUS and ISENSEP pins

ATTENTION: The *Full Scale Voltage Range* in the *VMTR ADC Voltage Range* table is fixed to fully accommodate the valid specification ranges for the associated pins, but the actual upper limits are governed by the maximum specifications. Consult the transceiver data sheet to identify the meaningful voltage range when programming the VMTR registers.

Example: Measurement of Voltage 6

As shown in the *VMTR ADC Voltage Range* table, the range for this monitored voltage is a function of the installed sense resistor between the VBUS and ISENSEP pins. According to the A2B_VMTR_VEN[6] row:

- 0x00 is the 8-bit encoding for the minimum voltage range value (0 V)
- 0xFF is the 8-bit encoding for the maximum voltage range value (0.24 V)
- For 2^8 discrete binary encodings within this range, the incremental step value is 0.94 mV (0.24 V / 2^8)

1. Because 0xFF encodes to 240 mV, several encodings may be above the maximum voltage that can be sensed on the pin, depending on the external circuit. The upper limit for the sensed voltage must first be calculated in order to select a valid maximum threshold value. If the sense resistor between the VBUS and ISENSEP pins is 50 mΩ and the current across it is 2 A, the maximum voltage that can be sensed is 50 mΩ 2A = 100 mV.

   Using the fixed incremental step value of 0.94 mV, 100 mV is encoded as 100/0.94 = 106.38, which must be rounded down to 106 (0x6A).

   NOTE: For this external circuit, the maximum value that has meaning in the maximum threshold register is 0x6A, and the minimum threshold values range from 0x00 0x69.

2. Program the VMTR registers:

   A2B_VMTR_VEN= 0x40 (set bit 6 to enable monitoring of voltage 6)

   A2B_VMTR_INTEN= 0x40 (set bit 6 to enable VMTR interrupt for voltage 6)

A2B_VMTR_VMIN6 ≤ 0x69 (set minimum voltage trip point)

A2B_VMTR_VMAX6 ≥ 0x6B (set maximum voltage trip point)

3. Read the A2B_VMTR_VLTG6 register to obtain the measured voltage and the A2B_VMTR_MXSTAT[6] and A2B_VMTR_MNSTAT[6] bits for maximum and minimum voltage exceeded errors, respectively.

Example: Measurement of Voltage 0

According to the A2B_VMTR_VEN[0] row in the *VMTR ADC Voltage Range* table:

- 0x00 is the 8-bit encoding for the minimum voltage range value (0 V)
- 0xFF is the 8-bit encoding for the maximum voltage range value (16 V)
- For 2^8 discrete binary encodings within this range, the incremental step value is 62.5 mV (16 V / 2^8)

1. Use the incremental step value to determine the binary encoding for the desired maximum voltage threshold. For example, to set the maximum voltage threshold to 10.1 V:

   10.1 V/62.5 mV = 161.6, which must be rounded down to 161 (0xA1)

2. Use the incremental step value to determine the binary encoding for the desired minimum voltage threshold. For example, to set the minimum voltage threshold to 3.7 V:

   3.7 V/62.5 mV = 59.2; which must be rounded down to 59 (0x3B)

3. Program the VMTR registers:

   A2B_VMTR_VEN= 0x01 (set bit 0 to enable monitoring of voltage 0)

   A2B_VMTR_INTEN= 0x01 (set bit 0 to enable VMTR interrupt for voltage 0)

   A2B_VMTR_VMIN6 = 0x3B (set minimum threshold to 3.7 V)

   A2B_VMTR_VMAX6 = 0xA1 (set maximum threshold to 10.1V)

4. Read the A2B_VMTR_VLTG0 register to obtain the measured voltage and the A2B_VMTR_MXSTAT[0] and A2B_VMTR_MNSTAT [0] bits for maximum and minimum voltages exceeded errors, respectively.

6 A^{2}B Event Management

The A^{2}B transceiver delivers digital, synchronous data and control information over distance to multiple nodes. The transceiver contains a state machine to manage the bit errors and A^{2}B bus link errors that occur during communication. The transceiver features a dedicated interrupt pin (IRQ) to indicate interrupts detected over the A^{2}B bus to the host processor. It includes status interrupts like main node PLL locked, sub node discovery done, bit errors, diagnostic interrupts, I/O interrupts,VMTR interrupt, SPI status/error, I^{2}C error, and mailbox interrupts. The A^{2}B error management system ensures that the robustness of the system is not affected by errors from signal interference or signal distortion.

The A^{2}B protocol engine contains a set of registers that provide support for interrupt processing. These registers include:

- A2B_INTMSK0 through A2B_INTMSK2: These registers contain mask bits for interrupts which needs to be reported on an IRQ pin.
- A2B_INTPND0 through A2B_INTPND2: These registers contain all the pending interrupt bits for the node. Interrupts that are unmasked using the A2B_INTMSK0 - A2B_INTMSK2 registers are only forwarded to report on the IRQ line of the main node.
- A2B_INTSTAT: This register indicates that there is an active interrupt pending in the A2B_INTTYPE register
- A2B_INTSRC : This register specifies which node has raised the interrupt that is pending in the A2B_INTTYPE register
- A2B_INTTYPE : This register specifies the interrupt type.
- A2B_LINTTYPE : This register is used in the sub node for indicating the I^{2}C mailbox or SPI interrupts to the local processor.

Main Node Interrupt Reporting

This section describes how main node interrupts are internally handled by the main transceiver. The ***Main Interrupt Reporting*** figure shows how A^{2}B main node raises the interrupt on the IRQ pin.

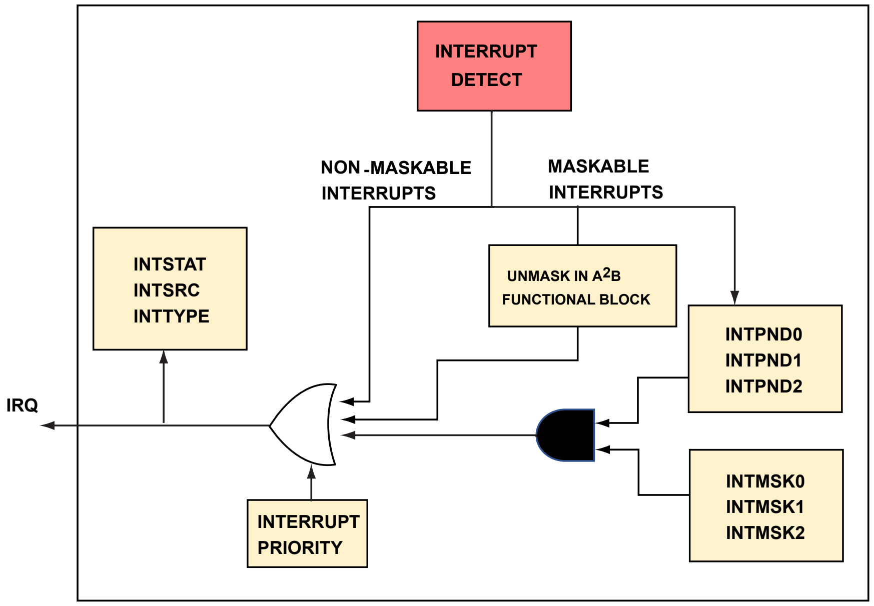


Figure 6-1: Main Interrupt Reporting

The maskable interrupts are raised to report on IRQ line based on the A2B_INTMSK0 - A2B_INTMSK2 registers settings. These interrupts include:

- Bit errors such as header counter error, CRC error, SRF CRC error, interrupt field CRC error, data decoding error, data parity error, bit error counter overflow
- Line fault interrupts like A^{2}B cable short-to-VBUS, short-to-GND, short together
- SRF Miss errors
- GP Input interrupts
- I^{2}C Error interrupt
- Sub node discovery interrupt
- Interrupts reported by subordinate nodes

Some interrupts can be enabled/unmasked by A^{2}B functional block to raise the interrupt on IRQ line of transceiver. These interrupts include:

- Mailbox full/empty interrupts
- SPI status and error interrupts
- VMTR interrupts
- GP Input interrupts also have enable bits at GPIO block level

Other interrupts are reported directly without need of any enable/unmask bit. These interrupts include:

- Main node PLL locked

- Interrupt messaging error (INTTYPE = 0x80)
- Sub node INTTYPE read error (INTTYPE = 0xFD)
- Standby done interrupt (INTTYPE = 0xFE)

The priority of interrupts depends on INTTYPE value – the lower the value, the higher the priority. For example, if CRC error, mailbox interrupt and I^{2}C error are pending at the same time, CRC error (INTTYPE = 0x02) is reported first, followed by the I^{2}C error (INTTYPE = 0x19) and finally the mailbox interrupt (INTTYPE = 0x30). In general, the following priority order applies:

- Lower number INTTYPE has priority over higher number.
- The A2B_INTPND0 register takes priority over the A2B_INTPND1 register, which takes priority over the A2B_INTPND2 register.
- Lower numbered bits in the pending registers (A2B_INTPND0 to A2B_INTPND2) take priority over higher numbered bits.

When masked interrupts (which are not enabled) occur, they are registered as sticky bits in the A2B_INTPND0 through A2B_INTPND2 registers, but, do not trigger interrupt requests. Some interrupts like mailbox interrupts update the status bits in their registers at the block level.

NOTE: Interrupts raised by the main node on the IRQ line does not mean interrupt from main node alone. The main node raises interrupts for the whole A^{2}B chain (from the main node and all the subordinate nodes). Therefore, the A2B_INTSRC register must be checked to identify the source of interrupt indicated in the A2B_INTTYPE register.

Subordinate Node Interrupt Reporting

The ***Subordinate Node Interrupt Detection*** figure shows how a A^{2}B subordinate transceiver raises an interrupt on the IRQ pin.

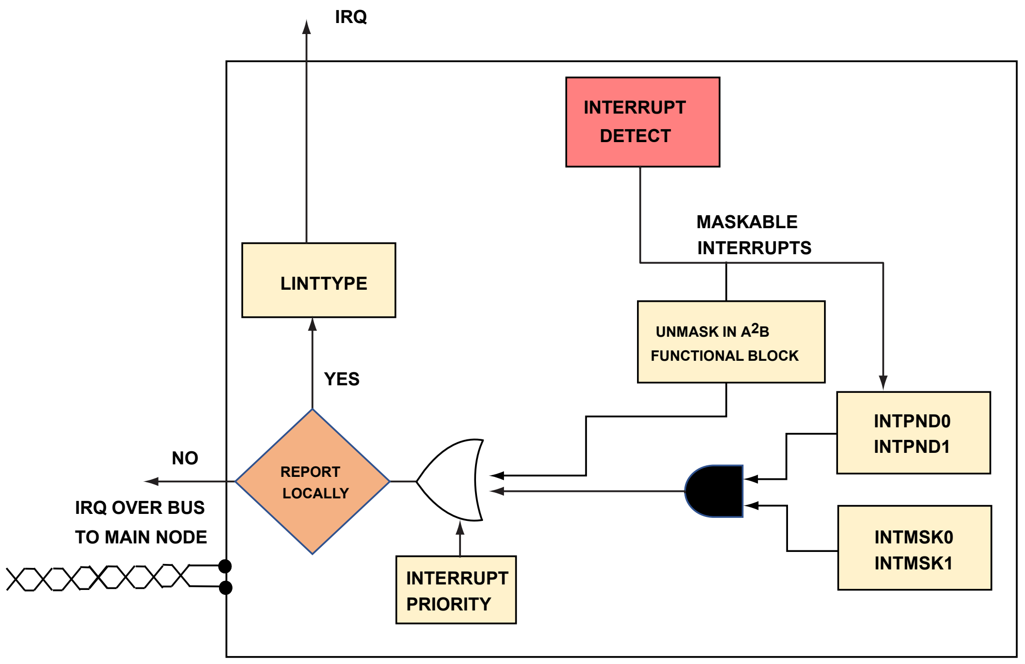


Figure 6-2: Subordinate Node Interrupt Detection

The interrupt reporting mechanism in subordinate node is same as main node, except most of the interrupts are conveyed to main node over A^{2}B bus instead of raising those on the local IRQ pin. Very few interrupts are raised on local IRQ pin during the operations that involves local DSP on board. Therefore, host processor on the main transceiver can manage the entire A^{2}B system with no or minimal intelligence/stack running at subordinate nodes.

The following interrupts are conveyed to A^{2}B main node transceiver over the A^{2}B bus:

The maskable interrupts as configured in the INTMASK0/1 registers setting. These interrupts include

- Bit errors such as header counter error, CRC error, SRF CRC error, interrupt field CRC error, data decoding error, data parity error, and bit error counter overflow
- Line fault interrupts such as A^{2}B cable short-to-VBUS, short-to-GND, and short together
- SRF miss errors
- GP input interrupts

Some peripheral interrupts can be enabled/unmasked by the A^{2}B functional block. These interrupts include:

- Mailbox full/empty interrupts
- SPI status and error interrupts
- VMTR interrupts
- GP input interrupts also have enable bits at GPIO block level

The subordinate nodes report the active interrupt to the main node over A^{2}B bus using a 6-bit IRQ field in the SRF. This field is protected by 4-bit CRC field. The *SRF Interrupt Response* figure shows the fields used in the interrupt response.

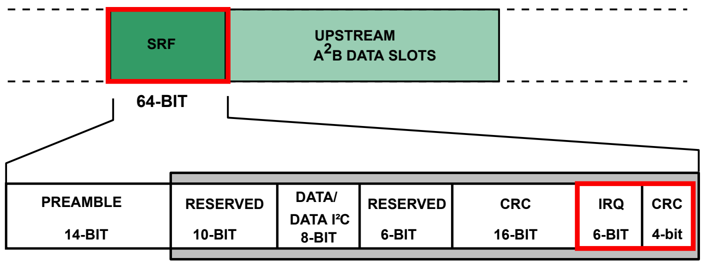


Figure 6-3: SRF Interrupt Response

The following interrupts are raised on the local IRQ pin depending on operations of those blocks:

- Mailbox full/empty interrupts
- SPI status and error interrupts

NOTE: Refer to the SPI, VMTR and mailbox sections for more information about when SPI/VMTR/MBOX interrupts are triggered.

A^{2}B Bus System Interrupt Reporting

The ***Interrupt Flow*** figure shows the interrupt flow in the A^{2}B system.

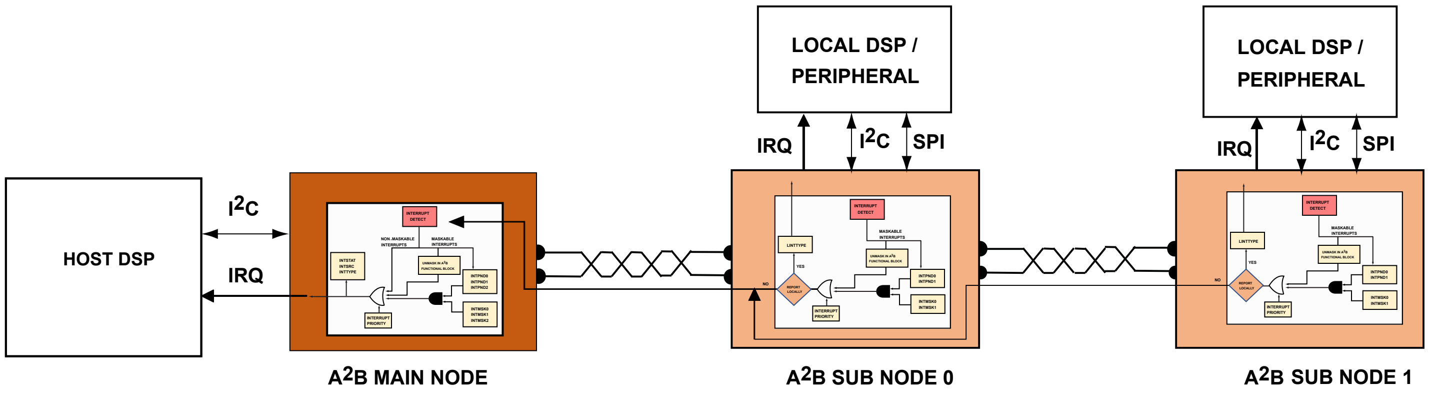


Figure 6-4: Interrupt Flow

The priority of interrupts among different nodes depends on node position – the closer the node is to the main node, the higher the priority. In general, the following priority order is applicable:

- Main node interrupts have priority over subordinate node interrupts
- Lower subordinate node IDs take priority over higher node IDs

For example, if different interrupts are reported by the main node, subordinate node 0 and subordinate node 1, then the interrupt raised by the main node is reported first on IRQ line, followed by the interrupt raised by subordinate node 0, and finally, the interrupt raised by subordinate node 1.

Subordinate Interrupt Handling

When an interrupt for a subordinate transceiver is triggered, the following sequence of events occurs:

1. The related bits (such as mailbox) in the interrupt registers (`A2B_INTPND0`/ `A2B_INTPND1`) or status registers in the A^{2}B functional blocks are updated in the subordinate transceiver.
2. If no interrupt is pending, the interrupt type is updated in the local `A2B_INTTYPE` register.
3. The subordinate transceiver begins signaling the IRQ in the interrupt field of the SRF. Any upstream subordinate transceivers without an active interrupt pass this field upstream.
4. When the main transceiver receives an interrupt notification through the IRQ field in the SRF and if no interrupt pending at that time, the main transceiver updates the `A2B_INTSTAT`, `A2B_INTSRC` and `A2B_INTTYPE` registers. It temporarily sets the `A2B_INTTYPE` register to 0x80. The IRQ pin of the main transceiver is driven active.
5. The main transceiver reads the `A2B_INTTYPE` register from the appropriate subordinate transceiver and updates its `A2B_INTTYPE` register.
6. Once the `A2B_INTTYPE` register is read, the main transceiver internally performs a write to the appropriate subordinate transceiver to clear the interrupt. The subordinate transceiver stops signaling the interrupt in the SRF.
7. When the IRQ pin of the main transceiver is asserted, the host processor reads the `A2B_INTSRC` and `A2B_INTTYPE` registers to determine the interrupt type and identify which subordinate node transceiver raised the interrupt.

NOTE: The internal INTTYPE read or clear process in step 5 and 6 can be held off when there is a new structure being applied (`A2B_CONTROL.NEWSTRCT` set within the last five superframes), or when a remote I^{2}C stop command needs to be sent.

If the host reads the `A2B_INTTYPE` register from the main transceiver after step 4, but before step 5 completes, the host may read 0x80 from the `A2B_INTTYPE` register. If the subordinate node does not drop off the bus, the `A2B_INTTYPE` field eventually updates. When the host reads INTTYPE= 0x80, an additional read of the `A2B_INTTYPE` register is recommended to confirm the interrupt type.

If a subordinate transceiver signals an interrupt and then drops off the bus (presumably, due to a switch fault), the next-in-line upstream subordinate transceiver eventually switches to being the last-in-line subordinate transceiver after 32 frames of missed SRFs. At this point, if the main transceiver (not the host processor) is still internally attempting to read the `A2B_INTTYPE` register from the missing subordinate transceiver, the newly last-in-line subordinate transceiver sends a special SRF. This SRF indicates to the main transceiver that the read cannot complete.

This event causes the A2B_INTTYPE register to be set to 0xFD from a temporary value of 0x80 (the interrupt identification process to terminate).

The subordinate node INTTYPE read error (0xFD) interrupt occurs when the main transceiver is attempting to read the INTTYPE from a subordinate transceiver based on a received interrupt, but receives a response from an upstream subordinate node indicating that it is now the last subordinate node. The main difference between INTTYPE = 0xFD and INTTYPE =0x80 is that INTTYPE = 0x80 can be seen while the main transceiver is still attempting to read INTTYPE. Therefore, it may subsequently resolve, whereas INTTYPE = 0xFD cannot resolve.

NOTE: If a subordinate node reports an interrupt to the main node without any additional line failures, and, the host reads the A2B_INTTYPE register too fast (after step 4), the main transceiver reads INTTYPE = 0x80, resulting in clearing the IRQ. The main transceiver does not reassert the IRQ when the A2B_INTTYPE register is read before the register value is updated from a subordinate transceiver. Therefore, it is recommended to read the A2B_INTTYPE register only when an active interrupt is confirmed (such as inside the IRQ handler or after reading the A2B_INTSTAT and A2B_INTSRC registers).

The subordinate node interrupt handling takes time. The time depends on whether other high priority interrupts are pending in the A^{2}B chain. When another interrupt appears during this time (for example, GPIO pin toggles), it may not be noticed. There is no interrupt FIFO inside the transceiver. The errors are latched in the A2B_INTPND0/ A2B_INTPND1 register of the node irrespective of whether they are enabled in A2B_INTMSK0/A2B_INTMSK1 register. The `INTMSKn` setting instructs which interrupts are forwarded into INTTYPE and raised on the IRQ line. When multiple interrupts are pending inside the `INTPNDx` register, the highest priority interrupt is forwarded to INTTYPE/main node. There is no FIFO or buffer inside the `INTPNDx` registers.

- When the active error interrupt is not yet serviced (because the host has not yet read the A2B_INTTYPE register or another high priority interrupt is being processed) and another error of the same type is raised, this error is not flagged to host twice.
- When the active error interrupt is being serviced (the host is reading the A2B_INTTYPE register) and towards the end of the operation (for example, after clearing the bit in the `INTPNDx` register and the IRQ lines are deasserted),and another error of the same `A2B_INTTYPE.TYPE` is raised, the IRQ line toggles back for a new interrupt (if it has a higher priority). Otherwise, the error is latched again inside the `INTPNDx` register. If the INTTYPE is not yet cleared, the corresponding bit in the `INTPNDx` register remains set. If a new error of same `A2B_INTTYPE.TYPE` comes now, the bit remains set. Each interrupt has 1-bit pending bit. There is no counter or FIFO. When the host handles the interrupt by reading the A2B_INTTYPE register, the interrupt is cleared, and the bit in the `INTPNDx` register is also cleared (even though multiple errors were raised during this time). For example, for a subordinate node IO interrupt, the `A2B_INTPND2.SLVIRQ` bit of the main node is set. When no other high priority interrupts are active, this interrupt is forwarded into the A2B_INTTYPE register and the IRQ signal is raised.

  Main node interrupts have the highest priority. Among subordinate nodes, a subordinate node closer to the main node has a higher priority than the last-in-line subordinate node. Therefore, if a subordinate node receives multiple active input edges, and the host does not service them quickly enough, some interrupts can be missed. In a typical use case, the IO toggle rate and interrupt generation rate may not be fast. A typical use case for IO interrupts includes interrupts from a codec/ADC in clip events or some handshake by a local subordinate processor. Therefore, the bit errors must be counted in the bit error counter (instead of reporting individual errors) and appropriate action (based on error density) must be taken.

Host Interrupt Handling

When an enabled interrupt triggers, the following interrupt registers are updated to reflect the interrupt information:

- A2B_INTSTAT - indicates that an interrupt is active
- A2B_INTSRC - indicates which node transceiver raised the interrupt
- A2B_INTTYPE - indicates the type of active interrupt
- A2B_INTPND0- A2B_INTPND2 - for some interrupts, a bit in these registers is also set irrespective of whether it is enabled to raise an interrupt on the IRQ pin using the A2B_INTMSK0 -A2B_INTMSK2 register. The IRQ signal is raised to an active level to indicate to the host processor that an interrupt is pending.

When there is an interrupt request from the A^2B main transceiver, the host can read the A2B_INTSRC and A2B_INTTYPE registers to obtain the subordinate node ID that generated the interrupt request and the type of interrupt request, respectively. At the completion of the A2B_INTTYPE register read, the active interrupt is cleared and the IRQ signal clears if there are no further pending interrupts. If there are any pending interrupt requests, the IRQ pin goes low for one $f_{SYSBCLK}$ cycle (~20 ns) when the A2B_INTTYPE register is read and immediately transitions to high.

By default, interrupt requests are indicated with a high level on the IRQ pin and the setting of the A2B_INTSTAT.IRQ bit. The active polarity of the IRQ can be changed using A2B_PINCFG.IRQINV bit.

NOTE: When the host processor services the interrupt by reading the A2B_INTTYPE register, the A2B_INTSTAT and A2B_INTSRC registers and the corresponding sticky bit (A2B_INTPND0- A2B_INTPND2) registers are automatically cleared along with the IRQ pin deassertion. But, the A2B_INTTYPE register is not cleared. INTTYPE = 0 is a valid interrupt type (HDCNTER). Therefore, it is important to note that if A2B_INTSTAT = 0, the value of theA2B_INTTYPE register is don't care, as it does not indicate an active interrupt. When another interrupt becomes active, the contents of the A2B_INTSTAT , A2B_INTSRC and A2B_INTTYPE registers are updated.

NOTE: The A2B_INTTYPE register should only be read when the interrupt is active (A2B_INTSTAT = 1 or A2B_INTSRC != 0).

Error Management

Communication on the A^2B bus occurs in periodic superframes with a frequency of 48 kHz or 44.1 kHz. This frequency is the same as the audio frequency used in the system. Each superframe on the A^2B bus contains one Synchronization Control Frame (SCF) at the beginning, followed by the downstream traffic and one Synchronization

Response Frame (SRF), which is then followed by the upstream traffic. Both the SRF and SCF contain a Cyclic Redundancy Check (CRC) field, which is used for error detection once the frames are received. Each synchronous data slot contains a parity bit that is checked at the receiving node. Additionally, all line code violations are monitored during data decoding to maximize the bit error detection capability of the A^2B transceiver. The ***Frame Structure Details*** figure shows the structure of an A^2B bus superframe.

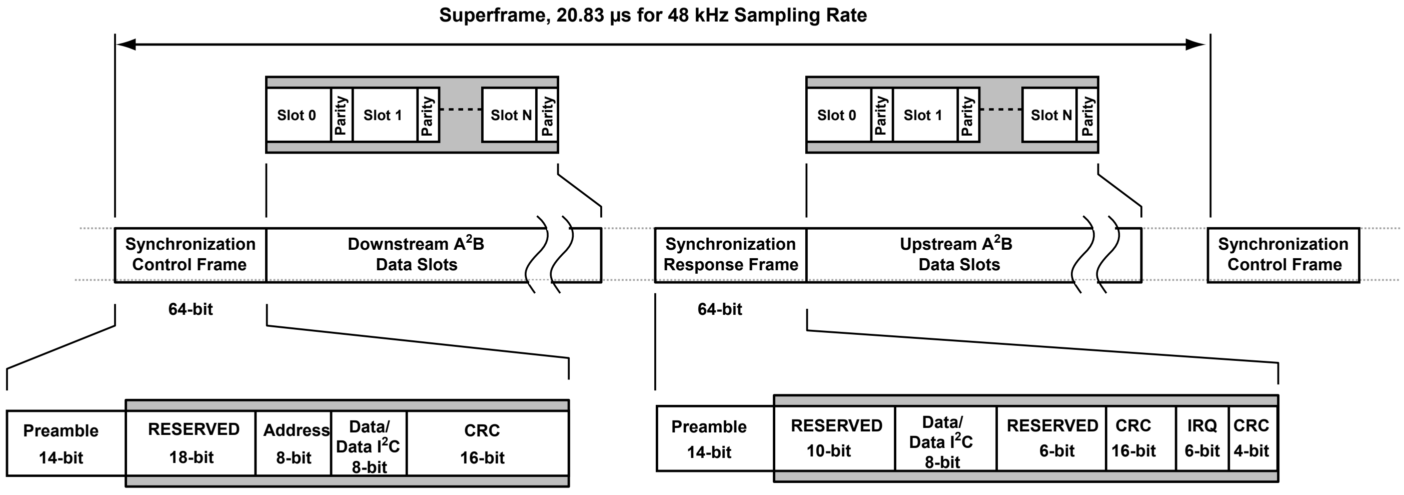


Figure 6-5: Frame Structure Details

The errors that arise during data communication can be classified into three categories: data errors, control and response errors, and bus link communication errors.

- Data Errors:
  - Data Parity Error (DPERR): INTTYPE = 3
  - Data Decoding Error (DDERR): INTTYPE = 1
- Control and Response Errors:
  - CRC Error (CRCERR): INTTYPE = 2
  - SRF CRC Error (SRFCRCERR): INTTYPE = 6
  - Interrupt Frame CRC Error (ICRCERR): INTTYPE = 26
  - Header Count Error (HDCNTERR): INTTYPE = 0
- Bus Link Communication Errors:
  - SRF Missed Error (SRFMISSERR): INTTYPE = 5
  - I^2C Error (I2CERR): INTTYPE = 25
  - Interrupt Messaging Error: INTTYPE = 0x80

- Sub node Interrupt Type (INTTYPE) Read Error: INTTYPE = 0xFD
- Main node PLL locked interrupt during run time: INTTYPE = 0xFF

All data and control and response errors except SRFCRCERR can be counted in the bit error counter using a configurable threshold. When the threshold is reached, the bit error counter overflow (BECOVF) interrupt (INTTYPE=4) is generated.

These errors are explained in the following sections.

Data Errors

The A^{2}B data communication errors include:

- Data Parity Error (DPERR)
- Data Decoding Error (DDERR)

Data Decoding Error (DDERR)

The DDERR error indicates a missing clock edge in the Differential Manchester data stream on the A^{2}B bus. The data decoding error is reported only on data slots that are being consumed by the particular node. A data decode error in an SCF/SRF results in a CRC error. The transceiver does not raise a data decoding error.

It is possible that the data decoding error is reported when a transceiver tries to sample a slot that is not present on the bus (for example, when there are no clock edges in a slot). This error can occur when a downstream transceiver node that was contributing to these slots is dropped. This error can also be reported when the SLOT registers are not configured correctly.

These errors are detected by both main node and subordinate nodes when a decoding error happens on the slots consumed by them.

Transceiver Action on Error

When a data decoding error is detected for a slot, that particular slot is replaced with last known good sample from a previous frame.

Guidelines for Host Processor Software

Consider the following host processor software guidelines:

1. It is not necessary to enable the reporting and handling of individual data decoding errors. The A^{2}B transceiver automatically takes action to discard the data and replace it with the last known good sample (`A2B_INTMSK0.DDEIEN` = 0).
2. This error can be enabled to count in the bit error counter (`A2B_BECCTL.ENDD` = 1). Main node and subordinate node transceivers have an individual bit error counter.

NOTE: When a downstream node that contributes to some upstream slots is dropped, the upstream nodes that are configured to consume the slots can continuously generate data errors (DPERR and/or DDERR) for the slots (for example, in each frame and for each data slot). So, the bit error counter threshold is reached quickly. This may not be desirable when a partial bus operation is required. In this case, use one of the following options:

- Reconfigure the slots when the node drops. This option can cause a change to the TDM data map.
- Disable counting the error when the node drops. This option disables the reporting of data decoding errors coming from active node transceivers for the dropped node. It is not possible to disable data parity reporting of a particular data slot.

Data Parity Error (DPERR)

Each data slot on the A^2B bus is protected by a parity bit. It is odd parity format. If the parity check fails for data slot contents, then DPERR is generated. The DPERR error is reported only on data slots that are consumed by the particular node.

If a transceiver node that contributes to some upstream slots is dropped, the node transceiver that consumes the slots raises data parity errors.

If an application enables ECC checksum instead of data parity (`A2B_SLOTFMT.UPFMT` / `A2B_SLOTFMT.DNFMT` =1 for 24-bit/32-bit audio), the ECC failure is captured as DPERR. With ECC for data slots, single bit error are corrected; no error is reported. If decode errors are seen on two or more bits of the data word, it is treated as bad. If the ECC logic reports an uncorrectable error, it is reported as DPERR.

In comparison, a 1-bit parity indicates whether the number of ones in the data word is high or low. So, the application can detect a single bit error only and not multi-bit errors. Both main node and subordinate nodes detect these errors when they detect a parity error on the consumed slots.

Transceiver Action on Error

When DPERR is detected for a slot, that particular slot is replaced with the last known good sample from a previous frame.

Guidelines for Host Processor Software

Consider the following host processor software guidelines:

1. It is not required to enable reporting and handling of individual data parity errors. The A^2B transceiver has automatically taken action to discard erroneous data and replace it with the last known good sample (`A2B_INTMSK0.DPEIEN` = 0).
2. Enable the counting of this error in the bit error counter (`A2B_BECCTL.ENDP` = 1). Main node and subordinate transceivers have individual bit error counters.

Control and Response Errors

The A^2B control and response errors include:

- CRC Error (CRCERR)

- SRF CRC Error (SRFCRCERR)
- Interrupt Frame CRC Error (ICRCERR)
- Header Count Error (HDCNTERR)

CRCERR

CRCERR indicates that a sub node detects a cyclic redundant code (CRC) error in the received SCF field. For the main node, the error indicates a CRC error in the received SRF field. The 16-bit CRC field protects the 64-bit SCF. Similarly, there is CRC field in the SRF (although the corresponding interrupt field is protected by a separate CRC). This CRC does not protect upstream or downstream data slots.

Transceiver Action on Error

When a sub node detects a SCF CRC error, the downstream slots consumed by node are replaced with the last known good samples from the previous frames. Data being passed downstream through the B-port are passed as is. If there is a command in the SCF and the sub node detects a CRC error, it discards the fields. The main node retries the access until it gets the acknowledgement or a -superframe timeout occurs.

When the main node detect a SRF CRC error, the upstream slots consumed by the node are replaced with the last known good samples from the previous frame. If there is a command response in the SRF and the main node detects the CRC error, it discards the fields. The main node retries to get a response in the next superframe.

Guidelines for Host Processor Software

It is not required to enable reporting and handling of individual CRC errors because the A^{2}B transceiver automatically takes the required action (`A2B_INTMSK0.CRCEIEN` = 0).

This error can be enabled to count in the bit error counter (`A2B_BECCTL.ENCRC` = 1).

Main and sub nodes have individual bit error counters.

SRF CRC Error (SRFCRCERR)

In the second half of the superframe, the last-in-line node generates the SRF field (protected by CRC) and passes it upstream to the main node through all the middle sub nodes. All the middle sub nodes check the CRC in the SRF field and report an SRFCRCERR error if it detects a CRC error in the SRF field. This error is only detected by the sub nodes. If the main node detects a CRC error in the SRF field, it is reported as CRCERR instead of SRFCRCERR.

Transceiver Action on Error

When a sub node detects SRFCRCERR, upstream slots consumed by node are replaced with the last known good samples from the previous frame. Data being passed upstream (from the B-port to the A-port) are passed as is. The sub node does not try to correct the SRF CRC error; it passes the SRF as is upstream. However, in the case of a response to a command or generating a GPIO-over-distance update, the sub node inserts its own SRF (including the CRC). The decision of whether to generate or pass the IRQ field is handled independently. In the main node, the CRCERR field is used to indicate a CRC error in the SRF.

Guidelines for Host Processor Software

It is not required to enable the reporting and handling of individual SRF CRC errors because the A^2B transceiver automatically takes the required action (`A2B_INTMSK0.SRFCRCEIEN` = 0). This error cannot be enabled to count in the bit error counter; the `A2B_BECCTL.ENCRC` bit enables the counting of CRCERR, but not SRFCRCERR on the sub nodes. In the main node, this error is reported as CRCERR, which can be counted in the bit error counter.

Interrupt CRC Error (ICRCERR)

The SRF field consists a 6-bit IRQ field that sub nodes use to report detected interrupts to the main node. This field is protected by a 4-bit CRC. If the CRC fails, the interrupt field CRC error (ICRCERR) is reported. These IRQ (6-bit) + CRC (4-bit) fields are checked only by the main node. The main node can generate ICRCERR when there is an error. Sub nodes do not generate the ICRCERR error.

Transceiver Action on Error

When a main node detects ICRCERR, it discards the IRQ field. The sub node continues to report the interrupt until the main node acknowledges the reported interrupt, therefore there is no incorrect reporting of interrupts or interrupt misses.

Guidelines for Host Processor Software

It is not required to enable the reporting and handling of individual ICRCERR errors because the A^2B transceiver automatically takes the required action (`A2B_INTMSK2.ICRCEIEN` = 0). This error can be counted in the bit error counter (`A2B_BECCTL.ENICRC` = 1), but it is typically not necessary to count.

Header Count Error (HDCNTERR)

The SCF and SRF fields contain a 2-bit field CNT. In the SCF, the CNT field is incremented (modulo 4) from the value used in the previous superframe. In the SRF, the received value of the CNT field in the SCF is transmitted back to the main node. HDCNTERR indicates that the current node has received a different header count than the expected header count. The main node raises this error when the received SRF has a different CNT value than expected. The sub nodes raise this error when the received SCF has a different CNT value than expected.

Transceiver Action on Error

HDCNTERR mostly results in a CRC error because the header count field is also protected by CRC in the SCF/SRF. If the CRC check fails, the CRC error is also raised with HDCNTERR and appropriate action is taken for the CRC error. If the CRC passes, only HDCNTERR is raised; no other action is taken with respect to the data slots.

Guidelines for Host Processor Software

1. It is not required to enable the reporting and handling of individual header counter errors (`A2B_INTMSK0.HCEIEN` = 0).

2. Since HDCNTERR mostly results in CRCERR, it is not recommended to enable counting this error in the bit error counter. Otherwise, there could be double counting of bit errors (`A2B_BECCTL.ENHDCNT` = 0).

Bit Error Counter Overflow (BECOVF)

Each node has a bit error counter which can count bit errors like CRCERR, DDERR, DPERR, HDCNTERR, and ICRCERR. The `A2B_BECCTL.THRESHLD` field configures the number of bit errors to be counted before the bit error counter overflow (BECOVF) interrupt is raised. Excessive bit errors set the `A2B_INTPND0.BECOVF` bit and signal the interrupt. Refer to bit error counter section for details.

Guidelines for Host Processor Software

The bit error counter threshold is useful when reporting each bit error to the host processor becomes undesirable. The threshold can be set based on acceptable noise and robustness over a period. The bit error counter should be cleared periodically so that the counter does not accumulate over time. The bit error counter clearing time depends on the acceptable level of bit error density for a system use case. It can differ for audio or microphone applications. Bit error thresholds can be independently configured at the main and sub nodes. It may be possible to implement a software counter in the host software on top of the bit error counter interrupt. The software can take the necessary action if the bit error counter overflows frequently.

Automatic Correction

Considering A^{2}B transceiver response for data errors and control and response errors, the possible cases for automatic correction of received data slots in a node include:

- If the frame sync preamble is not seen, all data slots received from the bus are automatically replaced with previous, good values.
- If a sub node detects a CRC error in the SCF, all downstream data slots received from the bus are replaced with previous good values.
- If a main node detects a CRC error in the SRF, all upstream data slots received from the bus are replaced with previous good values.
- If a data decoding error or a data parity error is detected within a data slot, the received erroneous data slot is replaced automatically with a previous good slot value.

If the error is caused in the control and response fields, the access is retried internally. When a host accesses the sub transceiver registers (for example, through I^{2}C over distance), the SCF and the SRF carry the data exchange. If there is a communication error in the SCF or SRF, the main node automatically initiates a retry of the register access. The main node retries multiple times until either the access is successful or an I^{2}C timeout occurs in the main node. During the retry time, I^{2}C clock stretching is applied, which signals to the host that the transaction is not complete. If there is an I^{2}C timeout (the I^{2}C timeout occurs after 30 superframes), the main flags an I2CERR interrupt, to which the host can respond.

Erroneous interrupt requests received in the main node are ignored. If a real interrupt event occurs, an interrupt is automatically regenerated by the sub node (since it is not cleared).

Bus Link Communication Errors

The A^2B communication and bit errors are:

- SRF Missed Error (SRFMISSERR)
- I^2C Error (I2CERR)
- Interrupt Messaging Error
- Sub node Interrupt Type (INTTYPE) Read Error
- Main node PLL Locked Interrupt (MSTR_RUNNING)

SRFMISSERR

The SRFMISSERR error indicates that the SRF of a next-in-line node is not received before the expiration of the local timing window. The affected node generates its own SRF which is being sent upstream to any earlier nodes. The error is valid for both main and sub nodes.

Each node expects the SRF from the downstream node at the timing specified at response cycle (the timing can be auto-adjusted during the node discovery process). If the node does not detect the SRF in a superframe, it reports an SRFMISS error.

When a node drops (the transceiver stops providing SRFs to an upstream node), the upstream node reports SRFMISSERR in consecutive 32 superframes. After the 32 superframes, the upstream node becomes the last-in-line node (`A2B_NODE.LAST`= 1) and stops reporting SRFMISSERR.

Transceiver Action on Error

When the node does not detect the SRF from the downstream node at the specified time, it generates its own SRF and passes it upstream. If the node does not detect the SRF consecutively for 32 superframes, it assumes a downstream node has dropped. It becomes the last-in-line node (`A2B_NODE.LAST` = 1) and stops reporting SRFMISS.

Guidelines for Host Processor Software

When SRFMISSERR is received by the host, it should read the `A2B_NODE` register (0x29) of the node that reported the error (known from the `A2B_INTSRC` register). If SRFMISSERR is raised because a downstream node dropped, the `A2B_NODE` register (of node that raised the error) indicates `A2B_NODE.LAST`=1. If the host confirms the condition, it can take further action such as a node drop response or a partial rediscovery of the dropped node. If `A2B_NODE.LAST`=0, the reported SRFMISSERR could be transient and can be ignored. Keep count of this type of error and take action if the error is getting reported, frequently.

I2CERR

The I^2C error (I2CERR) happens when the host tries to access sub transceiver that do not exist (incorrect `A2B_NODEADR` register value) or the node has dropped from the bus. Similarly, I2CERR can happen when a remote I^2C access fails as a result of:

- NACK

- a 30 superframe timeout while waiting for ACK/access completion
- a busy sub I^2C bus

The main node reports I2CERR when the host processor attempts the I^2C access over distance. Local register accesses never result in I2CERR.

Transceiver Action on Error

If the I^2C access results in error, the main node aborts the access and raisesI2CERR. The A^2B main node does not wait for more than 32 superframes when the access is held or stretched (for example, the peripheral the not ready). Therefore, the I^2C lockup condition is avoided.

Guidelines for Host Processor Software

If there is an I^2C error, retry the access (2-4 times). If retry accesses fail every time, check for a dropped node by reading known registers from the sub node. Some host processor applications rely on the error generated by the I^2C driver. If an application does not rely on the I^2C interrupt from the A^2B transceiver, disable the I^2C error (`A2B_INTMSK2.I2CEIEN` = 0). In this case, the `A2B_INTPND2.I2CERR` bit can be checked after the I^2C access over bus. This verification is important for burst type accesses where the host processor provides the data acknowledgement signal to access more bytes.

Interrupt Messaging Error

When the main node starts handling a sub node interrupt, the `A2B_INTTYPE` register is temporarily set to 0x80. This value is overwritten when the main node reads the `A2B_INTTYPE` register from the sub node. When the interrupt messaging error (INTTYPE = 0x80) is raised, it can indicate a line fault or a dropped node. See Subordinate Interrupt Handling for details. This error is reported by the main node only.

Guidelines for Host Processor Software

When this error is received, the host processor can check for a dropped node by reading a known register from the last sub node.

Subordinate Node Interrupt Type (INTTYPE) Read Error

The sub ode INTTYPE read error interrupt occurs when the main node attempts to read the INTTYPE from a sub node based on a received interrupt and it receives a response from an upstream sub node indicating that it is now the last-in-line sub node. This interrupt can indicate a line fault or dropped node. See Subordinate Interrupt Handling for details. This error is reported by the main node only.

Guidelines for Host Processor Software

When this error is received, the host processor can check for or dropped node by reading a known register from the last sub node.

Main Node PLL Locked Interrupt (MSTR_RUNNING)

After device power-up, the main node transceiver locks its PLL and raises the main node PLL locked interrupt MSTR_RUNNING (INTTYPE = 0xFF).

If the MSTR_RUNNING interrupt is reported during run time, it indicates that the main node PLL was unlocked and then relocked. The main transceiver went through a reset state, thereby dropping of all sub nodes as well. The cause for main node PLL unlock during run time is due to a SYNC signal issue (jittery or breaks/changes). Refer to Main Node Bring-Up for details.

The following sequence can cause this interrupt to raise again:

1. The A^2B chain is active with communication between the nodes ongoing.
2. The main node PLL unlocks.
3. The main node transitions from the PLL LOCKED state to the POWER-UP state. All registers return to reset, except the `A2B_CONTROL.MSTR` registers.
4. If `A2B_CONTROL.MSTR` =1 and the SYNC signal is available, the main node PLL relocks.
5. The main node transitions to the PLL LOCKED state again and raises MSTR_RUNNING (INTTYPE=0xFF).

Guidelines for Host Processor Software

When this error is received, the host processor should rediscover the bus and possibly any peripherals connected to sub nodes.

Bit Error Counter

To avoid host intervention for every bit error, bit error control can be used. The `A2B_BECCTL` register selects which communication errors are counted. A counter threshold must be exceeded for an interrupt request to be generated. The register controls what bit errors are counted and configures interrupt thresholds of 2^n, where n ranges from 1 to 8. The threshold can be set based on acceptable noise and robustness over a period. Using this feature, some single-bit communication errors do not report an interrupt on the IRQ line unless they significantly accumulate over a time period (after the `A2B_BECNT` register was last cleared).

To count bit errors in the counter register, it is not necessary to enable the corresponding interrupts in `INTMASK` registers. In this case, the interrupt is raised for each bit error detected. This situation is not recommended for processor software handling.

The *Bit Error Counter* figure shows how the `A2B_INTSTAT.IRQ` bit works when bit error counting is enabled.

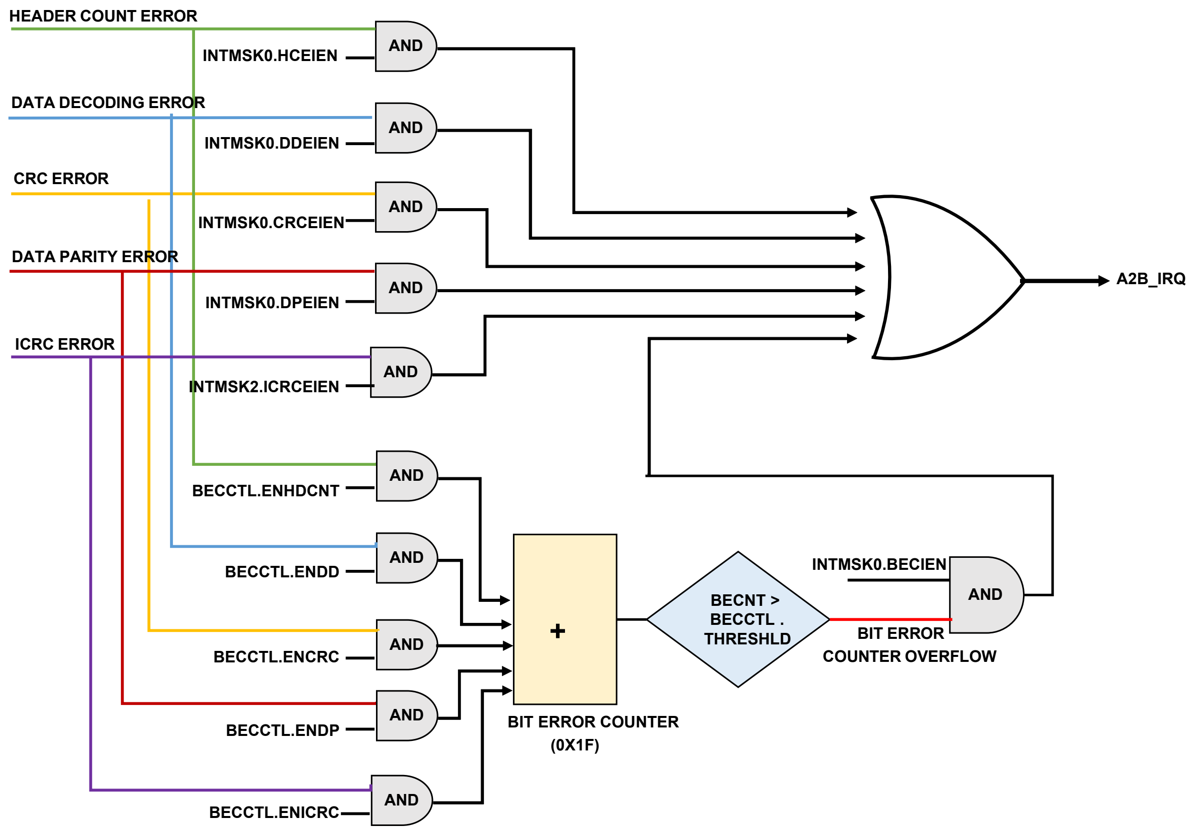


Figure 6-6: Bit Error Counter

NOTE: Enabling bit error counting in the `A2B_BECCTL` register automatically increments the counter whenever corresponding bit errors are raised. It is not necessary to enable the `INTMSK` bits of the corresponding interrupts.

Avoid double counting errors. For example, if there is a data decode error, there will likely also be a parity error. If the count is incremented for both errors, it results in a double count that artificially inflates the bit error rate. Similarly, if there is a header count error, there will also be a CRC error. Enabling all audio bit error fields in the `A2B_BECCTL` register results in double counting. To avoid double counting of bit errors, use the following settings in the `A2B_BECCTL` register:

1. Enable only the DDERR count (`A2B_BECCTL.ENDD` = 1) for an approximate bit error count (this method catches most bit errors). The true bit error count can only be achieved with PRBS testing.
2. Enable only the CRCERR count (`A2B_BECCTL.ENCRC` = 1) to count SCFCRC and SRFCRC errors. It gives a good indication of erroneous superframes during steady interference. This configuration works for errors due to Bulk Current Injection (BCI)), but would double the counts for erroneous superframes during severe interference.
3. Enable only the DPERR count (`A2B_BECCTL.ENDP` = 1) for an estimated slot error count.

Error Management Register

When A^2B data slots are not received correctly (detected by a parity error or a data decode error on any bit in the slot), the last good sample received for that slot is repeated. The A2B_ERRMGMT register also controls the ways in which bad data slots can be indicated across the I^2S/TDM interface.

When the A2B_ERRMGMT.ERRLSB bit is set, the LSB of each data slot is used to indicate whether the slot is received correctly or not. For example, in the main node with a 24-bit upstream slot size, the 24th data bit sent over DTX0 or DTX1 is low when the data is valid; it is high when the data is not valid. This method changes the meaning of the LSB in the received I^2S/TDM data words.

When the A2B_ERRMGMT.ERRSIG bit is set, all bits below the LSB of each data slot are used to indicate whether the slot is received correctly. With a 24-bit slot size, the last 8 bits in each 32-cycle data slot are low when the data is valid; the bits are high when the data is not valid. If the A2B_ERRMGMT.ERRSIG bit is not set, the extra eight bits are always low. This method preserves the meaning of the LSB in the received I^2S/TDM data words, but the data word size must be smaller than the data channel size for this method to work.

When the A2B_ERRMGMT.ERRSLOT bit is set, the number of slots generated on the A^2B bus is incremented by 1. In the main node, the protocol engine normally writes A2B_UPSLOTS pieces of data to the frame buffer in each superframe. In a subordinate node, the number of slots written is normally A2B_LDNSLOTS + A2B_BCDNSLOTS. The additional data slot enabled by using this method is appended to the end of the configured A^2B traffic. It contains a single bit of error information for each of the preceding data slots in that superframe. The MSB of the extra slot indicates that an error occurred in data slot 0. The next bit indicates an error in data slot 1, and so on. For example, 0x80000000 indicates that there was an error in slot 0, while 0xffffff00 indicates that slots 0 through 23 all contained errors.

Sub Node GPIO Interrupt Latency

Interrupts are signaled upstream from a sub transceiver to the main transceiver within the Synchronization Response Frame (SRF). Interrupts that engage after the beginning of the SRF (after the sub node starts driving the AP and AN pins) are signaled to the main node in the SRF of the next superframe. Assuming there are no other interrupts with a higher priority that mask the IO pin interrupt in question, the latency between a sub node IO pin and main node IRQ is the sum of:

- Four SYSBCLK cycles for pin interrupt generation (81.4 ns) +
- One superframe latency to get into SRF (20,833.3 μs) +
- 64 SYSBCLK cycles for the length of the SRF (1,302.1 ns) +
- Five SYSBCLK cycles for main RX latency (101.7 ns) +
- Two SYSBCLK cycles for IRQ logic in the main node (40.7 ns)
  - SYSBLK = system bus clock frequency = main node SYNC frequency * 1024

In addition to this total latency of 22.36 μs, there is an additional nine SYSBCLK cycles of latency for each sub node that the SRF must pass through (N × 183.1 ns). For example, in a system with three sub nodes, a GPIO interrupt from sub node 2 to the main node has a maximum latency of 22.36 μs + (2 x 0.183) μs= 22.73 μs.

7 Discovery Flow

This section provides information about node discovery and initialization for an A^{2}B bus system. There are two discovery flows: one for the discovery and initialization of the single pair system and the other for a XLR/DMX or RJ45 CAT cable based system as shown below.

Discovery Flow

All subordinate nodes are discovered sequentially from subordinate node 0 to the last-in-line subordinate node in the system with the software flow shown in the Discovery Flow figure. The following figures shows the stages show commands as issued over the I2C interface between the host and the main-enabled transceiver in different power scheme. Write commands are identified as "wr" and read commands are identified as "rd" along with the REGISTER_NAME being accessed. The"M" indicates an access to the BASE_ADDR, and the "S" indicates an access to the BUS_ADDR.

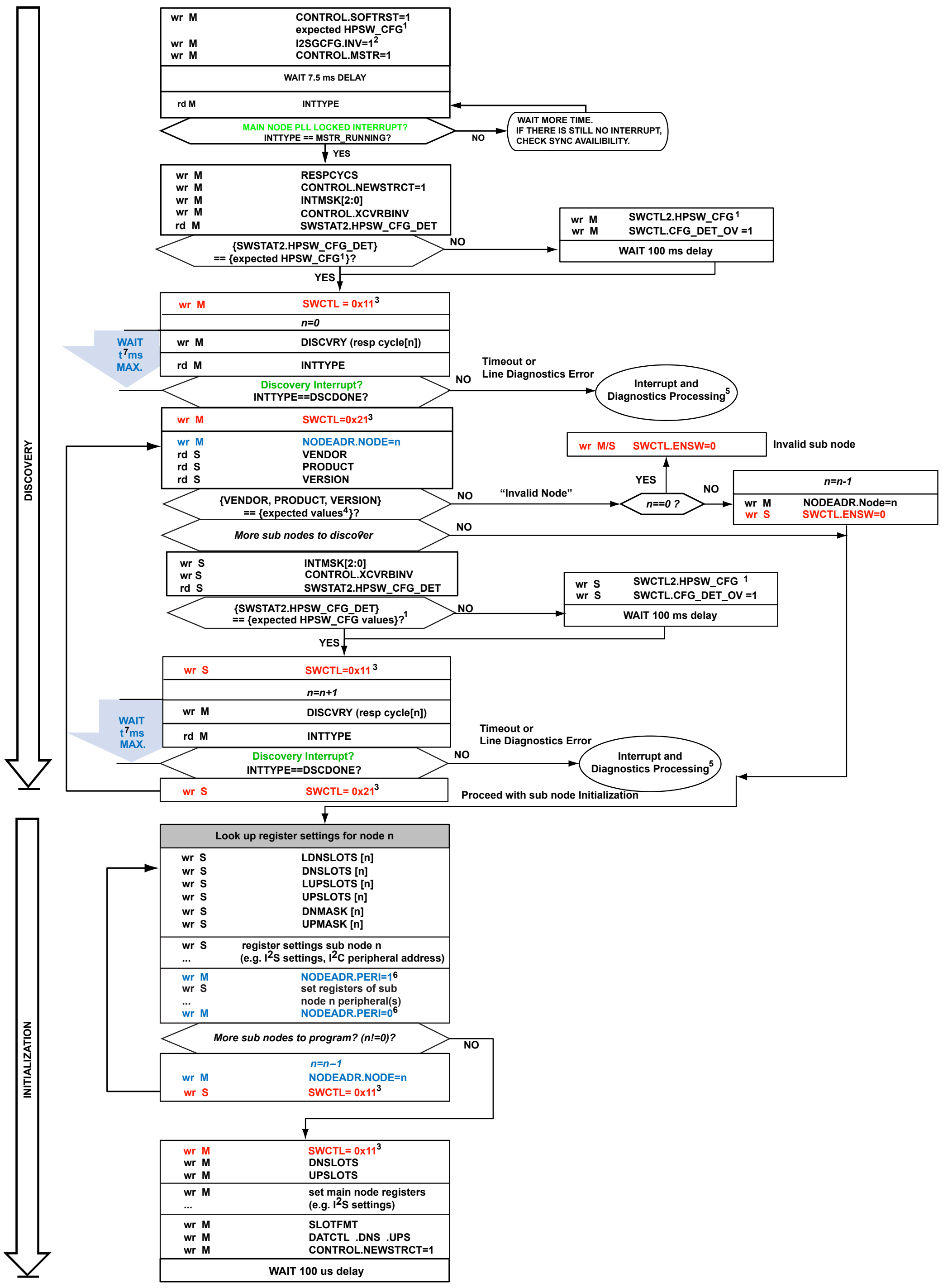


Figure 7-1: Discovery Flow in Single Wire Pair System

The following two figures show the discovery flow for XLR/DMX cable systems.

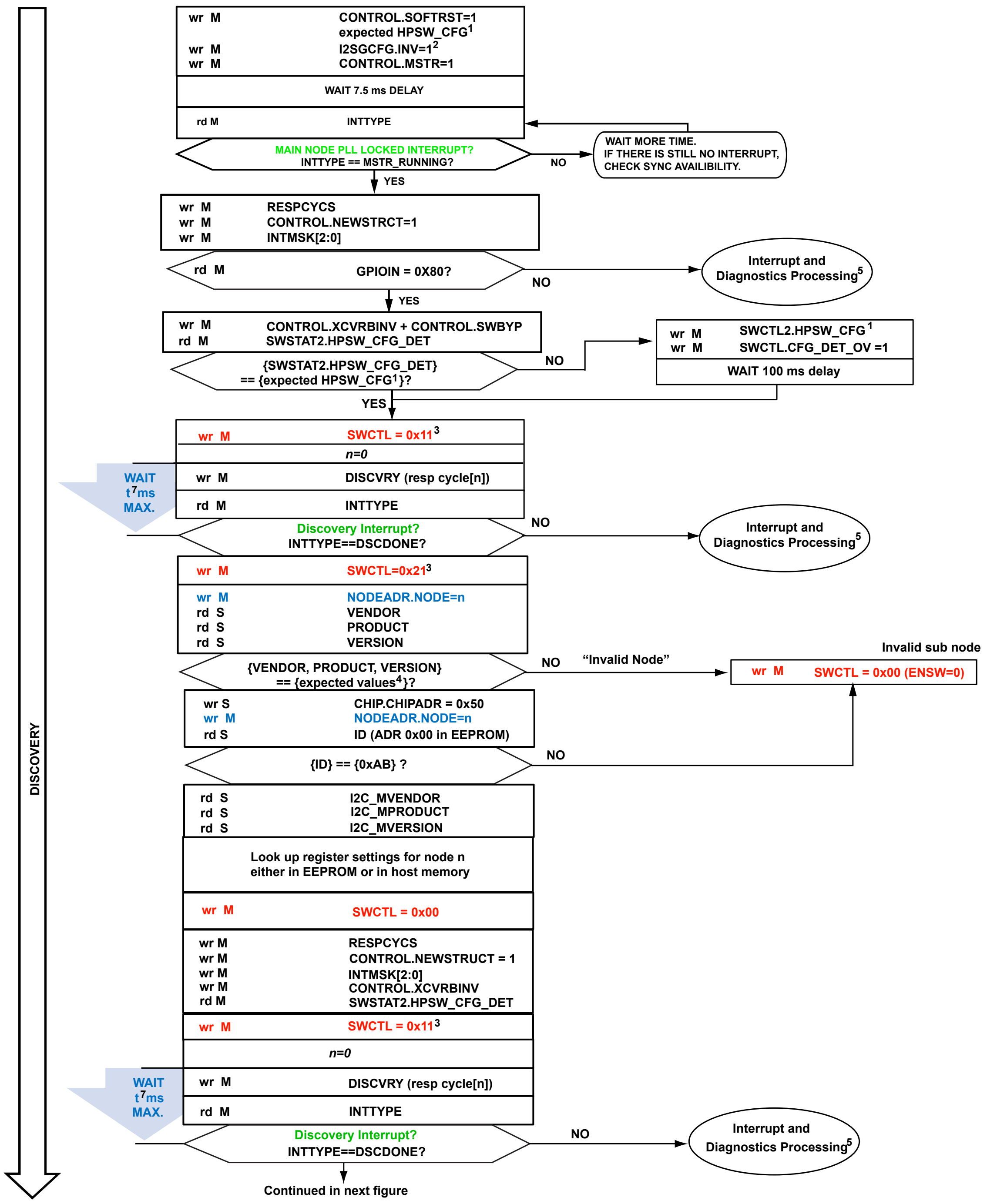


Figure 7-2: Discovery Flow in XLR/DMX and RJ45 CAT Cable Based System (part 1)

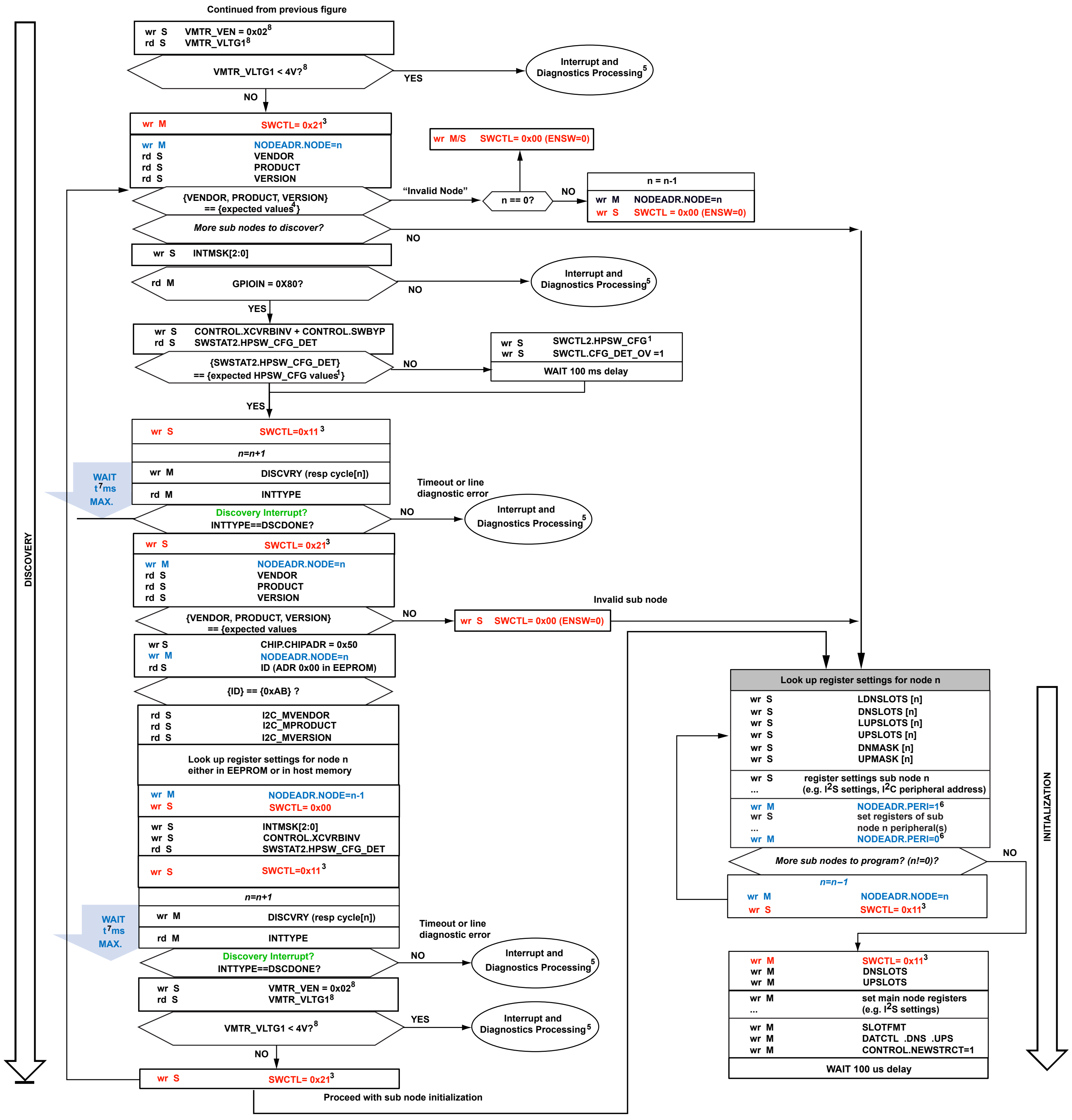


Figure 7-3: Discovery Flow in XLR/DMX and RJ45 CAT Cable Based System (part 2)

Note the following information referenced in the Discovery Flow figure.

1. The expected power configuration is:

Mode	SWSTAT2	SWCTL2
LVI Mode	0xA0	0x04
Non LVI Mode	0x20	0x04

2. Step not needed if host doesn't support inverted SYNC
3. Set the A2B_SWCTL.CFG_DET_OV bit if overriding detected A2B_SWCTL2.HPSW_CFG
4. From the host
5. Refer to A^{2}B System Debug chapter for details
6. It is not necessary to set the A2B_NODEADR.PERI bit when discovering through the SPI port
7. t = 70 ms
8. This step applies to RJ45/CAT cable-based systems only.

NOTE: Setting the A2B_SWCTL.ENSW bit in the main node or in any subordinate node causes it to begin sending SCFs downstream to the next connected subordinate node, thus allowing the next-in-line subordinate transceiver to begin locking its PLL before the main node initiates the discovery frames targeting it.

Use the following guidelines for the reverse-wire feature in a single wire pair is as follows:

- In the main node, set the A2B_CONTROL.XCVRBINV bit prior to writing to the A2B_SWCTL.ENSW bit. Be careful to avoid inadvertently clearing the A2B_CONTROL.XCVRBINV bit when writing to the A2B_CONTROL register for other purposes, such as writing to the A2B_CONTROL.NEWSTRCT bit.
- In any subordinate node, the A2B_CONTROL.XCVRBINV bit must be set before writing to the A2B_SWCTL.ENSW bit.

Once all of the subordinate nodes are discovered, initialize the nodes for synchronous data exchange. The example flow diagram starts initialization with the last-in-line node and finishes with the main node. The discovery finishes quickly, providing earlier access to all nodes and their I^{2}C peripherals, before the initialization for synchronous audio (which takes extra time to finish).

There is no further need for bus management after all of the nodes are discovered and programmed. Interrupt service routines can be used to react to special interrupt request (IRQ) events (for example, from an IO pin). Alternatively, the A2B_INTTYPE register can be polled to monitor interrupt events.

Response Cycles

The A2B_RESPCYCS register configures the relative time from the start of a synchronization control frame (SCF) to the moment the last subordinate node responds with a synchronization response frame (SRF). The register indicates to earlier nodes when to expect the response from the last subordinate node. If the last node does not respond, the previous node that is next to the presumed last node does respond.

The *Response Cycle Count* figure shows the window used to calculate the expected response time based on the position (probe point) of the node in the A^{2}B chain. In the example, subordinate node 2 has two speakers connected (spkr1 and spkr0). Other nodes have one speaker (spkr). Similarly, subordinate node 0 has two microphones connected (Mic1 and Mic 0). Other nodes have one microphonc only (MIC).

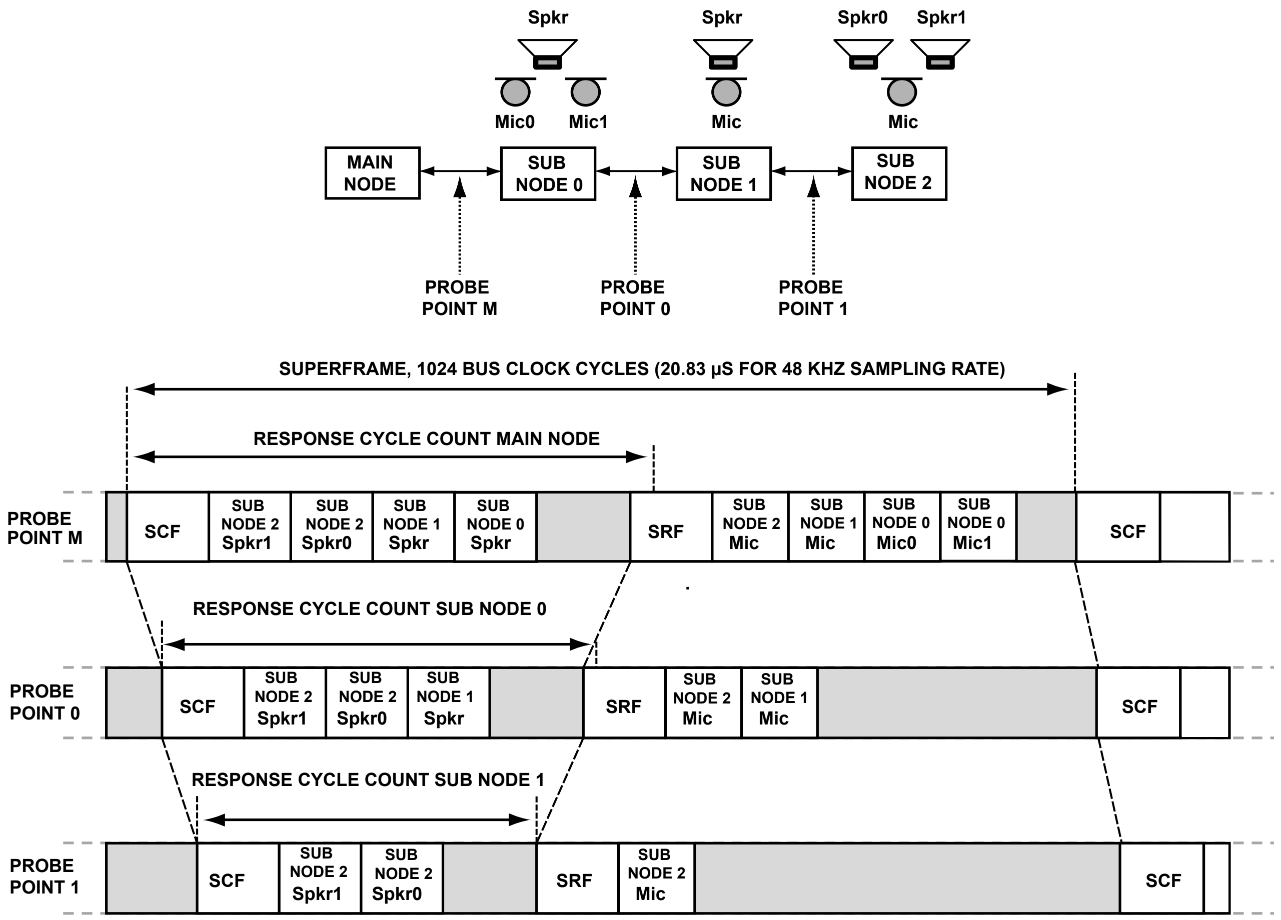


Figure 7-4: Response Cycle Count

The response cycle values for the transceivers are discussed in the following topic Response Cycle Formula as a function of the following parameters:

- Number of subordinate nodes
- Number of downstream slots
- Downstream slot size
- Number of upstream slots
- Upstream slot size
- Main Node I^{2}S/TDM channel configuration

NOTE: The main transceiver response cycle values are calculated using the above parameters in the response cycle calculator spreadsheet or in software. For more information, contact your local Analog Devices representative.

Subordinate Node Response Cycles

The ***Subordinate Node Response Cycle*** figure shows the relative timing between SCFs and SRFs on the A and B (XCVR) transceiver ports of a subordinate node. A subordinate node generates the SRF approximately ((4 `A2B_RESPCYCS`) + 7) bits after the SCF starts on the A transceiver. For example, when `A2B_RESPCYCS`= 128 (`0x80`), the subordinate node generates the SRF beginning at the 519th ((4 128) + 7 = 519) bit.

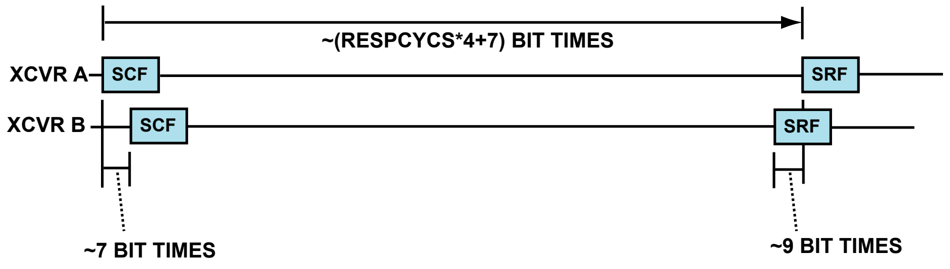


Figure 7-5: Subordinate Node Response Cycle

As shown in the ***Subordinate Node Response Cycle*** figure, there are transceiver delays (TD) incurred to pass the superframe from one side of the transceiver to the other. For the downstream portion of the superframe, there is a delay (TD_{DOWN}) of seven (± 2) bits incurred when going from the A-side to the B-side of the transceiver. Conversely, there is a delay (TD_{UP}) of nine (± 2) bits going from the B-side to the A-side during the upstream portion of the same superframe. These delays are summarized for the supported frame rates in the ***Transceiver Delays*** table, as governed by the equation:

Delay Range = Nominal Latency Range / (SYNC Rate 1024)

Table 7-1: Transceiver Delays

Time Delay (Direction)	SYNC Rate (kHz)	Nominal Latency Range (SYSBCLK)	Delay Range (ns)
TD_{DOWN} (A-Side to B-Side Downstream)	48.0	7 ± 2	101.7 183.1
TD_{UP} (B-Side to A-Side Upstream)	48.0	9 ± 2	142.4 223.8

In addition to these transceiver delays, cable delays (CD) between nodes also change the relative timing between when the SCF is received in the downstream potion of the superframe and when the complementary SRF returns to that point during the upstream portion of the same superframe. There is a 5-bit time window (expected bit time ± 2) in which the SRF is correctly received on the B-side and passed to the A-side of a subordinate node. An SRF outside of this window is still detected, and the expected response time is gradually (and automatically) adjusted by the transceiver during discovery to compensate for mismatches, with an adjustment range of -4 bit times to +15 bit times to span the cable length specifications.

The `A2B_RESPCYCS` formula works for all supported cable lengths. If the cable length is known during the system design phase, this recommendation can be applied for all discovery flows. If the cable lengths are unknown, the default response cycles calculation (assuming 4m cable length) is adequate. Although some errors can be observed during discovery (CRCERR, SRFMISSERR, or SRFCRCERR) when longer cables are used, the system runs cleanly after discovery completes due to this automatic adjustment capability.

The automatic response cycle adjustment performed during discovery works as follows:

1. The host programs the main to expect the SRF at the 519th bit of the superframe by setting `A2B_RESPCYCS` = 128 (`0x80`) , as detailed above ((4 128) + 7 = 519).
2. The main node initiates discovery of subordinate node 0 when the host writes `0x80` to its `A2B_DISCVRY` register. When subordinate node 0 starts sending SRFs, the main node adjusts its response time to align with subordinate node 0.

   Short cable lengths (up to 20 cm) do not impact the main node's ability to receive the SRF at the 519th bit of the superframe.

   Longer cable lengths, however, introduce a physical cable delay (CD) on the order of 5 ns/m to the time at which the SRF is captured at the receiving node. For example, a 10 m cable between the main node and the subordinate node 0 delays the SRF reception time at the main node by 100 ns (50 ns downstream CD plus 50 ns upstream CD). The 100 ns total CD equates to five A^{2}B bits. In this case, the main node adjust its response cycles to expect the SRF at the 524th (± 2) bit of the superframe.
3. The main node initiates discovery of subordinate node 1 when the host writes `0x7C` to the `A2B_DISCVRY` register. When subordinate node 1 starts sending SRFs, subordinate node 0 adjusts its response time to align with subordinate node 1. The adjustment causes the SRFs from subordinate node 0 to be delayed, thus adding further delay to the time at which the SRF reaches the main node.

   The main node receives the SRF as a function of the CD between the main node and subordinate node 0 and the CD between subordinate node 0 and subordinate node 1. Continuing with the above example, a second 10 m cable between subordinate node 0 and subordinate node 1 delays the SRF reception time at the main node by an additional five bits. The delay causes the main node to adjust its response cycles to expect the SRF at the 529th (± 2) bit of the superframe.

The ***SRF Response*** figure illustrates how cable and transceiver delays affect the SRF response. In this case, the SRF miss error is not observed because the response cycles are adjusted during the discovery phase.

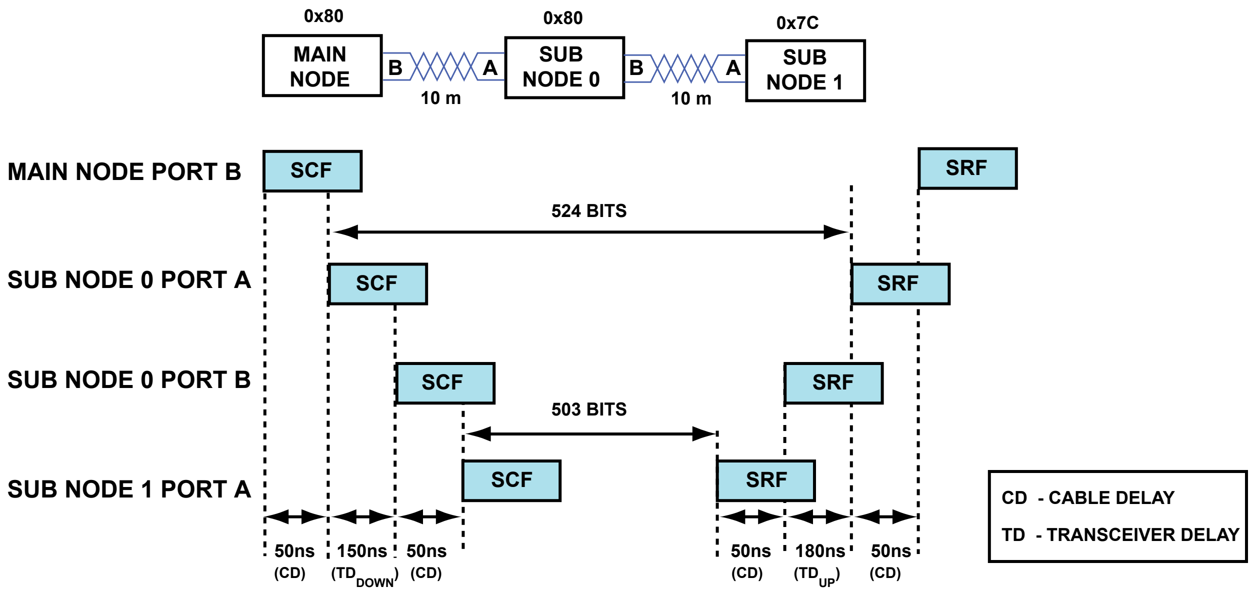


Figure 7-6: SRF Response

In this example:

- Subordinate node 1 is the last-in-line subordinate node, which is responsible for initiating the SRF to commence the upstream portion of the superframe. When programmed with `A2B_RESPCYCS` = 124 `(0x7C)`, subordinate node 1 is configured to generate the SRF at the 503rd bit of the superframe ((4 124) + 7 = 503).
- From the perspective of the upstream subordinate node 0, the total delay between the SCF arriving to the subordinate node 0 A-side of the transceiver during the downstream portion of the superframe and the corresponding SRF appearing there during the upstream portion of the same superframe is 430ns (21 bits). The delay is comprised of:
  - the downstream transceiver delay of subordinate node 0 (TD_{DOWN} = 150 ns),
  - the downstream cable delay between subordinate node 0 and subordinate node 1 (CD = 5 ns/m x 10 m = 50 ns),
  - the upstream cable delay between subordinate node 1 and subordinate node 0 (CD = 5 ns/m x 10 m = 50 ns), and
  - the upstream transceiver delay of subordinate node 0 (TD_{UP} = 180 ns)

  Therefore, the number of bits between SCF arrival to the subordinate node 0 A-side transceiver and the corresponding SRF being generated there is calculated to be 503 + 21 = 524 bits for a 10 m cable length between subordinate node 0 and subordinate node 1.
- From the perspective of the main node, the total delay between generating the SCF and the corresponding SRF appearing during the upstream portion of the same superframe is 100 ns (5 bits), which is comprised of:

- the downstream cable delay between the main node and subordinate node 0 (CD = 5 ns/m x 10 m = 50 ns) and
- the upstream cable delay between subordinate node 0 and the main node (CD = 5 ns/m x 10 m = 50 ns)

Therefore, the number of bits between SCF field generation and the corresponding SRF being received is calculated to be 524 + 5 = 529 bits.

Response Cycle Formula

The A2B_RESPCYCS register is used to set the relative time, from the start of a control frame (SCF) to the moment the last subordinate node responds with a response frame (SRF). The register setting defines when earlier nodes in the A^2B network should expect the response from the last subordinate node during the upstream portion of the superframe. If the last node fails to respond, the node immediately before the presumed last node does respond. The following sections provide information regarding how to program the main node and subordinate node A2B_RESPCYCS registers.

Configuring Main Node Response Cycles

The *Main Node Response Cycles* figure depicts how the main response cycle value is determined.

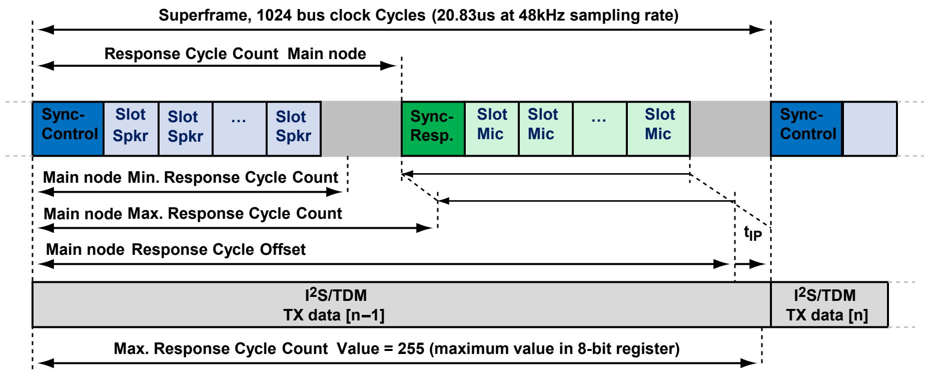

Superframe, 1024 bus clock Cycles (20.83us at 48kHz sampling rate)
Response Cycle Count Main node
Sync-Control
Slot Spkr
Slot Spkr
...
Slot Spkr
Sync-Resp.
Slot Mic
Slot Mic
...
Slot Mic
Sync-Control
Main node Min. Response Cycle Count
Main node Max. Response Cycle Count
Main node Response Cycle Offset
tIP
I2S/TDM TX data [n–1]
I2S/TDM TX data [n]
Max. Response Cycle Count Value = 255 (maximum value in 8-bit register)

Figure 7-7: Main Node Response Cycles

In the *Main Node Response Cycles* figure:

- The ***Main Node Minimum Response Cycle Count*** is determined by the length of the downstream data, the minimum bus turn-around time, and the number of subordinate nodes.
- The ***Main Node Maximum Response Cycle Count*** is determined by the length of the upstream data and the ***Main Node Response Cycle Offset***.
- The ***Main Node Response Cycle Offset*** ensures that sufficient internal processing time (t_{IP}) is available from the reception of the last upstream data bit in the receive buffer to the point at which this I^2S/TDM data is output, which starts synchronous to the next SCF and SYNC pin transition. The ***A^2B Main Node Response Offset (RESPOFFS)*** table defines the constant ***Main Node Response Cycle Offset***, which is a function of the A^2B main node's TDM mode A2B_I2SGCFG.TDMMODE) and I^2S/TDM channel size (A2B_I2SGCFG.TDMSS).

Table 7-2: A^{2}B Main Node Response Offset (RESPOFFS)

TDM Mode (A^{2}B Main Node)	TDM Data Width (A^{2}B Main Node)	RESPOFFS
TDM2/I^{2}S (`A2B_I2SGCFG.TDMMODE` = 0)	16 bits (`A2B_I2SGCFG.TDMSS` =1)	238
TDM2/I^{2}S (`A2B_I2SGCFG.TDMMODE` = 0)	32 bits (`A2B_I2SGCFG.TDMSS` =0)	245
TDM4 (`A2B_I2SGCFG.TDMMODE` = 1)	16 bits (`A2B_I2SGCFG.TDMSS` =1)	245
TDM4 (`A2B_I2SGCFG.TDMMODE` = 1)	32 bits (`A2B_I2SGCFG.TDMSS` =0)	248
TDM8 (`A2B_I2SGCFG.TDMMODE` = 2)	16 bits (`A2B_I2SGCFG.TDMSS` =1)	248
TDM8 (`A2B_I2SGCFG.TDMMODE` = 2)	32 bits (`A2B_I2SGCFG.TDMSS` =0)	248
TDM12 (`A2B_I2SGCFG.TDMMODE` = 3)	16 bits (`A2B_I2SGCFG.TDMSS` =1)	248
TDM12 (`A2B_I2SGCFG.TDMMODE` = 3)	32 bits (`A2B_I2SGCFG.TDMSS` =0)	248
TDM16 (`A2B_I2SGCFG.TDMMODE` = 4)	16 bits (`A2B_I2SGCFG.TDMSS` =1)	248
TDM16 (`A2B_I2SGCFG.TDMMODE` = 4)	32 bits (`A2B_I2SGCFG.TDMSS` =0)	248
TDM20 (`A2B_I2SGCFG.TDMMODE` = 5)	N/A	248
TDM24 (`A2B_I2SGCFG.TDMMODE` = 6)	N/A	248
TDM32 (`A2B_I2SGCFG.TDMMODE` = 7)	N/A	248

Programming the main node `A2B_RESPCYCS` register is a function of the ***Main Node Response Cycle Offset*** (RESPOFFS), as well as:

- the number of subordinate nodes in the system
- the number of downstream A^{2}B bus data slots received on the A-port at each subordinate node (NUM_DNSLOTS)
- the width of the downstream A^{2}B bus data slots (DNSLOT_SIZE),
- the number of upstream A^{2}B bus data slots driven to the A-port by each subordinate node (NUM_UPSLOTS)
- the width of the upstream A^{2}B bus data slots (UPSLOT_SIZE)

The upslot and downslot activity that is possible at any given node in the system is the first factor that contributes toward determining the value that must be programmed into the `A2B_RESPCYCS` register of the main node. For each subordinate node n in the A^{2}B topology, the following equations define the downstream (DNSLOT_ACTIVITY[n]) and upstream (UPSLOT_ACTIVITY[n]) activity for that node.

```
DNSLOT_ACTIVITY[n] = NUM_DNSLOTS * (DNSLOT_SIZE + 1)
UPSLOT_ACTIVITY[n] = NUM_UPSLOTS * (UPSLOT_SIZE + 1)
```

NOTE: The DNSLOT_SIZE and UPSLOT_SIZE slot sizes are offset by 1 in the calculations because the default slot format (`A2B_SLOTFMT`) appends a single parity bit to each data slot on the A^{2}B bus, thereby increasing the number of bits on the A^{2}B bus per slot by 1. For alternate slot formats, the number of bits that are appended for the chosen use case must be added instead of the 1 defined here. See the Table 4-3 Slot Formatting table.

Once the upslot and downslot activity for each subordinate node n is established, the equivalent upstream (RESPCYCS_UP[n]) and downstream (RESPCYCS_DN[n]) response cycle requirements can be calculated for each subordinate node.

- RESPCYCS_DN[n] is the minimum response cycle register setting possible at the main node when considering the downstream activity at subordinate node n. The maximum value among those calculated for RESPCYCS_DN[n] is the minimum main node `A2B_RESPCYCS` setting (MAX(RESPCYCS_DN[n])).
- RESPCYCS_UP[n] is the maximum response cycle register setting possible at the main node when considering the upstream activity at subordinate node n. The minimum value among those calculated for RESPCYCS_UP[n] is the maximum main node `A2B_RESPCYCS` setting (MIN(RESPCYCS_UP[n])).

CAUTION: If MAX(RESPCYCS_DN[n]) > MIN(RESPCYCS_UP[n]), the A^{2}B bus bandwidth cannot accommodate the configuration.

The value that must be programmed into the main node's `A2B_RESPCYCS` register is the average of these minimum and maximum values:

```
A2B_RESPCYCS = (MAX(RESPCYCS_DN[n]) + MIN(RESPCYCS_UP[n])) / 2   // Round Down
```

Configuring Sub Node Response Cycles

Each subordinate node has its `A2B_RESPCYCS` register set during the system discovery process. The main transceiver programs its `A2B_DISCVRY` register with the response cycle value associated with the subordinate transceiver that it is attempting to discover. The appropriate value for each subordinate node (SLV_RESPCYCS[n]) is a function of the subordinate node location in the A^{2}B topology and the value programmed to the main node 's `A2B_RESPCYCS` register (MSTR_RESPCYCS). The subordinate node nearest to the main node has a node number of 0, and the node number is incremented for each next-in-line subordinate node until the last-in-line subordinate node n. The `A2B_RESPCYCS` value to use for each subordinate node during discovery can be calculated using the following equations.

The following code sequence uses these values to proceed through the discovery process in the example system:

```
Write MSTR_RESPCYCS to the A2B_RESPCYCS register in the main node
Write 0x01 to the A2B_CONTROL register in the main node
Write 0x01 to the A2B_SWCTL register in the main node
Write 0x01 to the A2B_INTMSK2 register in the main node
Write SLV_RESPCYCS[0] to the A2B_DISCVRY register in the main node
 // Wait for Interrupt

Write 0x00 to the A2B_NODEADR register in the main node
Write 0x01 to the A2B_SWCTL register in sub node 0
Write SLV_RESPCYCS[1] to the A2B_DISCVRY register in the main node
  //Wait for Interrupt>
```

The equivalent upstream (RESPCYCS_UP[n]) and downstream (RESPCYCS_DN[n]) response cycle at subordinate node 0 when considering each subordinate node can be calculated using the following equations :

```
RESPCYCS_DN[n] = {[  (64 + DNSLOT_ACTIVITY[n] + Turnaround Time +
      (n * Downstream Propagation Delay ) +
      (n * Upstream Propagation Delay) +
      (Cable delay per meter *
      (Sum of length of cables from main node to node n) / Bit Time) +
      (Cable delay per meter *
      (Sum of length of cables from sub node 0 to node n) / Bit Time)  ) -
      (1 + (Cable delay per meter *
      (Length of cables from main node to sub node 0) / Bit Time) ] - 7} /4
      //Round Up

      RESPCYCS_UP[n] = { [  RESPOFFS × 4 -
      ( 64 + UPSLOT_ACTIVITY[n] + Turnaround Time +
      (Cable delay per meter × Length of cables from main node to sub node 0 /
Bit Time))  ]  -7 } /4
      // No Rounding
```

Example Main Node A2B_RESPCYCS Calculation

A system with three nodes, the main node and two subordinate nodes (subordinate node 0 and subordinate node 1), is configured as shown in the *Three-Node A^{2}B System Example* figure. The cable length between the main node and subordinate node 0 is 10 m. The cable length between subordinate node 0 and subordinate node 1 is 5 m.

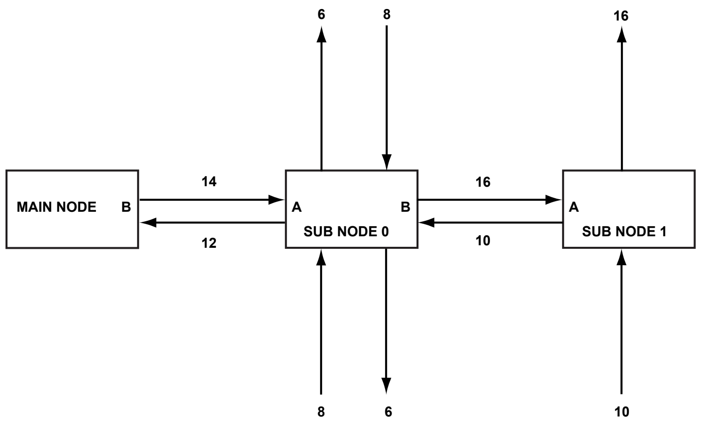


Figure 7-8: Three-Node A^{2}B System Example

For the downstream portion of the superframe:

- Main node (configured for 32-bit TDM8 mode): sends 14 slots with a 24-bit slot size
- Sub node 0: consumes six slots from the main node and passes the remaining eight slots to subordinate node 1, then contributes eight additional slots to the downstream traffic (16 total slots sent from subordinate node 0 to subordinate node 1)

- Sub node 1: consumes all 16 slots coming from subordinate node 0

For the upstream portion of the superframe:

- Sub node 1: sends ten slots with a 16-bit slot size
- Sub node 0: consumes six slots from subordinate node 1 and passes the remaining four slots to the main node, then contributes eight additional slots to the upstream traffic (12 total slots sent from subordinate node 0 to the main node).
- Main node (configured for 32-bit TDM8 mode): consumes all 12 slots coming from subordinate node 0.

The response cycle uses formula B and the following steps:

1. Calculate the upslot and downslot activity for each subordinate node:

```
DNSLOT_ACTIVITY[n] = NUM_DNSLOTS × (DNSLOT_SIZE + 1)
DNSLOT_ACTIVITY[0] = 14 × (24 + 1) = 350
DNSLOT_ACTIVITY[1] = 16 × (24 + 1) = 400
UPSLOT_ACTIVITY[n] = NUM_UPSLOTS × (UPSLOT_SIZE + 1)
UPSLOT_ACTIVITY[0] = 12 × (16 + 1) = 204
UPSLOT_ACTIVITY[1] = 10 × (16 + 1) = 170
```

2. With this information, calculate the response cycle requirements for each subordinate node. From the Table 7-2 A^{2}B Main Node Response Offset (RESPOFFS) table, the TDM8 mode and 32-bit data combination yields RESPOFFS = 248.

   This formula uses average values for the system:

   - Turnaround Time = 8
   - Downstream Propagation Delay = 7.5
   - Upstream Propagation Delay = 8.3
   - Cable delay per meter = 6
   - Bit Time = 20.35
   - Total length = 15

```
RESPCYCS_DN[n] = {[  (64 + DNSLOT_ACTIVITY[n] + Turnaround Time +
(n × Downstream Propagation Delay ) +
(n × Upstream Propagation Delay) +
(Cable delay per meter ×
(Sum of length of cables from main node to node n) / Bit Time) +
(Cable delay per meter ×
(Sum of length of cables from sub node 0 to node n) / Bit Time)  ) -
(1 + (Cable delay per meter ×
(Length of cables from main node to sub node 0) / Bit Time) ] - 7} /4

RESPCYCS_DN[0] = { [ (64 + 350 + 8 + ( 0 × 7.5) + ( 0 × 8.3) +
( 6×10/20.35) ) - ( 1 + (6 × 10/ 20.35)) ] - 7} /4 = 104
```

```
RESPCYCS_DN[1]= { [ (64 + 400 + 8 + ( 1 × 7.5) + ( 1 × 8.3) + ( 6×15/20.35) +
( 6×5/20.35)) - ( 1 + (6 × 10/ 20.35)) ] - 7} /4 = 121


RESPCYCS_UP[n] = { [  RESPOFFS×4 -
( 64 + UPSLOT_ACTIVITY[n] + Turnaround Time +
(Cable delay per meter × Length of cables from main node to sub node 0 / Bit
Time))  ]  -7 } /4

RESPCYCS_UP[0] = { [ 254 × 4 – ( 64 + 204 + 8 + (6×10/20.35)) ] -7 } /4 =
181.5
RESPCYCS_UP[1] = { [ 254 × 4 – ( 64 + 170 + 8 + (6×10/20.35)) ] -7 } /4 = 190
```

The minimum main node A2B_RESPCYCS setting is the maximum value among the RESPCYCS_DN[n] calculations, which is 121. The maximum setting is the minimum value among the RESPCYCS_UP[n] calculations, which is 181.5. The average of the minimum and maximum values is:

```
A2B_RESPCYCS = (MAX(RESPCYCS_DN[n]) + MIN(RESPCYCS_UP[n])) / 2
// Round Down
```

In this example, A2B_RESPCYCS = (121 + 181.5)/2 = 302.5/2 = 151.25 = 151.

3. For this system configuration, program the main node A2B_RESPCYCS value to 151 (0x97).

4. Calculate the subordinate node A2B_RESPCYCS value for each subordinate node.

```
SLV_RESPCYCS[0] = MSTR_RESPCYCS
```

For all other nodes,

SLV_RESPCYCS[n] = ((((MSTR_RESPCYCS 4) + 7) – ((2 cable delay per meter (Sum of cable length from subordinate node 0 to node n) / Bit time) +

(n downstream propagation delay) + (n upstream propagation delay))) – 7) / 4

Using the *Example Main Node* A2B_RESPCYCS *Calculation* (with MSTR_RESPCYCS = 151), the following equations determine the correct A2B_RESPCYCS value for the two subordinate nodes:

```
SLV_RESPCYCS[0] = MSTR_RESPCYCS = 151 (0x97)
        SLV_RESPCYCS[1] = ((((MSTR_RESPCYCS × 4) + 7)
        – ((2 × 5 × 6 / 20.35) + (1 × 7.5) + (1 × 8.3))) - 7 ) / 4 = 146
```

8 Line Fault Diagnostics

This section discusses the A^{2}B line fault diagnostics. It provides descriptions of the different faults and programming instructions for responding to line fault events in software.

Diagnostics During Discovery

The ***Line Faults*** tables shows the different types of line faults and the affected pins. In cases where a single wire pair pair is used, all faults can be detected during discovery of the bus. All faults, except some critical faults, can be localized during discovery of the bus. When a fault is detected during discovery, the switches that enable the bias current to the next-in-line transceiver are disconnected automatically.

The different line faults are identified in the A2B_INTTYPE register at the main transceiver and by the A2B_SWSTAT register in the subordinate node transceiver reporting the fault. In cases where a single wire pair pair is used, open wire and reverse wire faults are indicated by the A2B_INTTYPE register value of 0x0E. Timeout during discovery also occurs when an invalid value is programmed in the A2B_DISCVRY.DRESPCYC bit field, or if the next-in-line node has a physical defect that prevents the transceiver from responding.

NOTE: The A2B_SWCTL.ENSW bit is not cleared automatically when a line fault opens the bias switches; this has operation must be done in software. The A2B_SWCTL.ENSW bit in the main transceiver should be disabled (= 0) in the event of a critical line fault to disconnect bus bias to any bus segments.

CAUTION: The short to ground and short to V_{BUS} faults are critical faults for which the whole bus shuts off. Normal A^{2}B bus operation must be discontinued, including removal of bus power by the main node (independent of line fault location). Program the A2B_SWCTL.ENSW = 0 to the main transceiver until the fault is corrected.

For the following faults in a single wire pair, partial A^{2}B bus operation can continue between the main and subordinate nodes which are upstream of the line fault location.

- Open
- Reverse wires
- Defective node
- Wrong discovery parameter for next-in-line node

- Wires shorted together

NOTE: The LPS (Local Power Subordinate Node) is isolated from its B-port. This prevents the fault occurred at the B-port or the fault occurred at a downstream bus-powered node to back-propagate to previous nodes. In case of a critical fault, the bus can partially operate until the LPS node, which reports the fault or is the last LPS before the fault. However, if the downstream nodes are contributing upstream slots to the main node or any other upstream node present between the main and the LPS, then parity errors seen in cases of partial operation of the bus occur.

Registers for Line Diagnostics

The following registers are used to diagnose line faults on the A^{2}B bus. Refer to the *Register Descriptions* section for details.

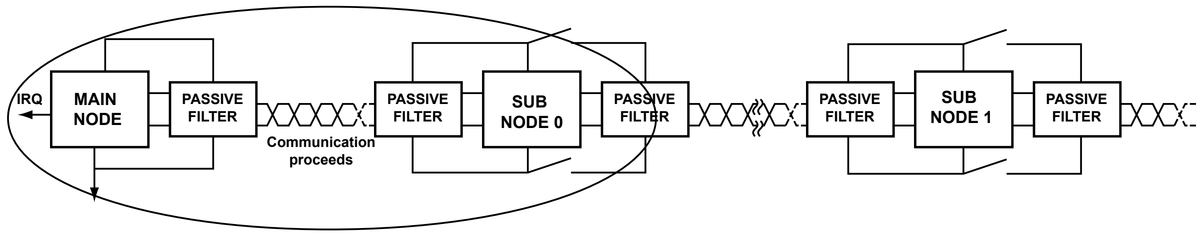
- The A2B_SWCTL register controls the discovery of the next-in-line transceiver on the A^{2}B bus link. It provides line fault sensing modes as well as the controls for localizing faults. The register provides an option to override the detected power configuration.
- The A2B_SWSTAT register provides line diagnostics status information.
- The A2B_INTSRC register contains information about the source of an active interrupt, which subordinate node generated it, or whether the interrupt originates from the main node. Line errors can be located with this register.
- The A2B_INTTYPE register stores information about the type of the current interrupt request. A read of this register clears the corresponding interrupt.

Line Faults in Single Wire Based Systems

Open Wire Fault

The ***Open Wire Fault*** figure shows an open wire fault between subordinate node 0 and subordinate node 1 transceivers. Communication continues between the main node and subordinate node 0 transceivers when this fault occurs between subordinate node 0 and subordinate node 1.

Figure 8-1: Open Wire Fault



Short of Wires Fault

The *Short of Wires Fault* figure shows a short of wires line fault between subordinate node 0 and subordinate node 1. Communication continues between the main node and subordinate node 0 transceivers when this fault occurs between subordinate node 0 and subordinate node 1.

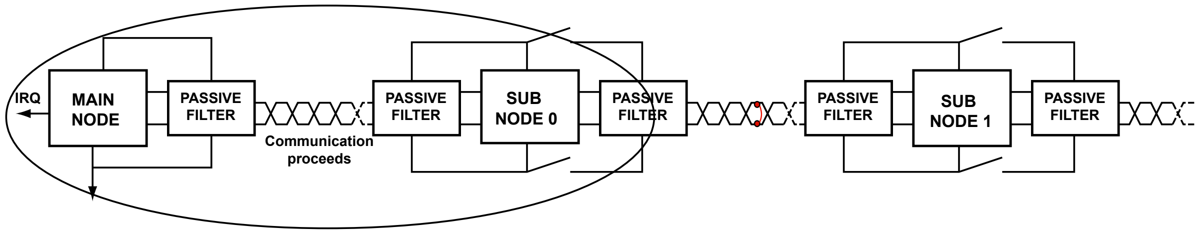


Figure 8-2: Short of Wires Fault

Short To GND BP

The *Short to GND BP* fault figure shows the BP wire shorted to ground between subordinate node 0 and subordinate node 1 transceivers. All bus communication stops when this fault occurs between subordinate node 0 and subordinate node 1 transceivers.

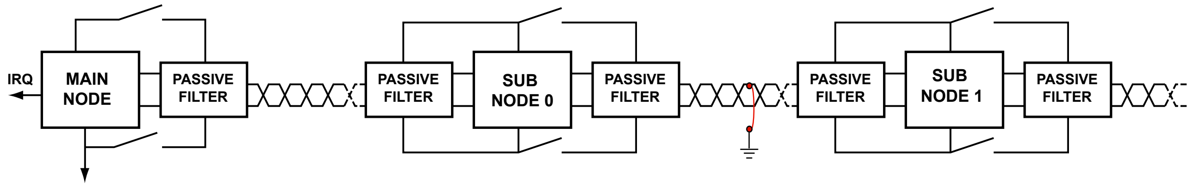


Figure 8-3: Short To GND BP

Short to VBUS

The Short to VBUS occurs when either the BP or BN wire is shorted to VBUS between subordinate node 0 and subordinate node 1 transceivers.

Short To GND BN

The *Short to GND BN* figure shows the BN wire shorted to ground between subordinate node 0 and subordinate node 1 transceivers. All bus communication stops when this fault occurs between subordinate node 0 and subordinate node 1 transceivers.

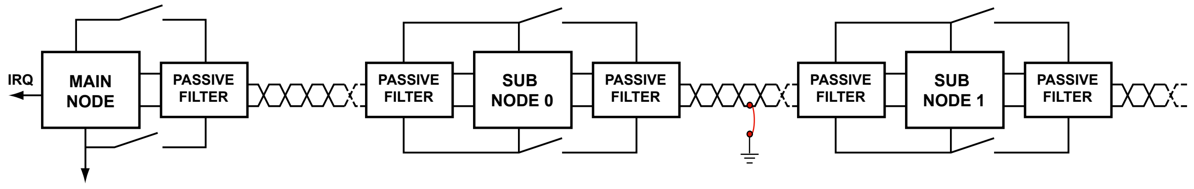


Figure 8-4: Short to GND BN

The following table shows the different types of line faults in a single wire pair system.

Table 8-1: Types of Line Faults in Single Wire Pair System

Wires	Affected pins	Detect	Localize	INTTYPE	Remarks
Partial bus operation may continue for nodes upstream of the fault					
Open	BP	Yes	yes	0x0E	Open/Reverse/Wrong port detected
	BN				
	BN and BP				
Wrong Port	B to B` port	Yes	Yes	0x0E or No 0x18 timeout (no DCSDONE interrupt)	
Reverse Wires	BN to AP and BP to AN	Yes	Yes	0x0E	
				No 0x18 timeout (no DCSDONE interrupt)	
Defective Node	N/A	Yes	Yes	No 0x18 timeout (no DCSDONE interrupt)	Defective node or wrong software parameter value for `A2B_DISCVRY.DRESPCYC`
Short of Wires	BP with BN	Yes	Yes	0x0B	Wires shorted together
Critical Faults					
Short to Ground	BP	Yes	Yes	0x09 or 0x29	Cable wire shorted to ground; software does not localize this fault
	BN		Yes		
Short to VBUS	BN	Yes	No	0x0A or 0x2A	Cable wire shorted to VBUS; software does not localize this fault
	BP				

Line Faults in XLR/DMX and RJ45 CAT Cable-based Systems

Line faults in XLR/DMX and RJ45 CAT cable-based systems can be categorized as no next node faults or critical short faults. The *Line Fault* tables show the detailed fault types, fault indicators, and associated registers. *Node n* in the tables refers to the discovery node; *node n+1* refers to the next-in-line node.

No Next Node

The no next node fault occurs when the port is not connected or connected to the wrong port or if there is a defective node. When this fault occurs between subordinate node 0 and subordinate node 1, communication continues between the main and subordinate node 0 transceivers.

The following type of line faults are identified under No Next Node.

- Open cable
- VBUS is disconnected
- BN (Pin 5 - BN) disconnected (RJ45 based system)

- No ground connection
- Wrong port

Critical Short

The critical short fault occurs when the wires are shorted to ground or shorted together. In this case all bus communications stop between subordinate node 0 and subordinate node 1.

The following type of line faults are identified under critical short.

- Crossover cable
- Pins shorted to supply
- Pins shorted to ground
- BP pin 4 shorted to BN pin 5 (RJ45 based system)

Table 8-2: Line Faults - RJ45 CAT Cable-based Systems

Fault Type	Fault	Register	Fault Indication	Comments
No Next Node	Open cable	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	VBUS is disconnected	(Write node n+1) A2B_VMTR_VEN = 0x02 (Read node n+1) A2B_VMTR_VLTG1	A2B_VMTR_VLTG1 on sub node < 4V	Enable VMTR after second stage discovery (after 0x18) and read VBUS. Will be close to 0
	BN pin 5 (5V) is disconnected*1	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	No ground connection	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	Wrong port	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	

Table 8-2: Line Faults - RJ45 CAT Cable-based Systems (Continued)

Fault Type	Fault	Register	Fault Indication	Comments
Critical Short	Crossover cable	(Read node n) A2B_GPIOIN	GPIOIN will read 0x00	No power good signal on 5V regulator.
	VBUS pin shorted to supply	(Read main) A2B_INTTYPE	0x2A or 0x0A during second stage discovery	
	BP pin 4 shorted to BN pin 5 [*1,*2]	(Read node n) A2B_GPIOIN	GPIOIN will read 0x00	No power good signal on 5V regulator.
	VBUS pin is shorted to GND	(Read main) A2B_INTTYPE	0x09 or 0x29 during second stage discovery	
	BN pin 5 (5V) is shorted to GND	(Read node n) A2B_GPIOIN	GPIOIN will read 0x00	No power good signal on 5V regulator.

*1 Refer to the *AD2437 RJ45 Reference Schematic* for pin numbering.

*2 Same fault scenario as port B pin 4 shorted to port A pin 5 and port B pin 5 shorted to port A pin 4.

Table 8-3: Line Faults - XLR Cable-based Systems

Fault Type	Fault	Register	Fault Indication	Comments
No Next Node	Open cable	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	VBUS is disconnected	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	No ground connection	(Read main) A2B_INTTYPE	Timeout during first stage discovery (No 0x18)	
	Reverse data wire connection[*1]	(Read main) A2B_INTTYPE	Timeout during first stage discovfirstery (No 0x18)	
Critical Short	BN or BP pin shorted to Supply	(Read node n) A2B_GPIOIN	GPIOIN will read 0x00	No power good signal on 5V regulator.
	BN or BP pin is shorted to GND	(Read node n) A2B_GPIOIN	GPIOIN will read 0x00	No power good signal on 5V regulator.

*1 BN shorted to AP and BP shorted to AN

Line Diagnostics After Discovery

Full line diagnostics are only performed during discovery. However, certain interrupts (if enabled) after discovery can indicate line faults during operation. Rediscovery detects the cause and location of the faults which are can be localized.

After discovery, any of the following interrupt types (A2B_INTTYPE) indicate that there is a line fault:

- 0x0A (10: PWRERR)
- 0x0F (15: PWRERR)
- 0x2A (42: PWRERR)
- 0x80 (128: interrupt messaging error)
- 0xFD (253: subordinate node INTTYPE read error)

When a subordinate node detects the SRF missed error (SRFMISSERR) in 32 consecutive frames, the node assumes a downstream bus drop and sets its last node bit (A2B_NODE.LAST = 1) to become the last node in the system. A bus drop condition resulting from a line fault occurring after discovery can be detected in a last node (A2B_NODE.LAST = 1) that has the SRFMISSERR latched.

Excessive accumulation of bit errors can happen if there is a slot configuration mismatch between nodes. This can also happen when positive A^2B wire shorts to a noisy supply or negative A^2B wire shorts to a noisy GND. The bus can operate under these conditions but is more susceptible to impairments (for example, electromagnetic interference).

Use the A2B_BECNT register to count accumulated errors as follows.

- Set the A2B_BECCTL register to 0xE4 (interrupt after 256 CRC errors). Acceptable audio noise and robustness is subjective and needs to be determined in vehicle tests. Adjust the threshold accordingly.
- Periodically write 0 to the A2B_BECNT register (once every second) to reset the error counter. Acceptable audio noise and robustness is subjective and needs to be determined in vehicle tests. Adjust time for the A2B_BECNT register accordingly.
- The bit error counter overflow (0x04: BECOVF) interrupt indicates bus issues.

Diagnostics Software Flow - Single Wire Pair System

Use the following software flow and the ***Diagnostics Software Flow*** figure for node discovery with diagnostics.

1. Configure A2B_SWCTL.MODE = 1 for diagnostics mode 1. Enable the power error interrupts and the A2B_INTPND2.DSCDONE interrupt in the main node.
2. Set A2B_SWCTL.ENSW = 1 to enable the power switch.
3. Wait for an interrupt to occur. If A2B_INTTYPE = 0x18 for A2B_INTPND2.DSCDONE (indicating a successful node discovery), proceed to step 10.
4. If A2B_INTTYPE = 0x0A or 0x2A, configure A2B_SWCTL.ENSW = 0 in the next upstream local powered node to disable the bus (after the A2B_INTSRC and A2B_INTTYPE register values have been communicated to the host). Proceed to step 9.

5. If A2B_INTTYPE = 0x0F, it may be due to a short to VBUS fault in a bus-powered node downstream to the present node. When this fault occurs, configure the A2B_SWCTL.ENSW = 0 in the next upstream local powered node to disable the bus (after the A2B_INTSRC and A2B_INTTYPE register values have been communicated to the host). In this case, the fault cannot be localized. Proceed to step 9.

   ADDITIONAL INFORMATION: Refer to Using VMTR ADC for Bus Monitoring for information on how to use the VMTR ADC to monitor the VBUS voltage and bus current to detect line faults.

6. If A2B_INTTYPE = 0x09 or 0x29, clear the A2B_SWCTL.ENSW bit in the next upstream local powered node to disable the bus (after the A2B_INTSRC and A2B_INTTYPE register values have been communicated to the host). Proceed to step 9.

7. If A2B_INTTYPE = 0x0B, read the A2B_INTSRC register to determine the location. Proceed to step 9.

8. If the discovery process times out, configure A2B_SWCTL.ENSW = 0 in the current node and wait 250 ms. Proceed to rediscovery with A2B_SWCTL.MODE = 0 for diagnostics mode 0 and set the A2B_SWCTL.ENSW bit = 1. Wait for an interrupt to occur and read the A2B_INTTYPE register.

   - If the A2B_INTTYPE register = 0x0E, read the A2B_INTSRC register to determine the location.
   - If the process times out, the problem is a reverse wire or wrong-port, or the node being discovered is faulty.
   - If the A2B_INTTYPE register indicates any A2B_INTPND0.PWRERR interrupt other than 0x18 or if there is a timeout while discovering a sub node, stop the discovery process by setting the A2B_CONTROL.ENDDSC bit.

9. Stop the discovery process and display the error type (A2B_INTTYPE) and location of the fault (A2B_INTSRC).

   ADDITIONAL INFORMATION: Once any other localized fault has been detected, halt the discovery process. Software can retry the discovery process periodically to determine whether the fault is cleared. There is no automatic retry mechanism within the transceiver.

10. If this is not the last-in-line node and A2B_INTTYPE = 0x18, reprogram the A2B_SWCTL.MODE bits = 2. This configuration ignores the fluctuation on VBUS due to downstream current draw. It prevents incorrect localization on errors that occur on nodes that are located further downstream. Program the downstream node register settings and repeat step 1 on the next-in-line node.

    ADDITIONAL INFORMATION: Continue this cycle until all nodes are discovered. Once all nodes are discovered, configure A2B_SWCTL.MODE = 1 to all nodes while keeping A2B_SWCTL.ENSW = 1. Full A^{2}B bus discovery is now complete.

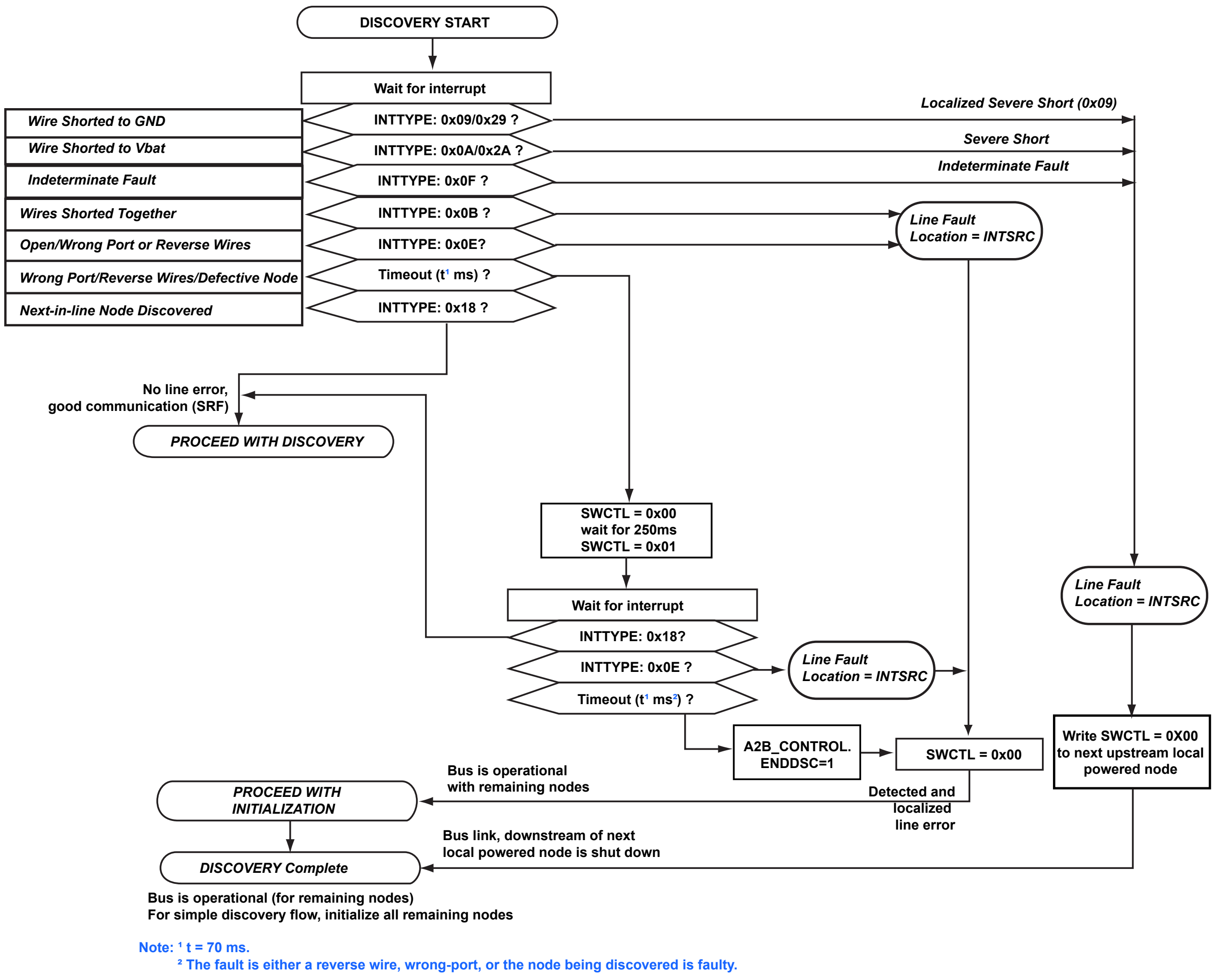

DISCOVERY START
Wait for interrupt
INTTYPE: 0x09/0x29 ?
INTTYPE: 0x0A/0x2A ?
INTTYPE: 0x0F ?
INTTYPE: 0x0B ?
INTTYPE: 0x0E?
Timeout (t¹ ms) ?
INTTYPE: 0x18 ?
Wire Shorted to GND
Wire Shorted to Vbat
Indeterminate Fault
Wires Shorted Together
Open/Wrong Port or Reverse Wires
Wrong Port/Reverse Wires/Defective Node
Next-in-line Node Discovered
Localized Severe Short (0x09)
Severe Short
Indeterminate Fault
Line Fault Location = INTSRC
No line error, good communication (SRF)
PROCEED WITH DISCOVERY
SWCTL = 0x00 wait for 250ms SWCTL = 0x01
Wait for interrupt
INTTYPE: 0x18?
INTTYPE: 0x0E ?
Timeout (t¹ ms²) ?
Line Fault Location = INTSRC
Line Fault Location = INTSRC
A2B_CONTROL. ENDDSC=1
SWCTL = 0x00
Write SWCTL = 0X00 to next upstream local powered node
Bus is operational with remaining nodes
Detected and localized line error
PROCEED WITH INITIALIZATION
Bus link, downstream of next local powered node is shut down
DISCOVERY Complete
Bus is operational (for remaining nodes)
For simple discovery flow, initialize all remaining nodes
Note: ¹ t = 70 ms.
² The fault is either a reverse wire, wrong-port, or the node being discovered is faulty.

Figure 8-5: Diagnostics Software Flow

Diagnostics Software Flow - XLR/DMX and RJ45/CAT Cable-based System

Use the following software flow and the ***Diagnostics Software Flow*** figure for node discovery with diagnostics.

1. Check the A2B_GPIOIN register for power good signal from the 5V regulator. If A2B_GPIOIN = 0x00, proceed to step 10.
2. Configure A2B_SWCTL.MODE = 1, A2B_CONTROL.XCVRBINV = 1 and A2B_CONTROL.SWBYP = 1 .
3. Set A2B_SWCTL.ENSW = 1 for first stage discovery.
4. Wait for an interrupt to occur. If A2B_INTTYPE = 0x18 for A2B_INTPND2.DSCDONE (indicating a successful node discovery), proceed to step 6.
5. If the discovery process times out, proceed to step 11.

6. Check for a valid sub node by reading the EEPROM on the sub node. If invalid, stop the discovery and report the error.
7. Disable A2B_SWCTL.ENSW and A2B_CONTROL.SWBYP and then set the A2B_SWCTL.ENSW = 1 again for second stage discovery
8. Wait for an interrupt to occur. If A2B_INTTYPE = 0x18, enable VMTR and read A2B_VMTR_VLTG1 on the discovered node. If the voltage is less than 3V, proceed to step 11
9. If A2B_INTTYPE = 0x0A, 0x2A, 0x09, or 0x29, proceed to step 10.
10. Stop discovery and report the critical short line fault. See Line Faults in XLR/DMX and RJ45 CAT Cable-based Systems .
11. Stop discovery and report the no next node line fault. See Line Faults in XLR/DMX and RJ45 CAT Cable-based Systems.
12. If this is not the last-in-line node and A2B_INTTYPE = 0x18, reprogram the A2B_SWCTL.MODE bits = 2. This configuration ignores fluctuation on VBUS due to downstream current draw; it prevents incorrect localization on errors that occur on nodes that are located further downstream. Program the downstream node register and repeat step 1 on the next node.

Using VMTR ADC for Bus Monitoring

VMTR (Voltage Monitor) ADC can be used to monitor bus conditions such as bus-bias (VBUS) and bus current (high-side and low-side current). Interrupts can be generated based on a pre-set threshold when a bus-bias or bus-current goes above or below the threshold.

Example 1: Monitoring bus-bias

1. Use VMTR ADC to monitor the VBUS voltage by writing A2B_VMTR_VEN = 0x02 (enable VLTG1 channel).
2. Enable VMTR interrupt for VBUS channel by writing A2B_VMTR_INTEN = 0x02.
3. Set the threshold for generating a max interrupt by using the A2B_VMTR_VMAX1 register. For example, if we want to generate an interrupt whenever VBUS goes above 24V, write A2B_VMTR_VMAX1 = 0xC0.

Example 2: Monitoring bus current in the nodes

1. Enable VMTR ADC to monitor high and low-side bus current by writing A2B_VMTR_VEN = 0x60.
2. Enable VMTR interrupt for a high-side and low-side current channel by writing A2B_VMTR_INTEN = 0x60.
3. Configure a threshold for generating a max interrupt by using the A2B_VMTR_VMAX5 and A2B_VMTR_VMAX6 registers. For example, configure A2B_VMTR_VMAX5 = 0x35 to generate an interrupt whenever the low-side current goes above ~2A. Similarly, configure A2B_VMTR_VMAX6 = 0x6A to generate an interrupt whenever the high-side current goes above ~2A.

In the nodes, a BN short to GND or BP short to VBAT fault that occurred after discovery can be detected by monitoring the difference in high-side and low-side current.

For example:

- For a BN short to GND, the high-side current is higher than the low-side current.
- For a BP short to VBAT, the low-side current is higher than the high-side current.

Bus Drop Detection

To detect a dropped node during run time, periodically read registers such as A2B_VENDOR and A2B_PRODUCT from the sub nodes. When a sub node is dropped, an I^2C/SPI read request fails with an I2CERR/SPIREGERR error. The ***Diagnostics Software Flow*** figure shows the flow of bus drop detection in the A^2B system.

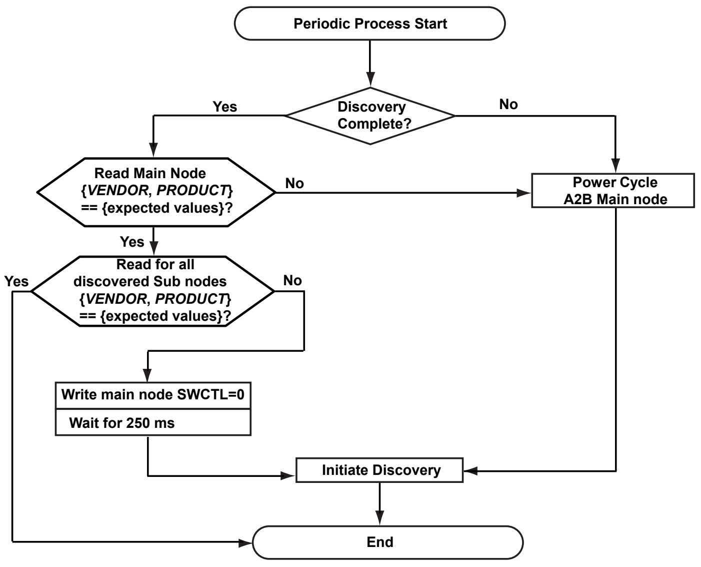

Periodic Process Start
Discovery Complete?
Yes
No
Read Main Node {VENDOR, PRODUCT} == {expected values}?
No
Power Cycle A2B Main node
Yes
Read for all discovered Sub nodes {VENDOR, PRODUCT} == {expected values}?
Yes
No
Write main node SWCTL=0
Wait for 250 ms
Initiate Discovery
End

Figure 8-6: Diagnostics Software Flow

A dropped node can also be detected by the host. Once the host detects an SRFMISSERR error, it can read the A2B_NODE register of the node flagging the error (the A2B_INTSRC indicate which node is flagging the SRFMISSERR error). If A2B_NODE.LAST =1, it indicates that nodes downstream of this node are disconnected.

Node Drop Detection

When a subordinate node drops from the A^2B bus in the run time, the host processor detects this drop using either a polling-based approach or an interrupt-based approach.

Polling-Based Approach

Using a polling-based approach, the host processor periodically reads the known registers from the A^2B nodes on the bus to confirm if the node status (connected or dropped).

Use the following sequence inside the polling routine:

1. Read a known register, that changes during that node initialization process, from the main node. An example is the A2B_I2SCFG or the A2B_DATCTL register.
    a. If the main node read accesses returns NAK, try the accesses again. If repeated accesses return NAK, the node must be in the reset state due to a power-supply drop or a RST pin assertion.

       In this case, bring the main node to the power-up state and then attempt the node rediscovery process.
    b. If the register reads returns power-on default value instead of one configured during node initialization process, then reset the main node and bring it up in the power-up state.

       In this case, the node rediscovery process can be attempted.
2. If the main node register reads are fine, then read the known registers from the first subordinate node (subordinate node 0). Choose a register that has a constant value, for example the VENDOR_ID or PRODUCT_ID.
    a. If the read access is successful and it has the expected value (for example, VENDOR_ID = 0xAD), then read the registers from next subordinate node. Continue this step until the last node is successfully read back with expected values.
    b. If the read access is successful (no NAK), but the register read-back value is not (for example, 0x00), then check for main node reset by reading a known register from main node. Note that, when the main node resets and returns to the power-up state, the bus read accesses (with BUS_ADDR) return the value 0x00 without any NAK to access or I2CERR.
    c. If the read is not successful (NAK), then the subordinate node lost connection. In this case, the attempt either full node rediscovery or partial rediscovery (of only the dropped nodes).
3. If there are many subordinate nodes and performing read accesses to all the nodes becomes a heavy task for the host processor, then first read the known registers from last node.
    a. If the registers from last node are successfully read, the A^2B chain is intact and with no bus drop.
    b. If the register reads are not successful, the localize the node drop and start either full rediscovery or partial rediscovery of dropped nodes only by returning to steps 1 and 2.

Interrupt-Based Approach

With an interrupt-based approach, the host processor gets SRFMISS errors (for example, INTTYPE = 0x05) from the upstream node after a node drops. Once the host processor detects an SRFMISSERR error, read the A2B_NODE register of the node flagging the error (the A2B_INTSRC register indicates which node is flagging the SRFMISSERR error). If the value in the A2B_NODE.LAST bit field is 0x1, the nodes downstream of this node are disconnected.

When a node drop is detected, choose to implement a full rediscovery or partial rediscovery depending on the system requirements.

If the main node resets during run time, there is no guarantee of an interrupt or indication to host processor, so, the host processor might be unaware of this condition for some time.

Note that when the main node resets due to a SYNC break, the main node returns to the power-up state with the MSTR bit maintained.

- If the MSTR bit was 1 during run-time, then when SYNC resumes, the node starts to lock the PLL again. There is a PLL LOCK interrupt (INTTYPE = 0xFF) upon PLL lock. If the host processor receives this interrupt during run-time, it can be an indication of a main node reset and subordinate nodes drop from the A^{2}B bus.
- If MSTR bit was 0, then SYNC resumption will have no effect, until the host processor detects the reset and sets the MSTR bit again.

Therefore, it is recommended to maintain the `A2B_CONTROL.MSTR` bit = 1 during run time, especially while applying NEWSTRCT.

- If the main node resets due to VIN drop or there is a hardware reset due to the assertion of the RST pin, then the `A2B_CONTROL` register is cleared. When a node moves to the power-up state, it remains idle until the host processor detects the reset and sets the MSTR bit again.

When a subordinate node is dropped, an I^{2}C read request fails with an I2CERR error.

9 A^{2}B System Debug

The A^{2}B transceivers provides debug features that can be used for A^{2}B bus link verification or for generating interrupts. These features are described in the following sections.

I^{2}S Loopback

I^{2}S loopback occurs inside the transceiver. Data driven to the DTX0 pad is sampled as A^{2}B receive data instead of the data on the DRX0 pin. Data driven to the DTX1 pad is sampled as A^{2}B receive data instead of the data on the DRX1 pin.

When I^{2}S loopback mode is enabled, there should be an equal number of TX and RX pins that are enabled. Program the value of the `A2B_I2SCFG.RXPINS` bit to match the value of the `A2B_I2SCFG.TXPINS` bit.

If the `A2B_SLOTFMT.UPSIZE` and `A2B_SLOTFMT.DNSIZE` bit field values are different, looped back data, which changes direction on the bus, is either truncated to a smaller bit width or zero-filled to a larger bit width.

When this mode is enabled, the program is responsible for ensuring that the data received from the A^{2}B bus and looped back through the serial blocks can be transmitted on the A^{2}B bus.

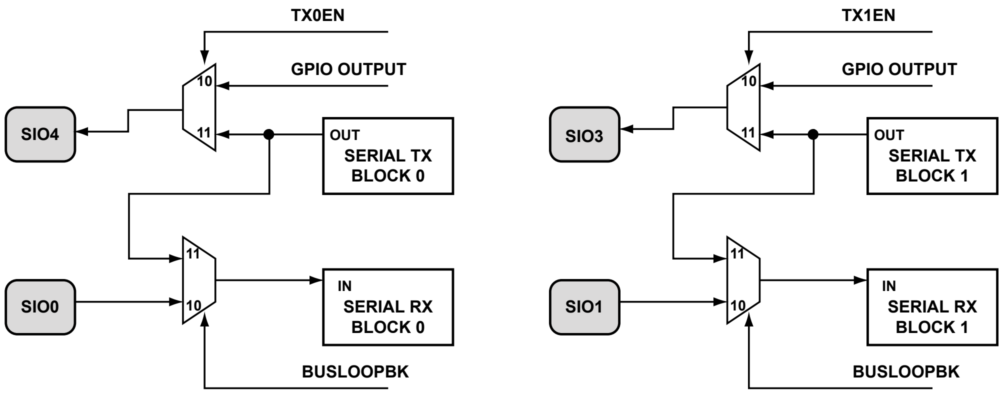


Figure 9-1: Serial TX Block to Serial RX Block

I^{2}S TDM Test Mode (I^{2}S Loopback)

Pattern generation and loopback test modes are provided for easy validation of I^{2}S TDM connectivity in main and subordinate nodes. The transmit pattern generator uses the default bit pattern `1011_0011_1000_1111__0000_1110_0011_0010` on all channels, where `1011` is the most significant nibble and `0010` is the least significant nibble.

Use the following procedure for I^{2}S TDM testing.

1. For main node to host link verification, set the `A2B_I2STEST.PATTRN2TX` bit in the main transceiver and verify that the TX interface with the default bit pattern matches the expected timing (possibly using a scope, logic analyzer, or other device).
2. For host to main node link verification. Set the `A2B_I2STEST.RX2LOOPBK` and `A2B_I2STEST.LOOPBK2TX` bits in the main transceiver, wait one cycle, and verify that the DTX data received at the host matches the DRX data sent from the previous frame.

   ADDITIONAL INFORMATION: The RX to TX loopback does not working correctly when the main node is also receiving TX data from the bus. The `A2B_DATCTL` register must be 0x00 while looping back from RX to TX.
3. For sub node to peripheral link verification, if a sub node is connected to a DAC (for example, to send to a speaker), set the `A2B_I2STEST.PATTRN2TX` bit in the sub node and verify the expected DTX timing.
4. For peripheral to sub node link verification, if a sub node has a peripheral that provides input signals over the I^{2}S/ TDM interface, set the `A2B_I2STEST.RX2LOOPBK` and `A2B_I2STEST.LOOPBK2TX` bits. Verify that the DTX interface matches the DRX interface with a one frame delay. Alternatively (without using the `A2B_I2STEST` register) check the RX data at the previously verified main node I^{2}S/TDM DTX interface.
5. System verification with bus loopback. Connect the SIO4/SIO3 pins with the SIO0/SIO1 pins in a sub node to generate a digital loopback. The default bit pattern can be verified at the main node DTX pins when the `A2B_I2STEST.PATTRN2TX` bit is set at the sub node.

   ADDITIONAL INFORMATION: If the `A2B_I2STEST.RX2LOOPBK` bit is cleared while the `A2B_I2STEST.LOOPBK2TX` bit is set, the last received frame is repeated on the TX pins. This behavior continues until the `A2B_I2STEST.RX2LOOPBK` bit is set or the `A2B_I2STEST.LOOPBK2TX` bit is cleared. If the `A2B_I2STEST.LOOPBK2TX` bit is enabled after reset, the default pattern is generated until the `A2B_I2STEST.RX2LOOPBK` bit is set.

   ADDITIONAL INFORMATION: The ***Frame Buffer*** figure shows the TX frame buffer that is used for loopback tests.

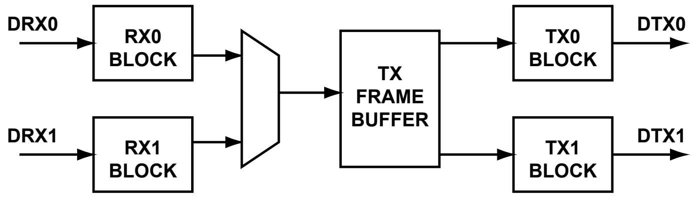


Figure 9-2: Frame Buffer

I^2S External Loopback

In external loopback, the value of an I^2S/TDM RX pin is driven directly onto an I^2S/TDM TX pin. This loopback can be applied even when the PLL is not locked. The I^2S external loopback functionality is limited when enabled. In mode 1, I^2S with one pin in each direction can be used on an A^2B main transceiver. In mode 2, no I^2S functionality is available. Unused SIO pins are available for alternate functions.

The *External Loopback Modes* table describes modes available for external loopback.

Table 9-1: External Loopback Modes

`A2B_I2STEST.EXTLOOPBK` Bit Field Setting	Mode
00	External loopback disabled
01	SIO0 (RX0) → SIO3 (TX1)
10	SIO0 (RX0) → SIO3 (TX1) SIO1 (RX1) → SIO4 (TX0)
11	Reserved

The *External Loopback Modes* figure provides a pictorial description of external loopback. In the figure:

- 1 - indicates data from the DTX0 pin of a SOC/peripheral to the A2B_ DRX0(SIO0) pin
- 2 - indicates the external loopback from the A2B_DRX0 (SIO0) pin to the A2B_DTX1(SIO3) pin
- 3 - indicates data from the A2B_DTX1(SIO3) pin to the DRX0 pin of the SOC/Peripheral

Similarly, data on DTX1 of SOC/peripheral can be looped back to the DRX1 pin of the SOC/peripheral using external loopback from the A2B_DRX1(SIO1) pin to the A2B_DTX0(SIO4) pin.

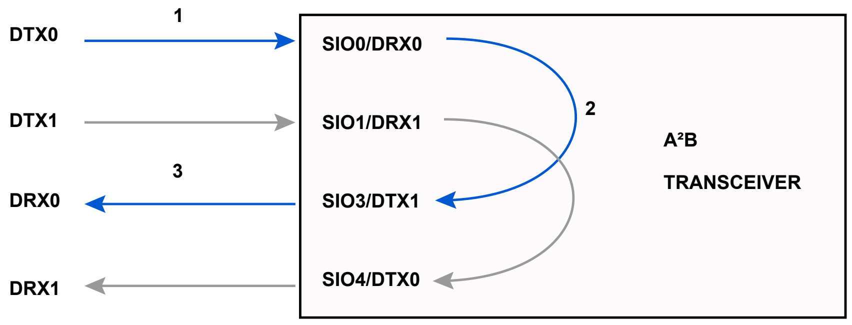


Figure 9-3: External Loopback

Raising Interrupts

For testing and debugging, the transceiver allows the generation of interrupts and bit errors using the raise `A2B_RAISE` register (register address = 0x54). The `A2B_RAISE` register allows the host to use software to generate an interrupt in any node in the system. The register must be written over the A^{2}B bus because writes to the register from the local I^{2}C/SPI port have no effect.

Mailbox and SPI interrupts, when raised, set the corresponding bit in `A2B_MBOX0STAT`/`A2B_MBOX1STAT` and `A2B_SPIINT` register.

NOTE: When writing to the `A2B_RAISE` register, an interrupt is raised only when the corresponding `A2B_INTMSK0`/`A2B_INTMSK1`/`A2B_INTMSK2` and `A2B_SPIMSK` register bits are set.

Generating Bit Errors

For testing and debugging, the transceiver allows the generation of interrupts and bit errors using the generate error `A2B_GENERR` register.

Generate Error (`A2B_GENERR`) Register

- *0x01 Generate Header Count Error* (`A2B_GENERR.GENHCERR`)

  1. When the main node generates the header count error:

     The main node changes the 2-bit CNT field in the SCF for one frame only. In the subsequent frame, it sends the correct CNT field.

     Because each sub node receives the SCF, all sub transceivers detect the (`A2B_INTPND0.HDCNTERR`) error.

  2. When a sub node generates the header count error:

     The sub node changes the 2-bit CNT field in the SRF. Generally, the sub node passes the received SRF as-is from a downstream sub node. In this case (because the sub node is receiving the write to the

A2B_GENERR command in the frame), it generates its own SRF for a single superframe, but with the wrong CNT field, as the command indicated.

Although the upstream sub nodes receive the SRF, the nodes do not check whether the CNT field is correct. The sub nodes only generate the A2B_INTPND0.HDCNTERR value upon checks of the SCF. Therefore, when the sub node generates this error, only the main node detects it.

- ***0x02 Generate Data Decoding Error*** (A2B_GENERR.GENDDERR)

  Generating a data decoding error requires a Manchester coding violation to be applied to data slots, not to the SCF and SRF fields.

  1. When the main node generates the data decoding error:

     The main node induces a Manchester encoding error on the first downstream data slot (slot 0 only). It does not inject the error on any other data slots. Since nodes report the data decode error on data slots that are consumed, only the sub nodes that consume slot 0 detect the error when the main node generates it. When a sub node passes (without consuming) the data downstream, it sends the same bit stream that it receives and does not detect the error.

  2. When a sub node generates the data decoding error:

     The sub node induces a Manchester encoding error on the first upstream data slot it contributes, not on any passed data slots. If the sub node contributes more than one upstream slot, it only induces the error on the first one. Sub nodes do not induce encoding errors on downstream data.

     Since data decode errors are reported on data slots which are consumed, only the upstream nodes that consume the first contributed upslot detect the error. If an upstream sub node or a main node does not consume the first contributed data slot, it does not detect the error.

- ***0x04 Generate CRC Error*** (A2B_GENERR.GENCRCERR)

  1. When the main node generates the CRC error:

     The main node induces the error in the CRC field of the SCF for one frame only. Because each sub node receives the SCF, all sub nodes detect the error in the CRC.

  2. When a sub node generates the CRC error:

     Sub nodes induce the error in the CRC field of the SRF for one frame only. Since all upstream sub nodes receive the SRF and check the CRC, all of them detect this error when any downstream sub node generates it. The sub nodes report the SRF CRC errors in the A2B_INTPND0.SRFCRCERR field, but these errors are not counted by the bit error counter. The main node detects the error as A2B_INTPND0.CRCERR and increments the bit error counter, when enabled.

- ***0x08 Generate Data Parity Error***, A2B_GENERR.GENDPERR

  1. When the main node generates the data parity error:

The main node induces the data parity error on the first downstream data slot (slot 0). It does not induce the error on other data slots. When the main node generates the data parity error, only the sub nodes that consume slot 0 detect it. sub nodes that do not consume slot 0 do not detect it.

2. When a sub node generates the data parity error:

   A sub node induces the data parity error on only the first upstream data slot it contributes. It does not induce the error in the downstream portion of the superframe. When a sub node generates the error, all of the upstream nodes that consume the first contributed slot detect it. If an upstream sub node or a main node does not consume the first contributed data slot, it does not detect it.

- *0x10 Generate Interrupt Frame CRC Error* (A2B_GENERR.GENICRCERR)

  1. The main node cannot generate the interrupt frame CRC error.
  2. When a sub node generates the interrupt frame CRC error, only the main node is able to detect it. Other upstream sub nodes do not check the CRC in the interrupt frame.

PRBS Test

Pseudo Random Binary sequence (PRBS) is used for error checking between nodes. When PRBS node-to-node check enabled, each node checks all incoming data bits and transmits the expected data to the next node. This feature allows for better determination of where bus errors occur.

The A2B_TESTMODE.PRBSUP and A2B_TESTMODE.PRBSDN bits are used to enable the use of pseudo-random data in the downstream and upstream data slots on the A^2B bus, respectively. PRBS sequence is generated using the Linear Feedback Shift Registers (LFSR). The taps and seeds are not programmable. Each node has same LFSR design to generate the PRBS sequence when enabled and each node generates the same PRBS data in each superframe. Therefore, in a PRBS test, the PRBS mode must be enabled in all nodes at the same time using a broadcast write to the A2B_TESTMODE register.

PRBS data transmission and checking is based on the location in the frame buffer. The frame buffer is filled with PRBS data in every superframe. The data is taken from the frame buffer to transmit over the bus based on the slot number. The frame buffer is indexed differently for downstream data versus upstream data, but it is the same 32 words in each superframe. Downstream data is checked in the last sub node and upstream data is checked in the main node based on the programming of slot registers. Data mismatches increment a 32-bit PRBS error counter (which can be read using the A2B_ERRCNT0 through A2B_ERRCNT3 registers).

Example

Consider a typical microphone system M – S0 (MIC) - S1 (MIC), in which each MIC sub node contributes two slots. The main node receives four slots.

Assume that PRBS logic generates the data 0x00, 0x01, 0x02, 0x03, 0x04 ... and so on. Each node generates the same PRBS sequence in every frame for transmission of contributed data and comparison of consumed data. The S1 sub node sends the data from frame buffer locations 0 and 1 upstream; S0 sub node sends the data from frame buffer locations 2 and 3. The main node knows it is receiving four upstream slots. It checks the slot data against the expected values for its frame buffer locations 0 through 3. The process repeats each superframe.

In frame 1, S1 sub node sends 0x00 and 0x01 data in its two upstream slots; the S0 sub node adds 0x02 and 0x03 data (depending on the contributed slot numbers) to the bus. The main node receive this data as 0x00, 0x01, 0x02 and 0x03. It compares the received stream with the internally generated PRBS sequence.

In frame 2, nodes generate data as 0x20, 0x21, 0x22, 0x23, 0x24 ... and so on. In this frame, the S1 sub node sends 0x20, 0x21 data in its two upstream slots; S0 sub node contributes 0x22 and 0x23 data (depending on the contributed slot numbers) to the bus. The main node receives this data as 0x20, 0x21, 0x22 and 0x23. It compares the received stream with the internally generated PRBS sequence. Any mismatch in received data causes the PRBS error count byte registers (0-3) to increment. The PRBS data mismatch errors do not generate any interrupt on the IRQ line. The PRBS error count byte registers can be analyzed at the end of the PRBS test. When the `A2B_TESTMODE.PRBSN2N` bit is set, each middle sub nodes checks all incoming data bits and transmits the expected data to the next-in-line node. This sequence allows for better determination of where bus errors occur.

NOTE: To run a PRBS test, it is not required to modify any of the existing A^{2}B nodes settings in an A^{2}B chain except for the registers required to run PRBS.

Enabling PRBS

The PRBS sequence internally generated by nodes in each frame should match with other nodes. Each node uses the frame data for the transmission of contributed data and comparison of consumed data. Therefore, enable the PRBS sequence in all nodes at the same time using a broadcast write to the `A2B_TESTMODE` register. Disable the downstream and upstream traffic while the enabling PRBS sequence to start PRBS data transmission and reception simultaneously.

Use the following procedure to enable the PRBS test in a system.

1. Read the PRBS error count byte registers (`A2B_ERRCNT0`- `A2B_ERRCNT3`) (RegAdr: 0x21 to 0x24) of all nodes.
2. Clear the `A2B_DATCTL` register (RegAdr: 0x11) in main node transceiver to disable the downstream and upstream traffic on the bus.
3. Apply `A2B_CONTROL.NEWSTRCT` (RegAdr: 0x12) = 0x81.
4. Set the `A2B_NODEADR.BRCST` bit (=1) to enable broadcast mode in the main transceiver (RegAdr: 0x01).
5. Enable the PRBS sequence. Write to the `A2B_TESTMODE` register (RegAdr: 0x20) of any sub node transceiver.
6. Clear `A2B_NODEADR.BRCST` bit (=0) in the main transceiver (RegAdr: 0x01).
7. Set the `A2B_DATCTL` register (RegAdr: 0x11) in main node transceiver to enable downstream and upstream traffic on bus.
8. Apply `A2B_CONTROL.NEWSTRCT` (RegAdr: 0x12) = 0x81.

ADDITIONAL INFORMATION: Note: If the PRBS sequence is repeatedly being enabled and disabled during system run time, disable the `A2B_TESTMODE` register before enabling to make sure PRBS sequence was not running already.

Disabling PRBS

Use the following procedure to disable the PRBS test in a system. While disabling the PRBS sequence, the `A2B_TESTMODE` register can be written in broadcast mode. Disabling upstream or downstream may not be needed.

1. Set the `A2B_NODEADR.BRCST` bit (RegAdr: 0x01) of the main node transceiver .
2. Disable the PRBS sequence. Write to the `A2B_TESTMODE` register (RegAdr: 0x20) of any sub node transceiver.
3. Clear the `A2B_NODEADR.BRCST` bit (RegAdr: 0x01) of the main node transceiver.

Data-Only and Power-Only Bus Operation

The A^{2}B bus can be operated without closing the NMOS switch to send a DC bias downstream. This requires that the `A2B_CONTROL.SWBYP` bit is set instead of the `A2B_SWCTL.ENSW` bit during discovery.

Conversely, the `A2B_SWCTL.DISNXT` bit allows a DC bias to be sent downstream without the presence of data. This setting should be applied at the same time as the write to set the `A2B_SWCTL.ENSW` bit during discovery. These modes are used primarily for debug purposes.

Standby Mode

Standby is a low power mode in which only a minimal SCF exists to keep all the subordinate node transceivers synchronized. There is no downstream and upstream data traffic on the A^{2}B bus and no SRF field. The A^{2}B bus can be put in standby mode by setting the `A2B_DATCTL.STANDBY` bit followed by the `A2B_CONTROL.NEWSTRCT` bit in the main node transceiver. The SCF in standby mode is 19 bits long, instead of 64 bits. This keeps the A^{2}B bus power in the lowest power state while maintaining clock synchronization between nodes (PLL of all nodes are in the locked state).

The bus activity level in standby mode is:

- Downstream activity level = 19 ÷ 1024 = 1.9%
- Upstream activity level = 0%

The GPIO settings retain their values while the transceiver is in standby mode.

Standby mode is useful to reduce the device currents of all A^{2}B transceivers in the A^{2}B chain.

- PLLVDD current does not change in standby mode; it is same as in normal mode. The PLL is in locked state in the standby mode.
- DVDD current reduces in standby mode.
- TRXVDD current significantly reduces in standby mode because the A^{2}B bus is active only for 1.9% of the total superframe time.
- IOVDD current depends in the number of IO pins (especially output pins) active during standby mode. The BCLK, SYNC and DTX pins can stop toggling (by configuring the `A2B_I2SCFG` register) before entering standby mode.
- VIN current is reduced significantly in standby mode because most of the power domain are supplied from internally generated VOUT1 and VOUT2 voltage regulators.

Refer to the data sheet for details on current consumptions by different power domains during standby mode.

Standby mode can be exited by clearing the `A2B_DATCTL.STANDBY` bit. System traffic can be resumed without the need for rediscovery of the A^{2}B node transceiver. As soon as the system comes out of standby mode, the A^{2}B main node transceiver generates a standby done interrupt (INTTYPE = 0xFE).

Bus Monitor Support

Bus Monitor Support is only available in single pair wire systems.

Bus monitor mode enables the transceiver to act as a passive audio bus monitor, also referred to as a *sniffer*. The A^{2}B test equipment uses this mode. A node can be configured into Bus Monitor mode by programming the `A2B_BMMCFG` register.

A bus monitor is passive in the system; it does not respond to bus synchronization control frames (SCFs) or contribute any data to the bus. It only uses the A-side transceiver while the B-side transceiver is deactivated. When in bus monitor mode, the transceiver synchronizes itself to SCFs and may snoop SCF control writes to configure its bus interface to match the downstream node being monitored. The A^{2}B bus monitor transceiver uses its I^{2}S/TDM port to transmit A^{2}B bus traffic to a protocol analyzer circuit.

The ***Bus Monitor Behavior*** figure shows a bus monitor node inserted between subordinate nodes in an A^{2}B network.

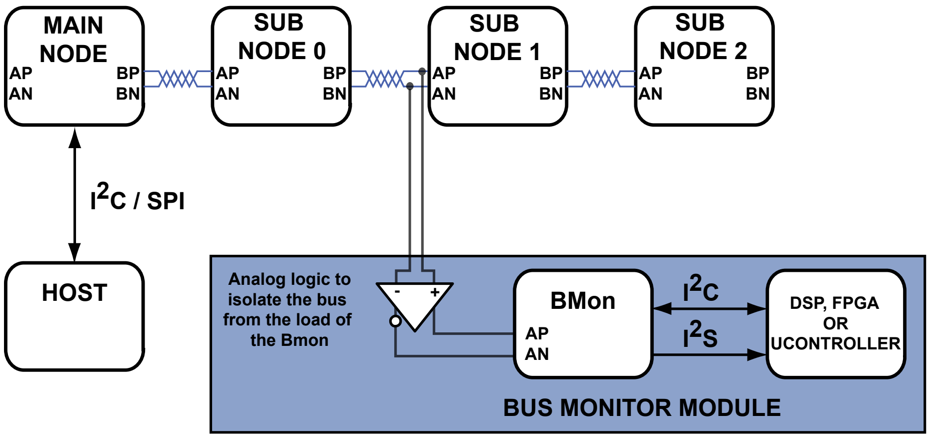


Figure 9-4: Bus Monitor Behavior

Only the host processor can permit bus monitors on A^{2}B bus segments to monitor the synchronous data content. To permit synchronous data monitoring, the host must set the `A2B_DATCTL.ENDSNIFF` bit in the main transceiver. This configuration triggers an A^{2}B bus broadcast of information to the attached bus monitor devices.

A bus monitor node behaves as follows:

1. The B-Side (downstream) transceiver is disabled.
2. The A-Side (upstream) transceiver is enabled to receive only (not to transmit).
3. SRF generation is disabled.
4. The I^{2}S/TDM interface is configured for 32-bit data width:
   - Downstream SCFs are transmitted on the DTX0 pin
   - Upstream SRFs are transmitted on the DTX1 pin
   - Data slot bits can only stream out of the DTXn pins if the A^{2}B bus main node is programmed to enable this feature
     - Downstream slots are streamed out on the DTX0 pin
     - Upstream slots are streamed out on the DTX1 pin
     - If there are more data slots on the A^{2}B bus than there are available I^{2}S/TDM channels, then a programmable offset determines which data slots to monitor on the I^{2}S/TDM channels

NOTE: When the bus monitor receiver is disabled, an external switch must be used to control the LVDS traffic going to the of the transceiver in bus monitor mode.

10 Register Summary

The memory-mapped register (MMR) space of the A^{2}B transceiver features a pagination scheme, as controlled by the `A2B_MMRPAGE` register. When an access is made to the MMR space, the transceiver extracts the supplied 8-bit address and concatenates the content of the `A2B_MMRPAGE` register to create a 16-bit address, where:

- the value in the `A2B_MMRPAGE` register is the upper byte, and
- the 8-bit address furnished in the access is the lower byte

NOTE: In the Register Summary table and the supporting register drawings (and throughout this book), 8-bit hexadecimal addresses assume that the upper byte is 0x00, thereby defining registers on page 0. When an address width is greater than 8-bit, the zero-filled upper byte defines the page.

Direct accesses to the MMR space are initiated by an externally-connected I^{2}C or SPI host on the local node, whereas *remote accesses* come to a subordinate node transceiver over the A^{2}B bus from the initiating node as part of the I^{2}C and SPI over distance protocols. The described concatenation scheme applies to both access types. Remote accesses are forwarded over the bus by the protocol itself using the 8-bit address that was furnished in the bus access on the initiating node. On the target subordinate transceiver, the forwarded address is concatenated with the content of the `A2B_MMRPAGE` register of the subordinate transceiver to form the 16-bit address used to access the MMR space of the subordinate transceiver.

NOTE: Though described as having an address of 0xE0 (page 0), the `A2B_MMRPAGE` register itself can be programmed at any time, regardless of what the current value of the `A2B_MMRPAGE` register is.

REMEMBER: The originator of any direct or remote register access must ensure that the `A2B_MMRPAGE` register on the subordinate transceiver is properly programmed at all times. Carefully manage the manipulation of the `A2B_MMRPAGE` register when accessing MMR space.

The ***Register Summary*** table provides the map of the AD2437 registers and bits.

Table 10-1: Register Summary

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x00	CHIP	Reserved	CHIPADR							0x50	R/W
0x01	NODEADR	BRCST	Reserved	PERI	Reserved	NODE				0x00	R/W
0x02	VENDOR	VENDOR								0xAD	R/NW

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x03	PRODUCT	PRODUCT								0x37	R/NW
0x04	VERSION	VERSION								0x00	R/NW
0x05	CAPABILITY	Reserved						SPIAVAIL	I2CAVAIL		R/NW
0x09	SWCTL	Reserved	DISNXT	MODE		DIAGMODE	Reserved	CFG_DET_ OV	ENSW	0x00	R/W
0x0A	BCDNSLOTS	Reserved		BCDNSLOTS						0x00	R/W
0x0B	LDNSLOTS	DNMASKEN	Reserved	LDNSLOTS						0x00	R/W
0x0C	LUPSLOTS	Reserved		LUPSLOTS						0x00	R/W
0x0D	DNSLOTS	Reserved		DNSLOTS						0x00	R/W
0x0E	UPSLOTS	Reserved		UPSLOTS						0x00	R/W
0x0F	RESPCYCS	RESPCYCS								0x40	R/W
0x10	SLOTFMT	UPFMT	UPSIZE			DNFMT	DNSIZE			0x00	R/W
0x11	DATCTL	STANDBY	Reserved	ENDSNIFF	Reserved			UPS	DNS	0x00	R/W
0x12	CONTROL	MSTR	Reserved	I2SMSINV	XCVRBINV	SWBYP	SOFTRST	ENDDSC	NEWSTRCT	0x00	R/W
0x13	DISCVRY	DRESPCYC								0x00	R/W
0x14	SWSTAT	FAULT_ NLOC	FAULT_CODE			Reserved		FAULT	FIN	0x00	R/NW
0x15	INTSTAT	Reserved							IRQ	0x00	R/NW
0x16	INTSRC	MSTINT	SLVINT	Reserved		INODE				0x00	R/NW
0x17	INTTYPE	TYPE								0x00	R/NW
0x18	INTPND0	SRFCRCERR	SRFMISSERR	BECOVF	PWRERR	DPERR	CRCERR	DDERR	HDCNTERR	0x00	R/W
0x19	INTPND1	IO7PND	IO6PND	IO5PND	IO4PND	IO3PND	IO2PND	IO1PND	IO0PND	0x00	R/W
0x1A	INTPND2	Reserved				SLVIRQ	ICRCERR	I2CERR	DSCDONE	0x00	R/W
0x1B	INTMSK0	SRFCRCEIEN	SRFMISSEIEN	BECIEN	PWREIEN	DPEIEN	CRCEIEN	DDEIEN	HCEIEN	0x00	R/W
0x1C	INTMSK1	IO7IRQEN	IO6IRQEN	IO5IRQEN	IO4IRQEN	IO3IRQEN	IO2IRQEN	IO1IRQEN	IO0IRQEN	0x00	R/W
0x1D	INTMSK2	Reserved				SLVIRQEN	ICRCEIEN	I2CEIEN	DSCDIEN	0x00	R/W
0x1E	BECCTL	THRESHLD			ENICRC	ENDP	ENCRC	ENDD	ENHDCNT	0x00	R/W
0x1F	BECNT	BECNT								0x00	R/W
0x20	TESTMODE	Reserved					PRBSN2N	PRBSDN	PRBSUP	0x00	R/W
0x21	ERRCNT0	ERRCNT0[7:0]								0x00	R/NW
0x22	ERRCNT1	ERRCNT1[15:8]								0x00	R/NW
0x23	ERRCNT2	ERRCNT2[23:16]								0x00	R/NW
0x24	ERRCNT3	ERRCNT3[31:24]								0x00	R/NW
0x29	NODE	LAST	NLAST	DISCVD	Reserved	NUMBER				0x80	R/NW
0x2B	DISCSTAT	DSCACT	Reserved			DNODE				0x00	R/NW
0x3E	LINTTYPE	LTYPE								0x00	R/NW
0x3F	I2CCFG	DISI2C	Reserved			FMPLUS	FRAMERATE	EACK	DATARATE	0x00	R/W
0x41	I2SGCFG	INV	EARLY	ALT	TDMSS	SYNCDIS	TDMMODE			0x00	R/W
0x42	I2SCFG	RXBCLKINV	RX2PINS			TXBCLKINV	TXPINS			0x00	R/W
0x43	I2SRATE	SHARE	REDUCE	BCLKRATE			I2SRATE			0x00	R/W

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x44	I2STXOFFSET	TSBEFORE	TSAFTER	TXOFFSET						0x00	R/W
0x46	SYNCOFFSET	SYNCOFFSET								0x00	R/W
0x47	PDMCTL	Reserved	PDMRATE		HPFEN	PDM1SLOTS	PDM1EN	PDM0SLOTS	PDM0EN	0x00	R/W
0x48	ERRMGMT	Reserved					ERRSLOT	ERRSIG	ERRLSB	0x00	R/W
0x4A	GPIODAT	IO7DAT	IO6DAT	IO5DAT	IO4DAT	IO3DAT	IO2DAT	IO1DAT	IO0DAT	0x00	R/W
0x4B	GPIODATSET	IO7DSET	IO6DSET	IO5DSET	IO4DSET	IO3DSET	IO2DSET	IO1DSET	IO0DSET	0x00	R/W
0x4C	GPIODATCLR	IO7DCLR	IO6DCLR	IO5DCLR	IO4DCLR	IO3DCLR	IO2DCLR	IO1DCLR	IO0DCLR	0x00	R/W
0x4D	GPIOOEN	IO7OEN	IO6OEN	IO5OEN	IO4OEN	IO3OEN	IO2OEN	IO1OEN	IO0OEN	0x00	R/W
0x4E	GPIOIEN	IO7IEN	IO6IEN	IO5IEN	IO4IEN	IO3IEN	IO2IEN	IO1IEN	IO0IEN	0x00	R/W
0x4F	GPIOIN	IO7IN	IO6IN	IO5IN	IO4IN	IO3IN	IO2IN	IO1IN	IO0IN	0x00	R/NW
0x50	PINTEN	IO7IE	IO6IE	IO5IE	IO4IE	IO3IE	IO2IE	IO1IE	IO0IE	0x00	R/W
0x51	PINTINV	IO7INV	IO6INV	IO5INV	IO4INV	IO3INV	IO2INV	IO1INV	IO0INV	0x00	R/W
0x52	PINCFG	GPIOSEL		IRQTS	IRQINV	Reserved			DRVSTR	0x01	R/W
0x53	I2STEST	EXTLOOPBK		Reserved	BUSLOOPBK	SELRX1	RX2LOOPBK	LOOPBK2TX	PATTRN2TX	0x00	R/W
0x54	RAISE	RTYPE								0x00	R/W
0x55	GENERR	Reserved			GENICRCERR	GENDPERR	GENCRCERR	GENDDERR	GENHCERR	0x00	R/W
0x56	I2SRRATE	RBUS	Reserved	RRDIV						0x01	R/W
0x57	I2SRRCTL	Reserved		STRBDIR	ENSTRB	Reserved		ENXBIT	ENVLSB	0x00	R/W
0x58	I2SRRSOFFS	Reserved						RRSOFFSET		0x00	R/W
0x59	CLK1CFG	CLK1EN	CLK1INV	CLK1PDIV	Reserved	CLK1DIV				0x00	R/W
0x5A	CLK2CFG	CLK2EN	CLK2INV	CLK2PDIV	Reserved	CLK2DIV				0x00	R/W
0x5B	BMMCFG	Reserved				BMMDT	BMMNDSC	BMMRXEN	BMMEN	0x00	R/W
0x5C	SUSCFG	Reserved		SUSDIS	SUSOE	Reserved	SUSSEL			0x00	R/W
0x5D	PDMCTL2	HPFCORNER		PDMINVCLK	PDMALTCLK	PDM1FFRST	PDM0FFRST	PDMDEST		0x00	R/W
0x60	UPMASK0	RXUPSLOT07	RXUPSLOT06	RXUPSLOT05	RXUPSLOT04	RXUPSLOT03	RXUPSLOT02	RXUPSLOT01	RXUPSLOT00	0x00	R/W
0x61	UPMASK1	RXUPSLOT15	RXUPSLOT14	RXUPSLOT13	RXUPSLOT12	RXUPSLOT11	RXUPSLOT10	RXUPSLOT09	RXUPSLOT08	0x00	R/W
0x62	UPMASK2	RXUPSLOT23	RXUPSLOT22	RXUPSLOT21	RXUPSLOT20	RXUPSLOT19	RXUPSLOT18	RXUPSLOT17	RXUPSLOT16	0x00	R/W
0x63	UPMASK3	RXUPSLOT31	RXUPSLOT30	RXUPSLOT29	RXUPSLOT28	RXUPSLOT27	RXUPSLOT26	RXUPSLOT25	RXUPSLOT24	0x00	R/W
0x64	UPOFFSET	Reserved			UPOFFSET					0x00	R/W
0x65	DNMASK0	RXDNSLOT07	RXDNSLOT06	RXDNSLOT05	RXDNSLOT04	RXDNSLOT03	RXDNSLOT02	RXDNSLOT01	RXDNSLOT00	0x00	R/W
0x66	DNMASK1	RXDNSLOT15	RXDNSLOT14	RXDNSLOT13	RXDNSLOT12	RXDNSLOT11	RXDNSLOT10	RXDNSLOT09	RXDNSLOT08	0x00	R/W
0x67	DNMASK2	RXDNSLOT23	RXDNSLOT22	RXDNSLOT21	RXDNSLOT20	RXDNSLOT19	RXDNSLOT18	RXDNSLOT17	RXDNSLOT16	0x00	R/W
0x68	DNMASK3	RXDNSLOT31	RXDNSLOT30	RXDNSLOT29	RXDNSLOT28	RXDNSLOT27	RXDNSLOT26	RXDNSLOT25	RXDNSLOT24	0x00	R/W
0x69	DNOFFSET	Reserved			DNOFFSET					0x00	R/W
0x6A	CHIPID0	CHIPID[7:0]								0xXX	R/NW
0x6B	CHIPID1	CHIPID[15:8]								0xXX	R/NW
0x6C	CHIPID2	CHIPID[23:16]								0xXX	R/NW
0x6D	CHIPID3	CHIPID[31:24]								0xXX	R/NW
0x6E	CHIPID4	CHIPID[39:32]								0xXX	R/NW
0x6F	CHIPID5	CHIPID[47:40]								0xXX	R/NW

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x7C	DTCFG	Reserved					DTLAST	DTFRST	DTEN	0x00	R/W
0x7D	DTSLOTS	DTUPSLOTS				DTDNSLOTS				0x00	R/W
0x7E	DTDNOFFS	Reserved			DTDNOFFS					0x00	R/W
0x7F	DTUPOFFS	Reserved			DTUPOFFS					0x00	R/W
0x80	GPIODEN	IOD7EN	IOD6EN	IOD5EN	IOD4EN	IOD3EN	IOD2EN	IOD1EN	IOD0EN	0x00	R/W
0x81	GPIOD0MSK	IOD0MSK								0x00	R/W
0x82	GPIOD1MSK	IOD1MSK								0x00	R/W
0x83	GPIOD2MSK	IOD2MSK								0x00	R/W
0x84	GPIOD3MSK	IOD3MSK								0x00	R/W
0x85	GPIOD4MSK	IOD4MSK								0x00	R/W
0x86	GPIOD5MSK	IOD5MSK								0x00	R/W
0x87	GPIOD6MSK	IOD6MSK								0x00	R/W
0x88	GPIOD7MSK	IOD7MSK								0x00	R/W
0x89	GPIODDAT	IOD7DAT	IOD6DAT	IOD5DAT	IOD4DAT	IOD3DAT	IOD2DAT	IOD1DAT	IOD0DAT	0x00	R/NW
0x8A	GPIODINV	IOD7INV	IOD6INV	IOD5INV	IOD4INV	IOD3INV	IOD2INV	IOD1INV	IOD0INV	0x00	R/W
0x90	MBOX0CTL	Reserved		MB0LEN		MB0FIEN	MB0EIEN	MB0DIR	MB0EN	0x00	R/W
0x91	MBOX0STAT	Reserved		MB0EIRQ	MB0FIRQ	Reserved		MB0EMPTY	MB0FULL	0x02	R/NW
0x92	MBOX0B0	MBOX0[7:0]								0x00	R/W
0x93	MBOX0B1	MBOX0[15:8]								0x00	R/W
0x94	MBOX0B2	MBOX0[23:16]								0x00	R/W
0x95	MBOX0B3	MBOX0[31:24]								0x00	R/W
0x96	MBOX1CTL	Reserved		MB1LEN		MB1FIEN	MB1EIEN	MB1DIR	MB1EN	0x02	R/W
0x97	MBOX1STAT	Reserved		MB1EIRQ	MB1FIRQ	Reserved		MB1EMPTY	MB1FULL	0x02	R/NW
0x98	MBOX1B0	MBOX1[7:0]								0x00	R/W
0x99	MBOX1B1	MBOX1[15:8]								0x00	R/W
0x9A	MBOX1B2	MBOX1[23:16]								0x00	R/W
0x9B	MBOX1B3	MBOX1[31:24]								0x00	R/W
0xA0	SWCTL2	Reserved		CAP_DLY		Reserved	HPSW_CFG			0x00	R/W
0xA5	SWSTAT2	LVI_MODE	Reserved	HPSW_CFG_DET			Reserved	HS_ILIM	LS_ILIM	0x00	R/W
0xAF	SPIDTLCMD	LASTCMD								0x00	R/NW
0xB0	SPICFG	SPIFDSS		ENFDCS	SPI_CPOL	SPI_CPHA	TNLOWNER	SPIMODE		0x00	R/W
0xB1	SPISTAT	DTBADPKT	DTABORT	DTINVALID	Reserved			DTACTIVE	SPIBUSY	0x00	R/W
0xB2	SPICKDIV	Reserved		CKDIV						0x00	R/W
0xB3	SPIFDSIZE	FDSIZE								0x00	R/W
0xB4	SPIFDTARG	SSEL		MnS	Reserved	NODE				0x00	R/W
0xB5	SPIPINCFG	Reserved	SPIMSS2EN	SPIMSS1EN	SPIMSS0EN	SPIGPIOEN	SPIGPIOSEL			0x00	R/W
0xB6	SPIINT	Reserved	FIFOUNF	FIFOOVF	BADCMD	SPIDTERR	SPII2CERR	SPIREGERR	SPIDONE	0x00	R/W
0xB7	SPIMSK	Reserved	FIFOUIEN	FIFOOIEN	BADCMDIEN	SPIDTIEN	SPII2CIEN	SPIREGIEN	SPIDIEN	0x00	R/W
0xB8	RXMASK0	RXMASK[7:0]								0xFF	R/W
0xB9	RXMASK1	RXMASK[15:8]								0xFF	R/W

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0xBA	RXMASK2	RXMASK[23:16]								0xFF	R/W
0xBB	RXMASK3	RXMASK[31:24]								0xFF	R/W
0xBC	RXMASK4	RXMASK[39:32]								0xFF	R/W
0xBD	RXMASK5	RXMASK[47:40]								0xFF	R/W
0xBE	RXMASK6	RXMASK[55:48]								0xFF	R/W
0xBF	RXMASK7	RXMASK[63:56]								0xFF	R/W
0xC0	TXXBAR0	Reserved			LOC0					0x00	R/W
0xC1	TXXBAR1	Reserved			LOC1					0x01	R/W
0xC2	TXXBAR2	Reserved			LOC2					0x02	R/W
0xC3	TXXBAR3	Reserved			LOC3					0x03	R/W
0xC4	TXXBAR4	Reserved			LOC4					0x04	R/W
0xC5	TXXBAR5	Reserved			LOC5					0x05	R/W
0xC6	TXXBAR6	Reserved			LOC6					0x06	R/W
0xC7	TXXBAR7	Reserved			LOC7					0x07	R/W
0xC8	TXXBAR8	Reserved			LOC8					0x08	R/W
0xC9	TXXBAR9	Reserved			LOC9					0x09	R/W
0xCA	TXXBAR10	Reserved			LOC10					0x0A	R/W
0xCB	TXXBAR11	Reserved			LOC11					0x0B	R/W
0xCC	TXXBAR12	Reserved			LOC12					0x0C	R/W
0xCD	TXXBAR13	Reserved			LOC13					0x0D	R/W
0xCE	TXXBAR14	Reserved			LOC14					0x0E	R/W
0xCF	TXXBAR15	Reserved			LOC15					0x0F	R/W
0xD0	TXXBAR16	Reserved			LOC16					0x10	R/W
0xD1	TXXBAR17	Reserved			LOC17					0x11	R/W
0xD2	TXXBAR18	Reserved			LOC18					0x12	R/W
0xD3	TXXBAR19	Reserved			LOC19					0x13	R/W
0xD4	TXXBAR20	Reserved			LOC20					0x14	R/W
0xD5	TXXBAR21	Reserved			LOC21					0x15	R/W
0xD6	TXXBAR22	Reserved			LOC22					0x16	R/W
0xD7	TXXBAR23	Reserved			LOC23					0x17	R/W
0xD8	TXXBAR24	Reserved			LOC24					0x18	R/W
0xD9	TXXBAR25	Reserved			LOC25					0x19	R/W
0xDA	TXXBAR26	Reserved			LOC26					0x1A	R/W
0xDB	TXXBAR27	Reserved			LOC27					0x1B	R/W
0xDC	TXXBAR28	Reserved			LOC28					0x1C	R/W
0xDD	TXXBAR29	Reserved			LOC29					0x1D	R/W
0xDE	TXXBAR30	Reserved			LOC30					0x1E	R/W
0xDF	TXXBAR31	Reserved			LOC31					0x1F	R/W
0xE0	MMRPAGE	PAGE								0x00	R/W
0x100	VMTR VEN	Reserved	VLTG							0x00	R/W

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x101	VMTR INTEN	Reserved	VLTG							0x00	R/W
0x102	VMTR MXSTAT	Reserved	MXERR60							0x00	R/W
0x103	VMTR MNSTAT	Reserved	MNERR60							0x00	R/W
0x120	VMTR VLTG0	VLTG								0x00	R/NW
0x121	VMTR VMAX0	VMAX								0xFF	R/W
0x122	VMTR VMIN0	VMIN								0x00	R/W
0x123	VMTR VLTG1	VLTG								0x00	R/NW
0x124	VMTR VMAX1	VMAX								0xFF	R/W
0x125	VMTR VMIN1	VMIN								0x00	R/W
0x126	VMTR VLTG2	VLTG								0x00	R/NW
0x127	VMTR VMAX2	VMAX								0xFF	R/W
0x128	VMTR VMIN2	VMIN								0x00	R/W
0x129	VMTR VLTG3	VLTG								0x00	R/NW
0x12A	VMTR VMAX3	VMAX								0xFF	R/W
0x12B	VMTR VMIN3	VMIN								0x00	R/W
0x12C	VMTR VLTG4	VLTG								0x00	R/NW
0x12D	VMTR VMAX4	VMAX								0xFF	R/W
0x12E	VMTR VMIN4	VMIN								0x00	R/W
0x12F	VMTR VLTG5	VLTG								0x00	R/NW
0x130	VMTR VMAX5	VMAX								0xFF	R/W
0x131	VMTR VMIN5	VMIN								0x00	R/W
0x132	VMTR VLTG6	VLTG								0x00	R/NW
0x133	VMTR VMAX6	VMAX								0xFF	R/W
0x134	VMTR VMIN6	VMIN								0x00	R/W
0x140	PWMCFG	Reserved		PWMORAND	PWMPRAND	PWMOEEN	PWM3EN	PWM2EN	PWM1EN	0x00	R/W
0x141	PWMFREQ	PWMOFREQ				PWMPFREQ				0x00	R/W
0x142	PWMBLINK1	Reserved	PWM2BLINK			Reserved	PWM1BLINK			0x00	R/W

Table 10-1: Register Summary (Continued)

Reg. Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x143	PWMBLINK2	Reserved	PWMOEBLINK			Reserved	PWM3BLINK			0x00	R/W
0x148	PWM1VALL	PWM1VAL[7:0]								0x00	R/W
0x149	PWM1VALH	PWM1VAL[15:8]								0x00	R/W
0x14A	PWM2VALL	PWM2VAL[7:0]								0x00	R/W
0x14B	PWM2VALH	PWM2VAL[15:8]								0x00	R/W
0x14C	PWM3VALL	PWM3VAL[7:0]								0x00	R/W
0x14D	PWM3VALH	PWM3VAL[15:8]								0x00	R/W
0x14E	PWMOEVALL	PWMOEVAL[7:0]								0x00	R/W
0x14F	PWMOEVALH	PWMOEVAL[15:8]								0x00	R/W
0x1E0	MMRPAGE1	PAGE								0x00	R/W

11 AD2437 A2B Register Descriptions

The transceiver (A2B) contains the following registers.

Table 11-1: AD2437 A2B Register List

Name	Description
A2B_CHIP	I2C Chip Address Register (Sub Only)
A2B_NODEADR	Node Address Register (Main Only)
A2B_VENDOR	Vendor ID Register
A2B_PRODUCT	Product ID Register
A2B_VERSION	Version ID Register
A2B_CAPABILITY	Capability ID Register
A2B_SWCTL	Switch Control Register
A2B_BCDNSLOTS	Broadcast Downstream Slots Register (Sub Only)
A2B_LDNSLOTS	Local Downstream Slots Register (Sub Only)
A2B_LUPSLOTS	Local Upstream Slots Register (Sub Only)
A2B_DNSLOTS	Downstream Slots Register
A2B_UPSLOTS	Upstream Slots Register
A2B_RESPCYCS	Response Cycles Register
A2B_SLOTFMT	Slot Format Register (Main Only, Auto-Broadcast)
A2B_DATCTL	Data Control Register (Main Only, Auto-Broadcast)
A2B_CONTROL	Control Register
A2B_DISCVRY	Discovery Register (Main Only)
A2B_SWSTAT	Switch Status Register
A2B_INTSTAT	Interrupt Status Register
A2B_INTSRC	Interrupt Source Register (Main Only)
A2B_INTTYPE	Interrupt Type Register (Main Only)
A2B_INTPND0	Interrupt Pending 0 Register

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_INTPND1	Interrupt Pending 1 Register
A2B_INTPND2	Interrupt Pending 2 Register (Main Only)
A2B_INTMSK0	Interrupt Mask 0 Register
A2B_INTMSK1	Interrupt Mask 1 Register
A2B_INTMSK2	Interrupt Mask 2 Register (Main Only)
A2B_BECCTL	Bit Error Count Control Register
A2B_BECNT	Bit Error Count Register
A2B_TESTMODE	Testmode Register
A2B_ERRCNT0	PRBS Error Count Byte 0 Register
A2B_ERRCNT1	PRBS Error Count Byte 1 Register
A2B_ERRCNT2	PRBS Error Count Byte 2 Register
A2B_ERRCNT3	PRBS Error Count Byte 3 Register
A2B_NODE	Node Register
A2B_DISCSTAT	Discovery Status Register (Main Only)
A2B_LINTTYPE	Local Interrupt Type (Sub Only)
A2B_I2CCFG	I2C Configuration Register
A2B_I2SGCFG	I2S Global Configuration Register
A2B_I2SCFG	I2S Configuration Register
A2B_I2SRATE	I2S Rate Register (Sub Only)
A2B_I2STXOFFSET	I2S Transmit Data Offset Register (Main Only)
A2B_SYNCOFFSET	SYNC Offset Register (Sub Only)
A2B_PDMCTL	PDM Control Register
A2B_ERRMGMT	Error Management Register
A2B_GPIODAT	GPIO Output Data Register
A2B_GPIODATSET	GPIO Output Data Set Register
A2B_GPIODATCLR	GPIO Output Data Clear Register
A2B_GPIOOEN	GPIO Output Enable Register
A2B_GPIOIEN	GPIO Input Enable Register
A2B_GPIOIN	GPIO Input Value Register
A2B_PINTEN	Pin Interrupt Enable Register
A2B_PINTINV	Pin Interrupt Invert Register

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_PINCFG	Pin Configuration Register
A2B_I2STEST	I2S Test Register
A2B_RAISE	Raise Interrupt Register
A2B_GENERR	Generate Bus Error
A2B_I2SRRATE	I2S Reduced Rate Register (Main Only, Auto-Broadcast)
A2B_I2SRRCTL	I2S Reduced Rate Control Register
A2B_I2SRRSOFFS	I2S Reduced Rate SYNC Offset Register (Sub Only)
A2B_CLK1CFG	CLKOUT1 Configuration Register
A2B_CLK2CFG	CLKOUT2 Configuration Register
A2B_BMMCFG	Bus Monitor Mode Configuration Register
A2B_SUSCFG	Sustain Configuration Register (Sub Only)
A2B_PDMCTL2	PDM Control 2 Register
A2B_UPMASK0	Upstream Data RX Mask 0 Register (Sub Only)
A2B_UPMASK1	Upstream Data RX Mask 1 Register (Sub Only)
A2B_UPMASK2	Upstream Data RX Mask 2 Register (Sub Only)
A2B_UPMASK3	Upstream Data RX Mask 3 Register (Sub Only)
A2B_UPOFFSET	Local Upstream Channel Offset Register (Sub Only)
A2B_DNMASK0	Downstream Data RX Mask 0 Register (Sub Only)
A2B_DNMASK1	Downstream Data RX Mask 1 Register (Sub Only)
A2B_DNMASK2	Downstream Data RX Mask 2 Register (Sub Only)
A2B_DNMASK3	Downstream Data RX Mask 3 Register (Sub Only)
A2B_DNOFFSET	Local Downstream Channel Offset Register (Sub Only)
A2B_CHIPID0	Chip ID Register 0
A2B_CHIPID1	Chip ID Register 1
A2B_CHIPID2	Chip ID Register 2
A2B_CHIPID3	Chip ID Register 3
A2B_CHIPID4	Chip ID Register 4
A2B_CHIPID5	Chip ID Register 5
A2B_DTCFG	Data Tunnel Configuration Register
A2B_DTSLOTS	Data Tunnel Slots Register
A2B_DTDNOFFS	Data Tunnel Downstream Offset Register

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_DTUPOFFS	Data Tunnel Upstream Offset Register
A2B_GPIODEN	GPIO Over Distance Enable Register
A2B_GPIOD0MSK	GPIO Over Distance Mask 0 Register
A2B_GPIOD1MSK	GPIO Over Distance Mask 1 Register
A2B_GPIOD2MSK	GPIO Over Distance Mask 2 Register
A2B_GPIOD3MSK	GPIO Over Distance Mask 3 Register
A2B_GPIOD4MSK	GPIO Over Distance Mask 4 Register
A2B_GPIOD5MSK	GPIO Over Distance Mask 5 Register
A2B_GPIOD6MSK	GPIO Over Distance Mask 6 Register
A2B_GPIOD7MSK	GPIO Over Distance Mask 7 Register
A2B_GPIODDAT	GPIO Over Distance Data Register
A2B_GPIODINV	GPIO Over Distance Invert Register
A2B_MBOX0CTL	Mailbox 0 Control Register (Sub Only)
A2B_MBOX0STAT	Mailbox 0 Status Register (Sub Only)
A2B_MBOX0B0	Mailbox 0 Byte 0 Register (Sub Only)
A2B_MBOX0B1	Mailbox 0 Byte 1 Register (Sub Only)
A2B_MBOX0B2	Mailbox 0 Byte 2 Register (Sub Only)
A2B_MBOX0B3	Mailbox 0 Byte 3 Register (Sub Only)
A2B_MBOX1CTL	Mailbox 1 Control Register (Sub Only)
A2B_MBOX1STAT	Mailbox 1 Status Register (Sub Only)
A2B_MBOX1B0	Mailbox 1 Byte 0 Register (Sub Only)
A2B_MBOX1B1	Mailbox 1 Byte 1 Register (Sub Only)
A2B_MBOX1B2	Mailbox 1 Byte 2 Register (Sub Only)
A2B_MBOX1B3	Mailbox 1 Byte 3 Register (Sub Only)
A2B_SWCTL2	Switch Control Register 2
A2B_SWSTAT2	Switch Status Register 2
A2B_SPIDTLCMD	SPI Data Tunnel Last Command Register
A2B_SPICFG	SPI Configuration Register
A2B_SPISTAT	SPI Status Register
A2B_SPICKDIV	SPI Clock Divide Register
A2B_SPIFDSIZE	SPI Full Duplex Size Register

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_SPIFDTARG	SPI Full Duplex Target Register
A2B_SPIPINCFG	SPI Pin Configuration Register
A2B_SPIINT	SPI Interrupt Register
A2B_SPIMSK	SPI Interrupt Mask Register
A2B_RXMASK0	I2S/TDM RX Mask 0 Register
A2B_RXMASK1	I2S/TDM RX Mask 1 Register
A2B_RXMASK2	I2S/TDM RX Mask 2 Register
A2B_RXMASK3	I2S/TDM RX Mask 3 Register
A2B_RXMASK4	I2S/TDM RX Mask 4 Register
A2B_RXMASK5	I2S/TDM RX Mask 5 Register
A2B_RXMASK6	I2S/TDM RX Mask 6 Register
A2B_RXMASK7	I2S/TDM RX Mask 7 Register
A2B_TXXBAR0	Serial TX Crossbar Register 0
A2B_TXXBAR1	Serial TX Crossbar Register 1
A2B_TXXBAR2	Serial TX Crossbar Register 2
A2B_TXXBAR3	Serial TX Crossbar Register 3
A2B_TXXBAR4	Serial TX Crossbar Register 4
A2B_TXXBAR5	Serial TX Crossbar Register 5
A2B_TXXBAR6	Serial TX Crossbar Register 6
A2B_TXXBAR7	Serial TX Crossbar Register 7
A2B_TXXBAR8	Serial TX Crossbar Register 8
A2B_TXXBAR9	Serial TX Crossbar Register 9
A2B_TXXBAR10	Serial TX Crossbar Register 10
A2B_TXXBAR11	Serial TX Crossbar Register 11
A2B_TXXBAR12	Serial TX Crossbar Register 12
A2B_TXXBAR13	Serial TX Crossbar Register 13
A2B_TXXBAR14	Serial TX Crossbar Register 14
A2B_TXXBAR15	Serial TX Crossbar Register 15
A2B_TXXBAR16	Serial TX Crossbar Register 16
A2B_TXXBAR17	Serial TX Crossbar Register 17
A2B_TXXBAR18	Serial TX Crossbar Register 18

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_TXXBAR19	Serial TX Crossbar Register 19
A2B_TXXBAR20	Serial TX Crossbar Register 20
A2B_TXXBAR21	Serial TX Crossbar Register 21
A2B_TXXBAR22	Serial TX Crossbar Register 22
A2B_TXXBAR23	Serial TX Crossbar Register 23
A2B_TXXBAR24	Serial TX Crossbar Register 24
A2B_TXXBAR25	Serial TX Crossbar Register 25
A2B_TXXBAR26	Serial TX Crossbar Register 26
A2B_TXXBAR27	Serial TX Crossbar Register 27
A2B_TXXBAR28	Serial TX Crossbar Register 28
A2B_TXXBAR29	Serial TX Crossbar Register 29
A2B_TXXBAR30	Serial TX Crossbar Register 30
A2B_TXXBAR31	Serial TX Crossbar Register 31
A2B_MMRPAGE	MMR Page Register
A2B_VMTR_VEN	Enable Voltage Measurement
A2B_VMTR_INTEN	Min / Max Error Interrupt Enable
A2B_VMTR_MXSTAT	VMAX Check Result
A2B_VMTR_MNSTAT	VMIN Check Result
A2B_VMTR_VLTG0	Measured Voltage 0
A2B_VMTR_VMAX0	MAX Voltage Threshold
A2B_VMTR_VMIN0	VMIN Register 0
A2B_VMTR_VLTG1	Measured Voltage 1
A2B_VMTR_VMAX1	VMAX Register 1
A2B_VMTR_VMIN1	VMIN Register 1
A2B_VMTR_VLTG2	Measured Voltage 2
A2B_VMTR_VMAX2	VMAX Register 2
A2B_VMTR_VMIN2	VMIN Register 2
A2B_VMTR_VLTG3	Measured Voltage 3
A2B_VMTR_VMAX3	VMAX Register 3
A2B_VMTR_VMIN3	VMIN Register 3
A2B_VMTR_VLTG4	Measured Voltage 4

Table 11-1: AD2437 A2B Register List (Continued)

Name	Description
A2B_VMTR_VMAX4	VMAX Register 4
A2B_VMTR_VMIN4	VMIN Register 4
A2B_VMTR_VLTG5	Measured Voltage 5
A2B_VMTR_VMAX5	VMAX Register 5
A2B_VMTR_VMIN5	VMIN Register 5
A2B_VMTR_VLTG6	Measured Voltage 6
A2B_VMTR_VMAX6	VMAX Register 6
A2B_VMTR_VMIN6	VMIN Register 6
A2B_PWMCFG	PWM Configuration Register
A2B_PWMFREQ	PWM Frequency Register
A2B_PWMBLINK1	PWM Blink Register 1
A2B_PWMBLINK2	PWM Blink Register 2
A2B_PWM1VALL	PWM1 Value Low Bits Register
A2B_PWM1VALH	PWM1 Value High Bits Register
A2B_PWM2VALL	PWM2 Value Low Bits Register
A2B_PWM2VALH	PWM2 Value High Bits Register
A2B_PWM3VALL	PWM3 Value Low Bits Register
A2B_PWM3VALH	PWM3 Value High Bits Register
A2B_PWMOEVALL	PWM OE Value Low Bits Register
A2B_PWMOEVALH	PWM OE Value High Bits Register
A2B_MMRPAGE1	MMR Page Register

I2C Chip Address Register (Sub Only)

The `A2B_CHIP` register stores a 7-bit I^2C chip address for a connected I^2C peripheral. It is used during I^2C transactions to address a remote peripheral device connected to a subordinate node. The A^2B subordinate node acts as the I^2C controller in I^2C transactions with peripherals. This register only has an effect on I^2C when programmed in a subordinate node. The register can be written to and read from in a main node without any influence on the chip's functionality.

Address: 0x00

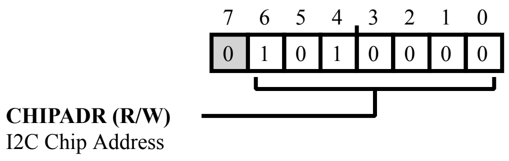


Figure 11-1: A2B_CHIP Register Diagram

Table 11-2: A2B_CHIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	CHIPADR	I2C Chip Address. The `A2B_CHIP.CHIPADR` bit field stores the I^2C address used by a subordinate transceiver for I^2C accesses to a locally-connected peripheral. The A^2B subordinate node acts as the I^2C controller in I^2C transactions with peripherals.

Node Address Register (Main Only)

The A2B_NODEADR register provides control bits for addressing subordinate nodes through the A^{2}B bus. This register can only be written in the main node. A write to this address in a subordinate node has no effect.

Address: 0x01

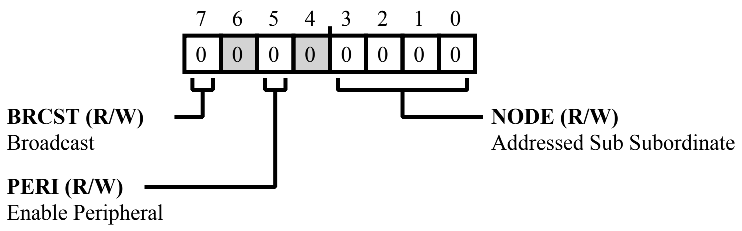


Figure 11-2: A2B_NODEADR Register Diagram

Table 11-3: A2B_NODEADR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	BRCST	Broadcast. The A2B_NODEADR.BRCST bit enables broadcast mode. When an I^{2}C write occurs in broadcast mode, the same control data is written to all nodes (main and subordinates) simultaneously. The broadcast allows simultaneous control of all discovered A^{2}B transceivers, but not their respective I^{2}C peripherals. Therefore, clear the A2B_NODEADR.PERI bit (=0) when the A2B_NODEADR.BRCST bit is set to 1.	
		0	Normal, directed register access
		1	Write to all nodes handled as broadcast access
5 (R/W)	PERI	Enable Peripheral. The A2B_NODEADR.PERI bit enables register access (over I^{2}C) of peripheral devices on subordinate nodes. The A2B_NODEADR.BRCST bit must be cleared (=0) when the A2B_NODEADR.PERI bit is set. When accessing subordinate node registers through BUS_ADDR, the A2B_NODEADR.PERI bit must be cleared.	
		0	Remote peripheral access disabled
		1	Remote peripheral access enabled
3:0 (R/W)	NODE	Addressed Sub Subordinate. The A2B_NODEADR.NODE bit field selects a subordinate node by its address. Addresses are assigned based on the position in the A^{2}B topology, starting with address 0 for the node connected directly to the main node. The value of the A2B_NODEADR.NODE field is irrelevant when the A2B_NODEADR.BRCST bit is set.	
		0-9	Node number
		10-15	Reserved

Vendor ID Register

The A2B_VENDOR register identifies the part as manufactured by Analog Devices.

Address: 0x02

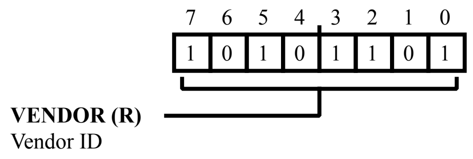


Figure 11-3: A2B_VENDOR Register Diagram

Table 11-4: A2B_VENDOR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VENDOR	Vendor ID. The A2B_VENDOR.VENDOR bit field contains the vendor identification number of the transceiver chip.

Product ID Register

The A2B_PRODUCT register identifies the last two digits of the part number in hexadecimal format (for example, 0x37=AD2437).

Address: 0x03

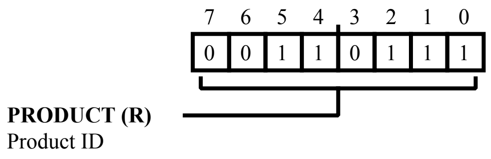


Figure 11-4: A2B_PRODUCT Register Diagram

Table 11-5: A2B_PRODUCT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/NW)	PRODUCT	Product ID. The A2B_PRODUCT.PRODUCT bit field contains the product identification number of the transceiver.	
		55	0x37 for Product ID AD2437

Version ID Register

The A2B_VERSION register identifies the version of the part.

Address: 0x04

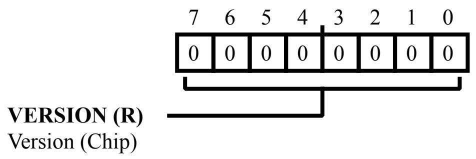


Figure 11-5: A2B_VERSION Register Diagram

Table 11-6: A2B_VERSION Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VERSION	Version (Chip). The A2B_VERSION.VERSION bit field contains the production version number of the chip . Bits 7:4 indicate major product revisions, while bits 3:0 are for minor revisions.

Capability ID Register

The A2B_CAPABILITY register identifies available control interfaces. Transceivers that have an EEPROM storage device connected can store specific descriptor information in the EEPROM module.

Address: 0x05

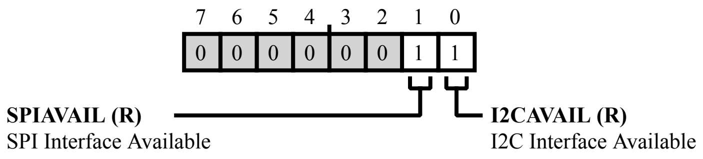


Figure 11-6: A2B_CAPABILITY Register Diagram

Table 11-7: A2B_CAPABILITY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	SPIAVAIL	SPI Interface Available. If the A2B_CAPABILITY.SPIAVAIL bit =1 then module descriptor information may be accessible over the SPI interface. Transceiver chips that have an EEPROM storage device connected to the SPI port can store module specific descriptor information in the following format. addr: data: 0x00: 0xAB identifies availability of module information in the module 0x01: Module Vendor ID 0x02: Module Product ID 0x03: Module Version ID Module Vendor ID can only be assigned by the A2B consortium leadership currently residing at Analog Devices. SPI EEPROM example part no: M95256	
		0	No SPI interface available on transceiver
		1	SPI interface available on transceiver
0 (R/NW)	I2CAVAIL	I2C Interface Available. The A2B_CAPABILITY.I2CAVAIL bit signals availability of the I^2C interface on the transceiver for access to peripheral devices. If this bit is set (=1), module descriptor information can be accessible through the I^2C interface. A connected EEPROM (for example, an AT24C01) with module descriptor information must have an I^2C device address of 0x50.	
		0	No I^2C interface is available
		1	I^2C interface is available

Switch Control Register

The A2B_SWCTL register controls the switching of A^{2}B bus power onto the downstream B-side of the A^{2}B bus. This register must be written over the A^{2}B bus. A write to this register from the local I^{2}C port has no effect.

Address: 0x09

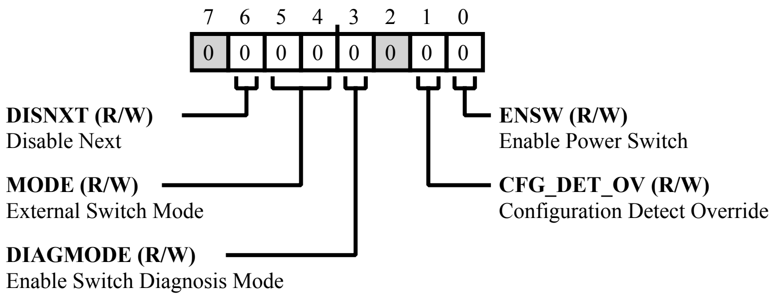


Figure 11-7: A2B_SWCTL Register Diagram

Table 11-8: A2B_SWCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DISNXT	Disable Next. The A2B_SWCTL.DISNXT bit controls how SYNC packets are sent to the next node for discovery. When cleared =0, automatically enable SYNC packets being passed to the next node after A2B_SWCTL.ENSW is programmed to 1 when the internal A2B_SWSTAT.FIN signal goes high (signaling successful switching). When set =1, disable SYNC packets to be sent automatically after A2B_SWCTL.ENSW is programmed to 1. The synchronization control frame and downstream data are not sent to LVDS XCVR B even if A2B_SWCTL.ENSW and A2B_SWSTAT.FIN are both high.	
		0	Enable Passing SYNC packets
		1	Disable Passing SYNC packets
5:4 (R/W)	MODE	External Switch Mode. The A2B_SWCTL.MODE bit field defines the diagnostic fault detection method for biasing the B-side A^{2}B bus with bus power for the next node. The setting depends on the external hardware configuration. {*E*, unexpected '$'. }	
		1	Downstream node not using bus power and not properly terminating the bias. Use for discovery
		2	Avoid fault due to inrush current Voltage on the VBUS pin differs from voltage on VSENSEP pin. To avoid faults due to large inrush current that may be perceived during discovery.
		3	Engineering Debug and Test Mode

Table 11-8: A2B_SWCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DIAGMODE	Enable Switch Diagnosis Mode. The `A2B_SWCTL.DIAGMODE` bit enables switch diagnosis mode.	
		0	Switch Diagnosis Mode Disabled
		1	Switch Diagnosis Mode Enabled
1 (R/W)	CFG_DET_OV	Configuration Detect Override. The `A2B_SWCTL.CFG_DET_OV` bit overrides the detected power configuration. Use the `A2B_SWCTL2.HPSW_CFG` field to indicate a new power configuration.	
0 (R/W)	ENSW	Enable Power Switch. The `A2B_SWCTL.ENSW` bit enables the power switches (FETs) for bus bias and discovery of the next-in-line node. It controls SWP pin and starts the communication signal for discovery.	
		0	Switch Disabled. No node discovery.
		1	Switch Enabled. Begin the next-in-line node discovery.

Broadcast Downstream Slots Register (Sub Only)

In a subordinate node, the A2B_BCDNSLOTS register defines the number of data slots which are captured by the node and also passed downstream (B-side) as broadcast data to the next node. If any bits are set in the A2B_DNMASK0 through A2B_DNMASK3 registers, the value of the A2B_BCDNSLOTS register is ignored. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node. The A2B_BCDNSLOTS register is not used in the main node.

Address: 0x0A

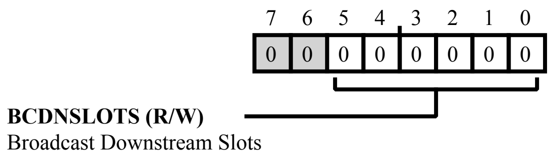


Figure 11-8: A2B_BCDNSLOTS Register Diagram

Table 11-9: A2B_BCDNSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	BCDNSLOTS	Broadcast Downstream Slots. The A2B_BCDNSLOTS.BCDNSLOTS bit field configures the number of broadcast downstream slots. This field must be programmed with a value between 0 and 32.

Local Downstream Slots Register (Sub Only)

In a subordinate node, the meaning of the A2B_LDNSLOTS register changes depending on whether or not the downstream broadcast mask enable bit (A2B_LDNSLOTS.DNMASKEN) is set. If A2B_LDNSLOTS.DNMASKEN=0 (default), the A2B_LDNSLOTS register defines the number of data slots which are captured by the local node during the downstream portion of the superframe. These data slots are consumed by the node and are not passed downstream to the next node. If A2B_LDNSLOTS.DNMASKEN=1, the A2B_LDNSLOTS register defines the number of data slots that are added by the local node during the downstream portion of the superframe after A2B_DNSLOTS.DNSLOTS data slots are passed downstream by the transceiver. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node.

Address: 0x0B

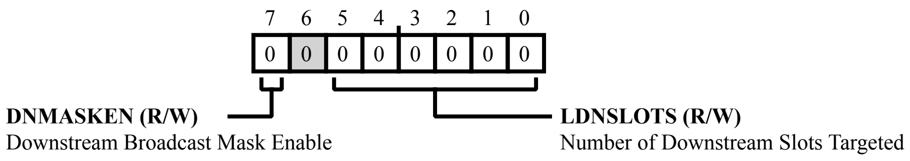

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
DNMASKEN (R/W)
Downstream Broadcast Mask Enable
LDNSLOTS (R/W)
Number of Downstream Slots Targeted

Figure 11-9: A2B_LDNSLOTS Register Diagram

Table 11-10: A2B_LDNSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	DNMASKEN	Downstream Broadcast Mask Enable. The A2B_LDNSLOTS.DNMASKEN bit enables the downstream mask enable bits in the A2B_DNMASK0 through A2B_DNMASK3 registers.	
		0	Downstream data slot masks disabled
		1	Downstream data slot masks enabled
5:0 (R/W)	LDNSLOTS	Number of Downstream Slots Targeted. When A2B_LDNSLOTS.DNMASKEN=0, the A2B_LDNSLOTS.LDNSLOTS bit field defines the number of data slots which are captured by the local node during the downstream portion of the superframe. When A2B_LDNSLOTS.DNMASKEN=1, the A2B_LDNSLOTS.LDNSLOTS bit field defines the number of data slots which are added by the local node during the downstream portion of the superframe. This field must be programmed with a value between 0 and 32 and be sufficient to accommodate all the data relative to its mode of TDM operation and the number of enabled data pins.	

Local Upstream Slots Register (Sub Only)

In a subordinate node, the A2B_LUPSLOTS register defines the number of data slots which are added by the local node during the upstream portion of the superframe. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node. The A2B_LUPSLOTS register is not used in the main node.

Address: 0x0C

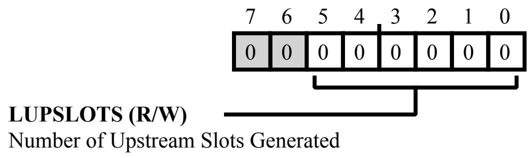


Figure 11-10: A2B_LUPSLOTS Register Diagram

Table 11-11: A2B_LUPSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	LUPSLOTS	Number of Upstream Slots Generated. The A2B_LUPSLOTS.LUPSLOTS bit field defines the number of data slots which are added by the transceiver during the upstream portion of the superframe. These bits must be programmed with a value between 0 and 32.

Downstream Slots Register

In a subordinate node, the A2B_DNSLOTS register defines the number of data slots (not including broadcast slots) that are passed downstream (B-side) after the transceiver begins to capture data slots. In the main node, the A2B_DNSLOTS register defines the total number of downstream data slots (including broadcast slots). Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node.

Address: 0x0D

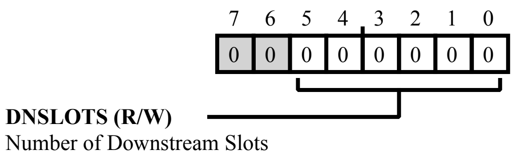


Figure 11-11: A2B_DNSLOTS Register Diagram

Table 11-12: A2B_DNSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	DNSLOTS	Number of Downstream Slots. In a main node, the A2B_DNSLOTS.DNSLOTS bit field is the number of downstream slots, including broadcast data slots. It must be sufficient to accommodate the data intended for downstream devices, which is a function of the TDM mode and the number of enabled data pins. In a subordinate node, the A2B_DNSLOTS.DNSLOTS bit field sets the number of data slots which are passed downstream. When calculating the value to program to this field, the same guidance as in the main node applies. But, subordinate nodes must also include any broadcast downstream slots, as programmed in the A2B_BCDNSLOTS register. Valid programming values are between 0 and 32.

Upstream Slots Register

In a subordinate node, the A2B_UPSLOTS register defines the number of data slots which are passed upstream by the B-side transceiver before the transceiver begins to add data slots. In the main node, the A2B_UPSLOTS register defines the total number of upstream data slots. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node.

Address: 0x0E

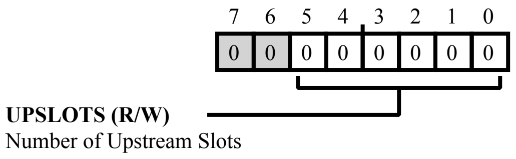


Figure 11-12: A2B_UPSLOTS Register Diagram

Table 11-13: A2B_UPSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	UPSLOTS	Number of Upstream Slots. In a main node, the A2B_UPSLOTS.UPSLOTS bit field is the number of upstream slots being received from the first-in-line subordinate node. It must be sufficient to accommodate all data intended for upstream devices, which is a function of TDM serial mode and the number of enabled data pins. In a slave node, the A2B_UPSLOTS.UPSLOTS bit field defines the number of data slots which are received from the next-in-line subordinate node and passed upstream before the transceiver begins to add data slots. Valid programming values are between 0 and 32.

Response Cycles Register

The A2B_RESPCYCS register defines the time between the start of the downstream header (the first SCF preamble bit) and the start of the upstream header (the first SRF preamble bit) in the superframe. The A^2B bus superframe consists of a total of 1024 bits, where one bus bit time = $1/(f_{SYSBCLK})$. As an 8-bit register, the Response Cycle register defines the response cycle time of a node as (A2B_RESPCYCS * 4) + 7. Refer to the Response cycle section for more details.

The DISCVRY register in the main transceiver is programmed with the A2B_RESPCYCS register value during discovery. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node. This register must be written over the A^2B bus, as writes to this register from the local I^2C port have no effect.

Address: 0x0F

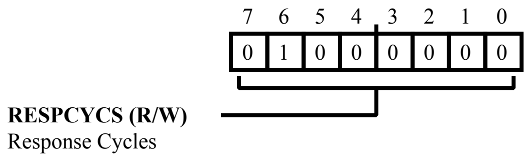


Figure 11-13: A2B_RESPCYCS Register Diagram

Table 11-14: A2B_RESPCYCS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RESPCYCS	Response Cycles. The A2B_RESPCYCS.RESPCYCS bit field is one-fourth the time from the start of a downstream frame to the start of an upstream frame.

Slot Format Register (Main Only, Auto-Broadcast)

The A2B_SLOTFMT register defines the size and format of the downstream and upstream data slots. Floating-point compression of A^2B data can be enabled to reduce bandwidth using this register, and ECC protection of A^2B data can alternately be enabled. All nodes in an A^2B system are subject to the same upstream and downstream slot format setting. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit in the main node.

When the A2B_SLOTFMT register is written in the main node, the new setting is automatically broadcast to all discovered subordinate nodes over the A^2B bus. Local host writes to this register in a subordinate node have no effect.

Address: 0x10

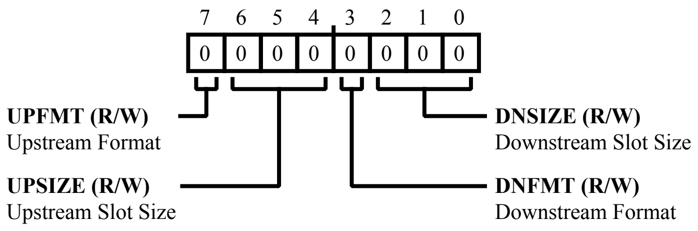


Figure 11-14: A2B_SLOTFMT Register Diagram

Table 11-15: A2B_SLOTFMT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	UPFMT	Upstream Format. The A2B_SLOTFMT.UPFMT bit configures the format of the upstream data on the A^{2}B bus. When A2B_SLOTFMT.UPFMT= 0, the format of the upstream data slots on the A^{2}B bus is normal (no compression, no ECC protection, and protected by one parity bit). When A2B_SLOTFMT.UPFMT = 1, an alternate data format is utilized, depending on the upstream data width (A2B_SLOTFMT.UPSIZE). When the A2B_SLOTFMT.UPSIZE bit is programmed for 12-, 16-, or 20-bit data, setting the A2B_SLOTFMT.UPFMT bit enables floating-point compression of upstream data. When this compression is used, the I^{2}S/TDM or PDM data is 4 bits wider than the A^{2}B data, which is compressed to reduce A^{2}B bus bandwidth, and the data is protected by a parity bit. When the A2B_SLOTFMT.UPSIZE bit is programmed for 24- or 32-bit data, setting the A2B_SLOTFMT.UPFMT bit enables ECC protection on upstream data slots, where ECC bits are added to each data slot instead of a parity bit (6 ECC bits for 24-bit data, 7 ECC bits for 32-bit data). Setting the A2B_SLOTFMT.UPFMT bit when A2B_SLOTFMT.UPSIZE is programmed for 8- or 28-bit data has no effect.	
		0	Normal upstream data slot format
		1	Alternate upstream data slot format
6:4 (R/W)	UPSIZE	Upstream Slot Size. The A2B_SLOTFMT.UPSIZE bit field selects the upstream data slot size.	
		0	8 bits
		1	12 bits
		2	16 bits
		3	20 bits
		4	24 bits
		5	28 bits
		6	32 bits
		7	Reserved

Table 11-15: A2B_SLOTFMT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DNFMT	Downstream Format. The `A2B_SLOTFMT.DNFMT` bit configures the format of the downstream data on the A^2B bus. When `A2B_SLOTFMT.DNFMT`= 0, the format of the downstream data slots on the A^2B bus is normal (no compression, no ECC protection, and protected by one parity bit). When `A2B_SLOTFMT.DNFMT` = 1, an alternate data format is utilized, depending on the downstream data width (`A2B_SLOTFMT.DNSIZE`). When the `A2B_SLOTFMT.DNSIZE` field is programmed for 12-, 16-, or 20-bit data, setting the `A2B_SLOTFMT.DNFMT` bit enables floating-point compression of downstream data. When this compression is used, the I^2S/TDM or PDM data is 4 bits wider than the A^2B data, which is compressed to reduce A^2B bus bandwidth, and the data is protected by a parity bit. When the `A2B_SLOTFMT.DNSIZE` bit is programmed for 24- or 32-bit data, setting the `A2B_SLOTFMT.DNFMT` bit enables ECC protection on downstream data slots, where ECC bits are added to each data slot instead of a parity bit (6 ECC bits for 24-bit data, 7 ECC bits for 32-bit data). Setting the `A2B_SLOTFMT.DNFMT` bit when `A2B_SLOTFMT.DNSIZE` is programmed for 8- or 28-bit data has no effect.	
		0	Normal downstream data slot format
		1	Alternate downstream data slot format
2:0 (R/W)	DNSIZE	Downstream Slot Size. The `A2B_SLOTFMT.DNSIZE` bit field selects the downstream data slot size.	
		0	8 bits
		1	12 bits
		2	16 bits
		3	20 bits
		4	24 bits
		5	28 bits
		6	32 bits
		7	Reserved

Data Control Register (Main Only, Auto-Broadcast)

The `A2B_DATCTL` register is used to enable data slots and standby mode on the A^{2}B bus. Changes to this register only take effect after setting the `A2B_CONTROL.NEWSTRCT` bit in the main node. When the `A2B_DATCTL` register is written in the main node, the new setting is automatically broadcast to all discovered subordinate nodes over the A^{2}B bus. Local host writes to this register in a subordinate node have no effect.

NOTE: To switch back to normal operation, first exit the standby mode by clearing the `A2B_DATCTL.STANDBY` bit, then write to the `A2B_DATCTL` register to enable the upstream and downstream slots.

Address: 0x11

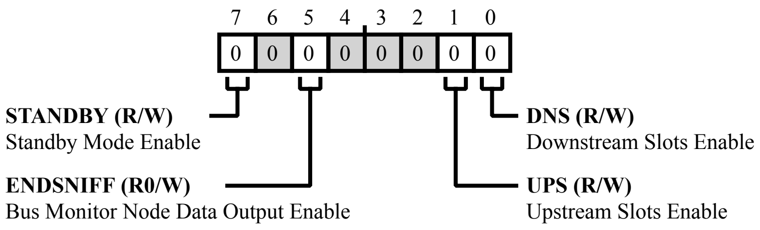


Figure 11-15: A2B_DATCTL Register Diagram

Table 11-16: A2B_DATCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	STANDBY	Standby Mode Enable. The `A2B_DATCTL.STANDBY` bit globally enables power saving mode for all nodes and minimizes bus activity. The only traffic required is a minimal downstream preamble to keep all of the PLLs in the subordinate nodes synchronized. Reads and writes across the A^{2}B bus are not supported in this mode.	
		0	Disabled
		1	Enabled
5 (R0/W)	ENDSNIFF	Bus Monitor Node Data Output Enable. The `A2B_DATCTL.ENDSNIFF` bit controls whether or not an attached Bus Monitor Node will produce data slots as output.	
		0	Disabled
		1	Enabled
1 (R/W)	UPS	Upstream Slots Enable. The `A2B_DATCTL.UPS` bit globally enables upstream synchronous data to be sent over the bus.	
		0	Disabled
		1	Enabled

Table 11-16: A2B_DATCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	DNS	Downstream Slots Enable. The `A2B_DATCTL.DNS` bit globally enables downstream synchronous data to be sent over the bus.	
		0	Disabled
		1	Enabled

Control Register

The A2B_CONTROL register provides bits which control nodes on the bus.

Address: 0x12

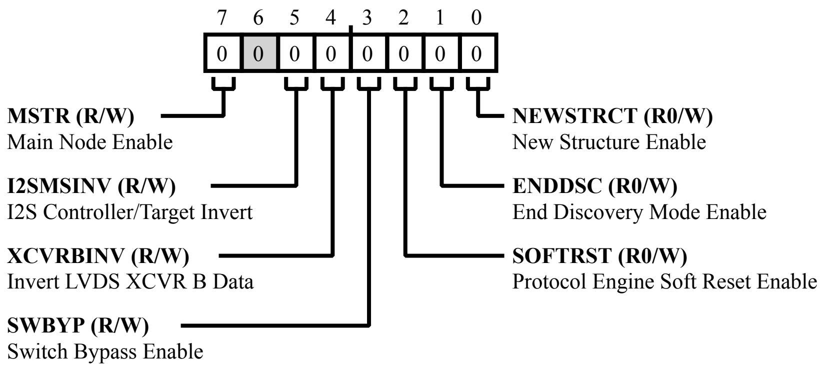


Figure 11-16: A2B_CONTROL Register Diagram

Table 11-17: A2B_CONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	MSTR	Main Node Enable. The A2B_CONTROL.MSTR bit controls whether the current node is a subordinate node or a main node.	
		0	Subordinate node
		1	Main node
5 (R/W)	I2SMSINV	I2S Controller/Target Invert. The A2B_CONTROL.I2SMSINV bit reverses whether the I^2C/TDM interface is a controller or target. When this bit is zero in a subordinate node, BCLK and SYNC are output pins. When this bit is one in a subordinate node, BCLK and SYNC are input pins. This bit currently has no impact in a main node or bus monitor node. The A2B_CONTROL.I2SMSINV bit must be zero in an increased rate subordinate node (A2B_I2SRATE.I2SRATE is 5 or 6) where the A2B_I2SRATE.REDUCE bit is one.	
4 (R/W)	XCVRBINV	Invert LVDS XCVR B Data. The A2B_CONTROL.XCVRBINV bit controls an optional inversion of data to/from LVDS XCVR B. Data is inverted when this bit is set.	
3 (R/W)	SWBYP	Switch Bypass Enable. The A2B_CONTROL.SWBYP bit enables the downstream LVDS XCVR without waiting for the line switch to be turned on. When this bit is set the line switch will not be enabled even if A2B_SWCTL.ENSW is set.	

Table 11-17: A2B_CONTROL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R0/W)	SOFTRST	Protocol Engine Soft Reset Enable. When the `A2B_CONTROL.SOFTRST` bit is set, the protocol engine in the bus node is reset, and all registers return to their respective reset states.	
		0	No action
		1	Reset protocol engine
1 (R0/W)	ENDDSC	End Discovery Mode Enable. In the main node, setting the `A2B_CONTROL.ENDDSC` bit ends discovery attempts to a new subordinate node.	
		0	No action
		1	End discovery
0 (R0/W)	NEWSTRCT	New Structure Enable. The `A2B_CONTROL.NEWSTRCT` bit synchronously applies a new structure to all nodes. When the `A2B_CONTROL.NEWSTRCT` bit is set in the main node, a new structure is applied within 5 superframe cycles unless communication errors create delays.	
		0	No action
		1	Enable new structure

Discovery Register (Main Only)

Programming the A2B_DISCVRY register with a response cycle value for a new node to be added allows the new subordinate node to be discovered. It triggers the start of full discovery frames being sent to the next-in-line subordinate node.

When the A2B_DISCVRY register is written in the main node, the new setting is automatically broadcast to all subordinate nodes over the A^2B bus. Local host and direct BUS_ADDR writes to this register in a subordinate node have no effect.

Note that the A2B_NODEADR.NODE bit field must be set to value of the previous node when writing to the A2B_DISCVRY register. For example, while discovering node-n, the A2B_NODEADR.NODE bit field must be (n-1) before writing to A2B_DISCVRY register.

Address: 0x13

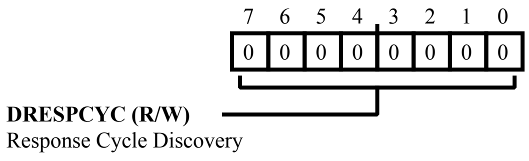


Figure 11-17: A2B_DISCVRY Register Diagram

Table 11-18: A2B_DISCVRY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	DRESPCYC	Response Cycle Discovery. The A2B_DISCVRY.DRESPCYC bit field is written with the value to be used for A2B_RESPCYCS by a to-be discovered subordinate node.

Switch Status Register

The A2B_SWSTAT register provides line diagnostics status information. Line diagnostics are performed when bias is switched onto the A^2B bus towards the next-in-line subordinate node.

Address: 0x14

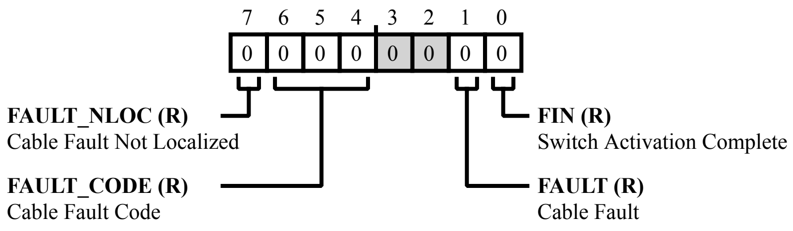


Figure 11-18: A2B_SWSTAT Register Diagram

Table 11-19: A2B_SWSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	FAULT_NLOC	Cable Fault Not Localized. The A2B_SWSTAT.FAULT_NLOC bit indicates that the identified line fault is not localized. Applicable only with single pair wire power scheme	
		0	Switch fault localized
		1	Switch fault not localized
6:4 (R/NW)	FAULT_CODE	Cable Fault Code. The A2B_SWSTAT.FAULT_CODE bit field contains downstream link cable diagnostic error codes.	
		0	No fault detected
		1	Cable terminal shorted to GND
		2	Cable terminal shorted to VBUS
		3	Cable terminals shorted together (Only with single pair wire)
		6	Cable Disconnected or Open Circuit or Reverse Connected
		7	Undetermined fault
1 (R/NW)	FAULT	Cable Fault. The A2B_SWSTAT.FAULT bit indicates a cable fault has been detected.	
		0	Cable fault not detected
		1	Cable fault detected

Table 11-19: A2B_SWSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	FIN	Switch Activation Complete. The `A2B_SWSTAT.FIN` bit indicates the successful completion of the switch activation sequence for biasing of the downstream link.	
		0	Switch is open or has not completed closing
		1	Switch completed closing

Interrupt Status Register

The A2B_INTSTAT register contains interrupt status information for the node.

Address: 0x15

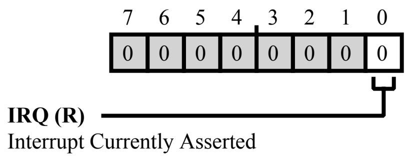


Figure 11-19: A2B_INTSTAT Register Diagram

Table 11-20: A2B_INTSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	IRQ	Interrupt Currently Asserted. When the A2B_INTSTAT.IRQ bit is set, the node is signaling an interrupt request, either through the IRQ pin for a main node or over the A^{2}B bus for a subordinate node.	
		0	No interrupt request
		1	Interrupt request

Interrupt Source Register (Main Only)

The A2B_INTSRC register contains information about the current highest priority interrupt. It is updated when the A2B_INTTYPE register is read. A value of 0x00 in this register indicates that no interrupts are present.

Address: 0x16

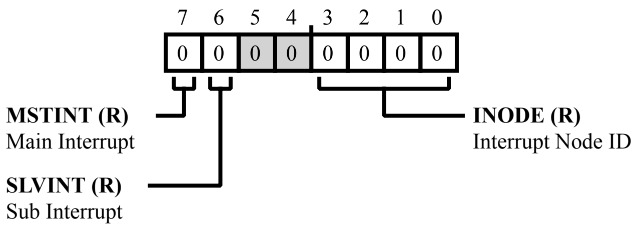


Figure 11-20: A2B_INTSRC Register Diagram

Table 11-21: A2B_INTSRC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	MSTINT	Main Interrupt. When the A2B_INTSRC.MSTINT bit is set, the current interrupt is being generated by the main node.
6 (R/NW)	SLVINT	Sub Interrupt. When the A2B_INTSRC.SLVINT bit is set, the current interrupt is being generated by a subordinate node.
3:0 (R/NW)	INODE	Interrupt Node ID. The A2B_INTSRC.INODE bit field contains the node number of the subordinate node that asserted the current interrupt.

Interrupt Type Register (Main Only)

The A2B_INTTYPE register contains information about the pending interrupt being generated by the node indicated in the A2B_INTSRC register and signaled with the IRQ pin. A host read of the A2B_INTTYPE register in the main node clears this pending interrupt in the main node and deasserts the IRQ pin. If other interrupts are pending, the A2B_INTSRC and A2B_INTTYPE registers are updated to reflect the highest priority pending interrupt, and the IRQ pin will again be asserted. Nodes closer to the main node have a higher priority when the same interrupt appears in more than one subordinate node.

Address: 0x17

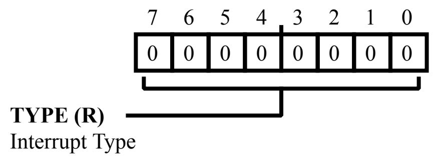


Figure 11-21: A2B_INTTYPE Register Diagram

Table 11-22: A2B_INTTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/NW)	TYPE	Interrupt Type. The A2B_INTTYPE.TYPE bit field contains the current interrupt type. Interrupt types are described in the interrupt pending registers (A2B_INTPND0 through A2B_INTPND2).	
		0	HDCNTERR - Header Count Error
		1	DDERR - Data Slot Decoding Error
		2	CRCERR- CRC Error (SCF CRC Error in subordinate node, SRF CRC Error in main node)
		3	DPERR - Data Slot Parity Error
		4	BECOVF - Bit Error Counter Overflow
		5	SRFMISSERR - SRF Missed Error
		6	SRFCRCERR - SRF CRC Error (sub node only)
		9	PWRERR - Shorted to GND
		10	PWRERR - Shorted to VBUS
		11	PWRERR - BP shorted to BN
		14	PWRERR - Cable is disconnected (open circuit) or wrong port or reverse connected
		15	PWRERR - Undetermined Cable Fault
		16	IO0PND - GPIO IO0 Pin Interrupt

Table 11-22: A2B_INTTYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		17	IO1PND - GPIO IO1 Pin Interrupt
		18	IO2PND - GPIO IO2 Pin Interrupt
		19	IO3PND - GPIO IO3 Pin Interrupt
		20	IO4PND - GPIO IO4 Pin Interrupt
		21	IO5PND - GPIO IO5 Pin Interrupt
		22	IO6PND - GPIO IO6 Pin Interrupt
		23	IO7PND - GPIO IO7 Pin Interrupt
		24	DSCDONE - Subordinate node discovery done (main only)
		25	I2CERR - I2C Error (controller only)
		26	ICRCERR - IRQ Field CRC Error (main only)
		41	PWRERR - Shorted to GND
		42	PWRERR - Shorted to VBUS
		47	Reserved
		48	Mailbox 0 full
		49	Mailbox 0 empty
		50	Mailbox 1 full
		51	Mailbox 1 empty
		64	SPI done
		65	SPI remote register access error - Main only
		66	SPI remote I2C access error - Main only
		67	SPI Data Tunnel Access Error
		68	SPI Bad Command
		69	SPI FIFO Overflow
		70	SPI FIFO Underflow
		80	VMTR Interrupt
		128	Interrupt messaging error - Main only
		252	Startup error - Return to factory
		253	Slave INTTYPE read error - Main only
		254	Standby done - Main only
		255	MSTR_RUNNING - Main node PLL locked

Interrupt Pending 0 Register

The A2B_INTPND0 register contains interrupt pending bits for the node.

Address: 0x18

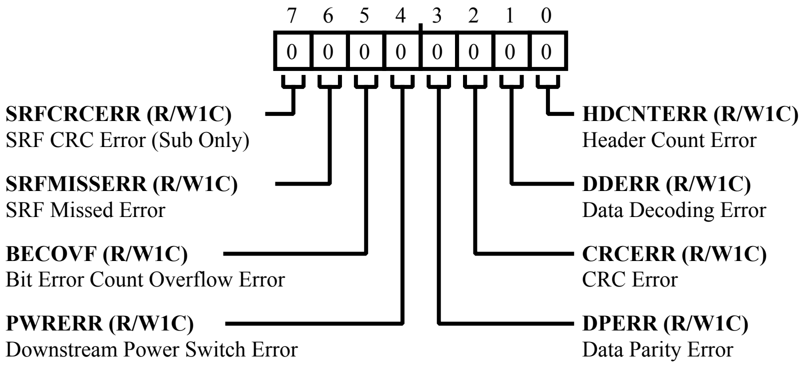


Figure 11-22: A2B_INTPND0 Register Diagram

Table 11-23: A2B_INTPND0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	SRFCRCERR	SRF CRC Error (Sub Only). The A2B_INTPND0.SRFCRCERR bit indicates that the current subordinate node has detected an SRF (upstream) CRC error.	
		0	No SRF CRC error
		1	SRF CRC error seen
6 (R/W1C)	SRFMISSERR	SRF Missed Error.	
		0	No missed SRF error
		1	Missed SRF error
5 (R/W1C)	BECOVF	Bit Error Count Overflow Error. The A2B_INTPND0.BECOVF bit indicates that the number of errors programmed into the bit error count control register has been exceeded.	
		0	No BEC Error Pending
		1	BEC Error Pending
4 (R/W1C)	PWRERR	Downstream Power Switch Error. The A2B_INTPND0.PWRERR bit indicates an error reported from the downstream power switch.	
		0	No power error
		1	Downstream power switch error

Table 11-23: A2B_INTPND0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	DPERR	Data Parity Error. The `A2B_INTPND0.DPERR` bit indicates that the current node has detected a data parity error. The error is detected only if the node consumes the data slot with a data parity error.	
		0	No data parity error
		1	Data parity error
2 (R/W1C)	CRCERR	CRC Error. The `A2B_INTPND0.CRCERR` bit indicates that the current node has detected a CRC error. For the main node, this applies to an upstream CRC error. For a subordinate node, this applies to a downstream CRC error.	
		0	No CRC Error
		1	CRC Error
1 (R/W1C)	DDERR	Data Decoding Error. The `A2B_INTPND0.DDERR` bit indicates that the current node has detected a data decoding error. The error is detected only if the node consumes the data slot with a data decoding error.	
		0	No data decoding error
		1	Data decoding error
0 (R/W1C)	HDCNTERR	Header Count Error. The `A2B_INTPND0.HDCNTERR` bit indicates the current node has detected a header count error. For the main node, this means that the SRF has a different count value than expected. For a subordinate node, this means that the SRF has a different value than expected.	
		0	No header count error
		1	Header count error

Interrupt Pending 1 Register

The A2B_INTPND1 register contains interrupt pending bits for the node.

Address: 0x19

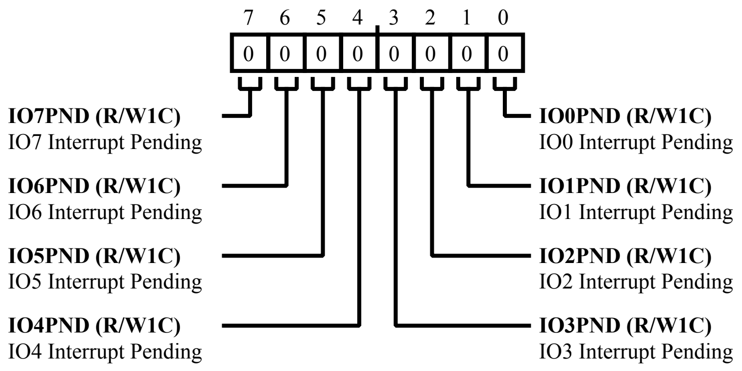


Figure 11-23: A2B_INTPND1 Register Diagram

Table 11-24: A2B_INTPND1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	IO7PND	IO7 Interrupt Pending. The A2B_INTPND1.IO7PND bit indicates that a pin interrupt request from IO7 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
6 (R/W1C)	IO6PND	IO6 Interrupt Pending. The A2B_INTPND1.IO6PND bit indicates that a pin interrupt request from IO6 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
5 (R/W1C)	IO5PND	IO5 Interrupt Pending. The A2B_INTPND1.IO5PND bit indicates that a pin interrupt request from IO5 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending

Table 11-24: A2B_INTPND1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	IO4PND	IO4 Interrupt Pending. The `A2B_INTPND1.IO4PND` bit indicates that a pin interrupt request from IO4 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
3 (R/W1C)	IO3PND	IO3 Interrupt Pending. The `A2B_INTPND1.IO3PND` bit indicates that a pin interrupt request from IO3 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
2 (R/W1C)	IO2PND	IO2 Interrupt Pending. The `A2B_INTPND1.IO2PND` bit indicates that a pin interrupt request from IO2 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
1 (R/W1C)	IO1PND	IO1 Interrupt Pending. The `A2B_INTPND1.IO1PND` bit indicates that a pin interrupt request from IO1 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending
0 (R/W1C)	IO0PND	IO0 Interrupt Pending. The `A2B_INTPND1.IO0PND` bit indicates that a pin interrupt request from IO0 is pending.	
		0	No Interrupt Pending
		1	Interrupt Pending

Interrupt Pending 2 Register (Main Only)

The A2B_INTPND2 register contains interrupt pending bits relevant only to main nodes.

Address: 0x1A

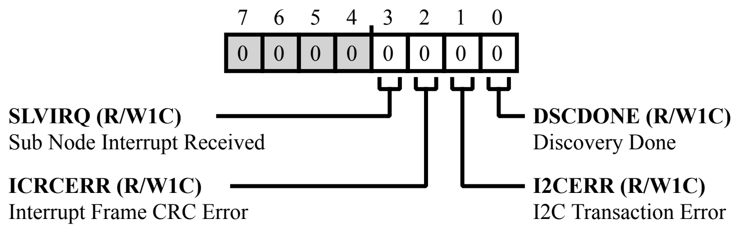


Figure 11-24: A2B_INTPND2 Register Diagram

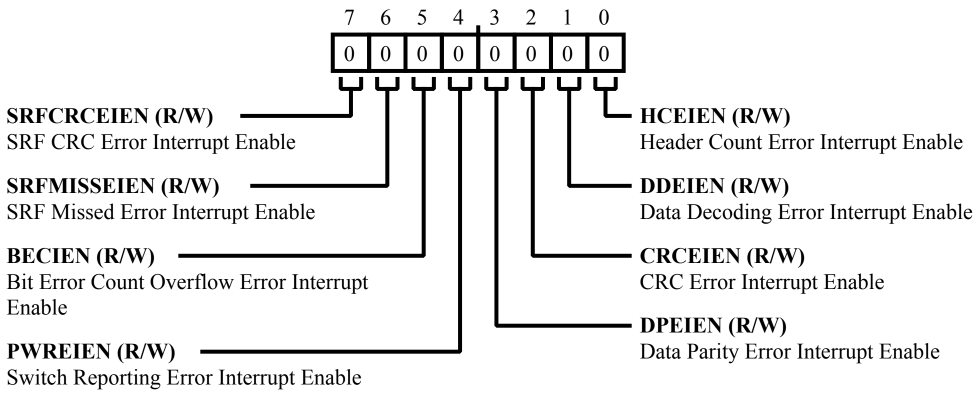
Table 11-25: A2B_INTPND2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	SLVIRQ	Sub Node Interrupt Received. In the main mode, the A2B_INTPND2.SLVIRQ bit indicates that a subordinate node has signaled an interrupt to the main node. This bit always reads zero in a subordinate node.	
		0	No interrupt
		1	Sub node has signaled an interrupt
2 (R/W1C)	ICRCERR	Interrupt Frame CRC Error. In the main mode, the A2B_INTPND2.ICRCERR bit indicates that the main node has detected an interrupt frame CRC error.	
		0	No error
		1	Interrupt frame CRC error detected
1 (R/W1C)	I2CERR	I2C Transaction Error. The A2B_INTPND2.I2CERR bit indicates that an I^2C access error has occurred. Examples of this are an I^2C write to a subordinate node with early acknowledge that did not complete or a broadcast write that timed out.	
		0	No error
		1	An I^2C access error occurred
0 (R/W1C)	DSCDONE	Discovery Done. The A2B_INTPND2.DSCDONE bit indicates that a new subordinate node has been discovered. This bit always reads zero in subordinate nodes.	
		0	No new sub node discovered
		1	New sub node discovered

Interrupt Mask 0 Register

The A2B_INTMSK0 register determines which A2B_INTPND0 register bits generate interrupts.

Address: 0x1B



7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
SRFCRCEIEN (R/W)
SRF CRC Error Interrupt Enable
SRFMISSEIEN (R/W)
SRF Missed Error Interrupt Enable
BECIEN (R/W)
Bit Error Count Overflow Error Interrupt Enable
PWREIEN (R/W)
Switch Reporting Error Interrupt Enable
HCEIEN (R/W)
Header Count Error Interrupt Enable
DDEIEN (R/W)
Data Decoding Error Interrupt Enable
CRCEIEN (R/W)
CRC Error Interrupt Enable
DPEIEN (R/W)
Data Parity Error Interrupt Enable

Figure 11-25: A2B_INTMSK0 Register Diagram

Table 11-26: A2B_INTMSK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	SRFCRCEIEN	SRF CRC Error Interrupt Enable.
6 (R/W)	SRFMISSEIEN	SRF Missed Error Interrupt Enable.
5 (R/W)	BECIEN	Bit Error Count Overflow Error Interrupt Enable.
4 (R/W)	PWREIEN	Switch Reporting Error Interrupt Enable.
3 (R/W)	DPEIEN	Data Parity Error Interrupt Enable.
2 (R/W)	CRCEIEN	CRC Error Interrupt Enable.
1 (R/W)	DDEIEN	Data Decoding Error Interrupt Enable.
0 (R/W)	HCEIEN	Header Count Error Interrupt Enable.

Interrupt Mask 1 Register

The A2B_INTMSK1 register determines which A2B_INTPND1 register bits generate interrupts.

Address: 0x1C

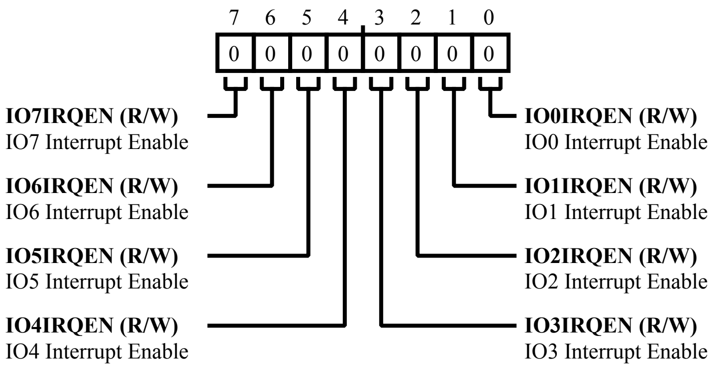


Figure 11-26: A2B_INTMSK1 Register Diagram

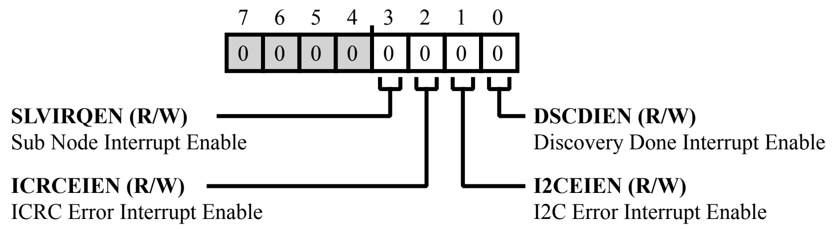
Table 11-27: A2B_INTMSK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	IO7IRQEN	IO7 Interrupt Enable.
6 (R/W)	IO6IRQEN	IO6 Interrupt Enable.
5 (R/W)	IO5IRQEN	IO5 Interrupt Enable.
4 (R/W)	IO4IRQEN	IO4 Interrupt Enable.
3 (R/W)	IO3IRQEN	IO3 Interrupt Enable.
2 (R/W)	IO2IRQEN	IO2 Interrupt Enable.
1 (R/W)	IO1IRQEN	IO1 Interrupt Enable.
0 (R/W)	IO0IRQEN	IO0 Interrupt Enable.

Interrupt Mask 2 Register (Main Only)

The A2B_INTMSK2 register determines which A2B_INTPND2 register bits generate interrupts.

Address: 0x1D



7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
SLVIRQEN (R/W)
Sub Node Interrupt Enable
ICRCEIEN (R/W)
ICRC Error Interrupt Enable
DSCDIEN (R/W)
Discovery Done Interrupt Enable
I2CEIEN (R/W)
I2C Error Interrupt Enable

Figure 11-27: A2B_INTMSK2 Register Diagram

Table 11-28: A2B_INTMSK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	SLVIRQEN	Sub Node Interrupt Enable.
2 (R/W)	ICRCEIEN	ICRC Error Interrupt Enable.
1 (R/W)	I2CEIEN	I2C Error Interrupt Enable.
0 (R/W)	DSCDIEN	Discovery Done Interrupt Enable.

Bit Error Count Control Register

The A2B_BECCTL register controls bit error counting, including interrupt thresholds.

Address: 0x1E

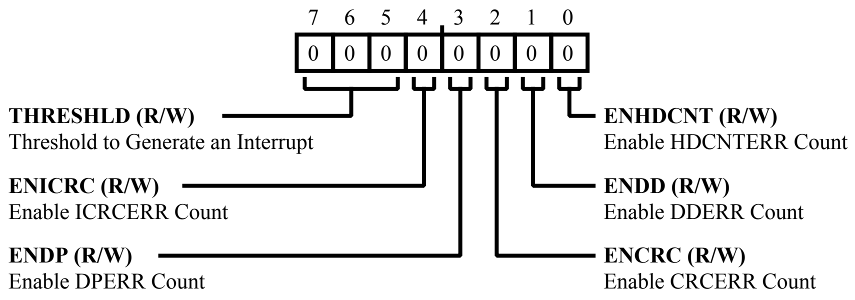


Figure 11-28: A2B_BECCTL Register Diagram

Table 11-29: A2B_BECCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:5 (R/W)	THRESHLD	Threshold to Generate an Interrupt. The A2B_BECCTL.THRESHLD bit field configures the number of errors counted before the A2B_INTPND0.BECOVF bit is set.	
		0	Interrupt if A2B_BECNT > 2
		1	Interrupt if A2B_BECNT > 4
		2	Interrupt if A2B_BECNT > 8
		3	Interrupt if A2B_BECNT > 16
		4	Interrupt if A2B_BECNT > 32
		5	Interrupt if A2B_BECNT > 64
		6	Interrupt if A2B_BECNT > 128
		7	Interrupt if A2B_BECNT > 256
4 (R/W)	ENICRC	Enable ICRCERR Count. When the A2B_BECCTL.ENICRC bit is set, the bit error count register is incremented every time a CRC error is detected in the interrupt response frame.	
		0	Disabled
		1	Enable Bit Error Counting

Table 11-29: A2B_BECCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	ENDP	Enable DPERR Count. When the `A2B_BECCTL.ENDP` bit is set, the bit error count register is incremented on every parity error of the streaming data.	
		0	No Parity error
		1	Parity Error
2 (R/W)	ENCRC	Enable CRCERR Count. When the `A2B_BECCTL.ENCRC` bit is set, the bit error count register is incremented on every CRC error in a control or response frame. This excludes interrupt frame CRC errors and SRF CRC errors.	
		0	No CRC Error
		1	CRC Error
1 (R/W)	ENDD	Enable DDERR Count. When the `A2B_BECCTL.ENDD` bit is set, the bit error count register is incremented on every data decoding error.	
		0	Disabled
		1	Enabled
0 (R/W)	ENHDCNT	Enable HDCNTERR Count. When the `A2B_BECCTL.ENHDCNT` bit is set, the bit error count register is incremented if there is a discrepancy between the actual and expected header count field.	
		0	Disabled
		1	Enabled

Bit Error Count Register

The A2B_BECNT register accumulates the error count of the error types selected in the A2B_BECCTL register. Any write to this register clears the count.

Address: 0x1F

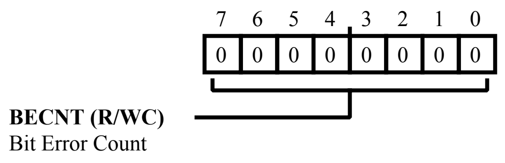


Figure 11-29: A2B_BECNT Register Diagram

Table 11-30: A2B_BECNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/WC)	BECNT	Bit Error Count. The A2B_BECNT.BECNT bit field provides the number of bit errors counted, based on the value programmed into the A2B_BECCTL register.

Testmode Register

The A2B_TESTMODE register provides control bits to be used in testing the A^{2}B link. The A2B_TESTMODE.PRBSDN and A2B_TESTMODE.PRBSUP bits are used to enable the use of pseudo-random data in the downstream and upstream data slots on the A^{2}B bus, respectively. Downstream data is checked in the last subordinate node based on the programming of the A2B_DNSLOTS, LDNSLOTS, and BCDNSLOTS registers. Upstream data is checked in the main node. Data mismatches increment a 32-bit counter (which can be read via the A2B_ERRCNT0 through A2B_ERRCNT3 registers). The A2B_TESTMODE register must be programmed via a broadcast write. Subordinate to subordinate communications adversely affect a Bit Error Rate Test (BERT).

Address: 0x20

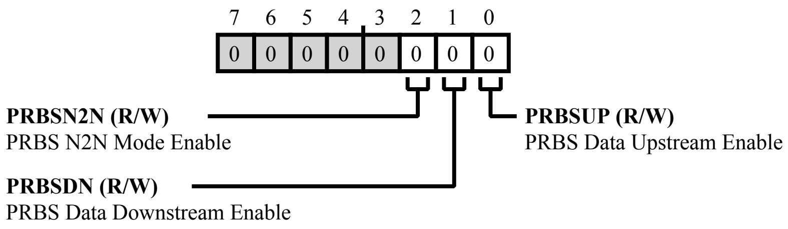


Figure 11-30: A2B_TESTMODE Register Diagram

Table 11-31: A2B_TESTMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	PRBSN2N	PRBS N2N Mode Enable. When the A2B_TESTMODE.PRBSN2N bit is set, each node checks all incoming data bits and transmits the expected data to the next node. This allows for better determination of where bus errors occur. This bit only takes effect when either or both of the A2B_TESTMODE.PRBSDN and A2B_TESTMODE.PRBSUP bits are set.	
		0	Disabled
		1	Enabled
1 (R/W)	PRBSDN	PRBS Data Downstream Enable. The A2B_TESTMODE.PRBSDN bit enables PRBS data to be sent downstream towards the last subordinate node.	
		0	Normal Data
		1	PRBS Data
0 (R/W)	PRBSUP	PRBS Data Upstream Enable. The A2B_TESTMODE.PRBSUP bit enables PRBS data to be sent upstream towards the main node.	
		0	Normal Data
		1	PRBS Data

PRBS Error Count Byte 0 Register

The A2B_ERRCNT0 register holds the least significant byte of the 32-bit error count accumulated during the PRBS bit error test.

Address: 0x21

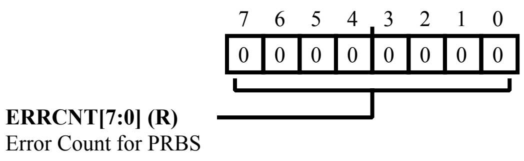


Figure 11-31: A2B_ERRCNT0 Register Diagram

Table 11-32: A2B_ERRCNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ERRCNT	Error Count for PRBS. The A2B_ERRCNT0.ERRCNT bit field contains one byte of the 32-bit PRBS bit error count.

PRBS Error Count Byte 1 Register

The A2B_ERRCNT1 register holds the second byte (bits 15:8) of the error count accumulated during the PRBS bit error test.

Address: 0x22

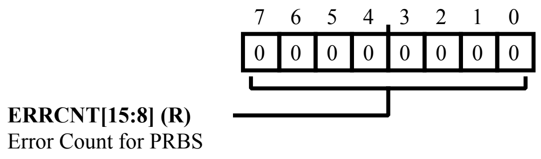


Figure 11-32: A2B_ERRCNT1 Register Diagram

Table 11-33: A2B_ERRCNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ERRCNT	Error Count for PRBS. The A2B_ERRCNT1.ERRCNT bit field contains one byte of the 32-bit PRBS bit error count.

PRBS Error Count Byte 2 Register

The A2B_ERRCNT2 register holds the third byte (bits 23:16) of the error count accumulated during the PRBS bit error test.

Address: 0x23

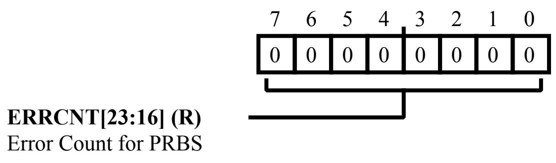


Figure 11-33: A2B_ERRCNT2 Register Diagram

Table 11-34: A2B_ERRCNT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ERRCNT	Error Count for PRBS. The A2B_ERRCNT2.ERRCNT bit field contains one byte of the 32-bit PRBS bit error count.

PRBS Error Count Byte 3 Register

The A2B_ERRCNT3 register holds the most significant byte (bits 31:24) of the 32-bit error count accumulated during the PRBS bit error test. The A2B_ERRCNT0 register is the least significant byte of the 32-bit error count.

Address: 0x24

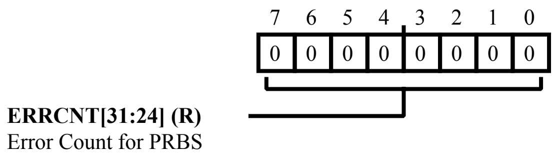


Figure 11-34: A2B_ERRCNT3 Register Diagram

Table 11-35: A2B_ERRCNT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ERRCNT	Error Count for PRBS. The `A2B_ERRCNT3.ERRCNT` bit field contains one byte of the 32-bit PRBS bit error count.

Node Register

The A2B_NODE register contains information required for node-to-node communication.

Address: 0x29

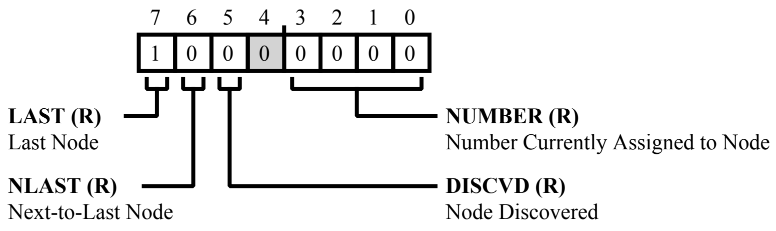


Figure 11-35: A2B_NODE Register Diagram

Table 11-36: A2B_NODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	LAST	Last Node. The A2B_NODE.LAST bit indicates that this node is not connected to a downstream node. It is set by default at reset and cleared during discovery.	
		0	Not Last Node
		1	Last Node
6 (R/NW)	NLAST	Next-to-Last Node. The A2B_NODE.NLAST bit indicates that this node is directly upstream of the last node. It is set during discovery.	
		0	Not Next-to-Last Node
		1	Next-to-Last Node
5 (R/NW)	DISCVD	Node Discovered. The A2B_NODE.DISCVD bit indicates that this node has been discovered. This bit always reads as 0 in a main node.	
		0	Not Discovered
		1	Discovered
3:0 (R/NW)	NUMBER	Number Currently Assigned to Node. The A2B_NODE.NUMBER bit field reports the node number assigned to the node during discovery. This field always reads as 0 in a main node.	

Discovery Status Register (Main Only)

The A2B_DISCSTAT register provides status for discovery transactions on the A^2B bus. An I^2C write to the A2B_DISCVRY register sets the A2B_DISCSTAT.DSCACT bit and causes the A2B_NODEADR.NODE field to be written to this register. Discovery mode can be aborted by writing to the A2B_CONTROL.ENDDSC bit.

Address: 0x2B

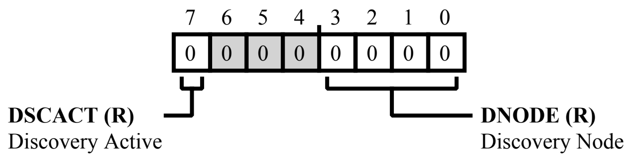


Figure 11-36: A2B_DISCSTAT Register Diagram

Table 11-37: A2B_DISCSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	DSCACT	Discovery Active. The A2B_DISCSTAT.DSCACT bit is set while the main node is in discovery mode.
3:0 (R/NW)	DNODE	Discovery Node. When the A2B_DISCSTAT.DSCACT bit is set, the A2B_DISCSTAT.DNODE bit field shows the node being used for discovery frames. If A2B_DISCSTAT.DSCACT is cleared, the A2B_DISCSTAT.DNODE bit field retains the value of the last node discovered.

Local Interrupt Type (Sub Only)

The `A2B_LINTTYPE` register contains information about the pending local interrupt from a subordinate node to a local processor signaled with the IRQ pin. A read of the `A2B_LINTTYPE` register in a subordinate node by a local processor has the effect of clearing this pending interrupt as well as deasserting the IRQ pin. A read of this register by the Host processor has no effect. This register is only used when signaling mailbox and/or SPI slave interrupts to a local processor in an A^{2}B subordinate node.

Address: 0x3E

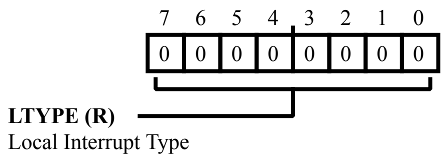


Figure 11-37: A2B_LINTTYPE Register Diagram

Table 11-38: A2B_LINTTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/NW)	LTYPE	Local Interrupt Type.	
		48	Mailbox 0 Full
		49	Mailbox 0 Empty
		50	Mailbox 1 Full
		51	Mailbox 1 Empty
		64	SPI Done
		67	SPI Data Tunnel Access Error
		68	SPI Bad Command
		69	SPI FIFO Overflow
		70	SPI FIFO Underflow

I2C Configuration Register

The A2B_I2CCFG register controls the data rate of the I^{2}C port in A^{2}B subordinate nodes and sets the I^{2}C behavior in the A^{2}B main node.

Address: 0x3F

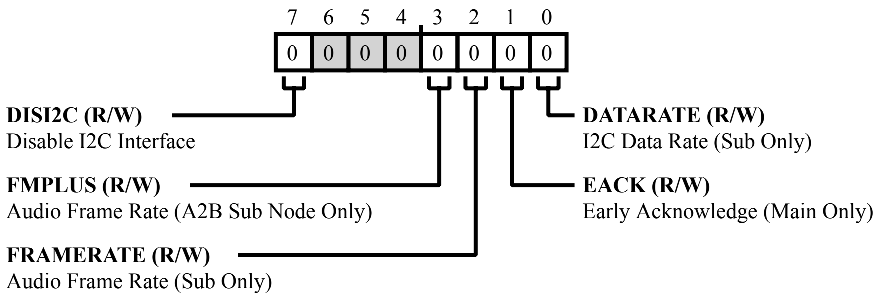


Figure 11-38: A2B_I2CCFG Register Diagram

Table 11-39: A2B_I2CCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	DISI2C	Disable I2C Interface. The A2B_I2CCFG.DISI2C bit disables the I^{2}C interface when set. The I^{2}C interface should be disabled only when it is not active.	
3 (R/W)	FMPLUS	Audio Frame Rate (A2B Sub Node Only). The A2B_I2CCFG.FMPLUS bit only works on a subordinate node. When set, this bit selects an I^{2}C Fast Mode Plus clock rate (1 MHz) on the I^{2}C bus. This bit takes priority over the A2B_I2CCFG.DATARATE bit.	
2 (R/W)	FRAMERATE	Audio Frame Rate (Sub Only). The A2B_I2CCFG.FRAMERATE bit defaults to 48 kHz. This bit only affects the local clock generation for the I^{2}C interface to match standard I^{2}C clock speeds.	
		0	48 kHz
		1	44.1 kHz

Table 11-39: A2B_I2CCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	EACK	Early Acknowledge (Main Only). When `A2B_I2CCFG.EACK` is set, the I^2C interface provides an acknowledge to writes addressed to a subordinate node before the write has completed on the A^2B bus. If there is an error (for example, a timeout or address error), the `A2B_INTPND2.I2CERR` bit is set. When `A2B_I2CCFG.EACK` is cleared, I^2C transactions are clock-stretched until they are complete in the system so that a correct ACK/NACK can be generated by the I^2C interface. The `A2B_I2CCFG.EACK` bit can be used for I^2C access of a subordinate node. For accesses to peripherals connected to subordinate nodes, the clock stretching feature is required for the I^2C interface of the host.	
		0	Stretch Transactions
		1	Provide Write Acknowledge
0 (R/W)	DATARATE	I2C Data Rate (Sub Only). The `A2B_I2CCFG.DATARATE` bit configures the I^2C data rate.	
		0	100 kHz
		1	400 kHz

I2S Global Configuration Register

The A2B_I2SGCFG register provides bits which control the operation of all I^{2}S units. The A2B_I2SGCFG register must be programmed before the A2B_I2SCFG.TXPINS, A2B_I2SCFG.RXPINS, A2B_PDMCTL.PDM0EN, and A2B_PDMCTL.PDM1EN bits are set.

For the main node, the A2B_I2SGCFG register must be programmed before discovery and not be modified after discovery.

Address: 0x41

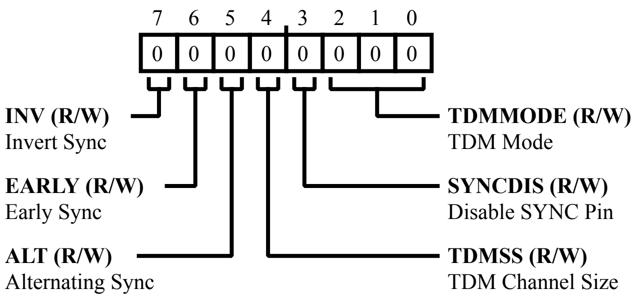


Figure 11-39: A2B_I2SGCFG Register Diagram

Table 11-40: A2B_I2SGCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	INV	Invert Sync. The A2B_I2SGCFG.INV bit determines whether the rising edge or the falling edge of the A2B_SYNC pin corresponds to the start of an audio frame. If the A2B_I2SGCFG.INV bit is to be set in a main node, it must be set before the A2B_CONTROL.MSTR bit is set.	
		0	Rising edge of SYNC pin at start of audio frame
		1	Falling edge of SYNC pin at start of audio frame
6 (R/W)	EARLY	Early Sync. The A2B_I2SGCFG.EARLY bit determines whether the A2B_SYNC pin changes in the same cycle as the MSB of data channel 0 or one cycle before the MSB of data channel 0.	
		0	Change SYNC pin in same cycle
		1	Change SYNC pin in previous cycle (early SYNC)

Table 11-40: A2B_I2SGCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	ALT	Alternating Sync. The `A2B_I2SGCFG.ALT` bit determines whether the `A2B_SYNC` pin is pulsed high for one cycle at the start of each sampling period or driven high during right channel data and low during left channel data for I^2S stereo mode operation.	
		0	Pulse SYNC Pin High for 1 Cycle
		1	Drive SYNC Pin for I^2S Operation
4 (R/W)	TDMSS	TDM Channel Size. The `A2B_I2SGCFG.TDMSS` bit determines whether the TDM channel size is 16 or 32 bits.	
		0	32 bit
		1	16 bit
3 (R/W)	SYNCDIS	Disable SYNC Pin. The `A2B_I2SGCFG.SYNCDIS` bit, when set (=1), disables the toggling of the SYNC signal. This feature can be used to hold the SYNC signal in an inactive state while the BCLK signal runs. The `A2B_I2SGCFG.SYNCDIS` bit can only be set while the I^2S port is disabled. Setting the `A2B_I2SGCFG.SYNCDIS` bit while the I^2S port is enabled does NOT disable the SYNC signal.	
2:0 (R/W)	TDMMODE	TDM Mode. The `A2B_I2SGCFG.TDMMODE` bit field selects the mode for the I^2S/TDM units.	
		0	TDM2
		1	TDM4
		2	TDM8
		3	TDM12 (No sub node support)
		4	TDM16
		5	TDM20 (No sub node support)
		6	TDM24 (No sub node support)
		7	TDM32

I2S Configuration Register

The A2B_I2SCFG register provides individual settings for both I^{2}S receive data signals and both I^{2}S transmit signals.

Address: 0x42

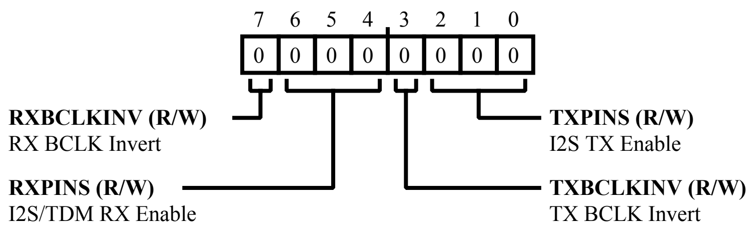


Figure 11-40: A2B_I2SCFG Register Diagram

Table 11-41: A2B_I2SCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXBCLKINV	RX BCLK Invert. For a main node, when A2B_I2SCFG.RXBCLKINV =0, the I^{2}S/TDM receive data pins (DRXn) and A2B_SYNC pin are sampled on the rising edge of BCLK (for stereo I^{2}S). For a main node, when A2B_I2SCFG.RXBCLKINV =1, the I^{2}S/TDM receive data pins (DRXn) and A2B_SYNC pin are sampled on the falling edge of BCLK (for pulsed-SYNC TDM mode). For a subordinate node, when A2B_I2SCFG.RXBCLKINV =0, the I^{2}S/TDM receive data pins (DRXn) are sampled on the rising edge of BCLK. For a subordinate node, when A2B_I2SCFG.RXBCLKINV =1, the I^{2}S/TDM receive data pins (DRXn) are sampled on the falling edge of A2B_BCLK.	
		0	Disabled
		1	Enabled
6:4 (R/W)	RXPINS	I2S/TDM RX Enable.	
		0	No I^{2}S/TDM RX Pins Enabled
		1	One I^{2}S/TDM RX Pin Enabled
		2	Two I^{2}S/TDM RX Pins Enabled
		3	Three I^{2}S/TDM RX Pins Enabled
		4	Four I^{2}S/TDM RX Pins Enabled
		7	Two I^{2}S/TDM RX Pins Enabled with Interleave

Table 11-41: A2B_I2SCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TXBCLKINV	TX BCLK Invert. For a main node, when `A2B_I2SCFG.TXBCLKINV` =0, the I^{2}S/TDM transmit data pins (DTXn) change on the rising edge of `A2B_BCLK`. For a main node, when `A2B_I2SCFG.TXBCLKINV` =1, the I^{2}S/TDM transmit data pins (DTXn) change on the falling edge of `A2B_BCLK`. For a subordinate node, when `A2B_I2SCFG.TXBCLKINV` =0, the I^{2}S/TDM transmit data pins (DTXn) and `A2B_SYNC` pin change on the rising edge of `A2B_BCLK`. For a subordinate node, when `A2B_I2SCFG.TXBCLKINV=1`, the I^{2}S/TDM transmit data pins (DTXn) and `A2B_SYNC` pin change on the falling edge of `A2B_BCLK`.	
		0	Disabled
		1	Enabled
2:0 (R/W)	TXPINS	I2S TX Enable.	
		0	No I^{2}S/TDM TX Pins Enabled
		1	One I^{2}S/TDM TX Pin Enabled
		2	Two I^{2}S/TDM TX Pins Enabled
		3	Three I^{2}S/TDM TX Pins Enabled
		4	Four I^{2}S/TDM TX Pins Enabled
		7	Two I^{2}S/TDM TX Pins Enabled with Interleave

I2S Rate Register (Sub Only)

The A2B_I2SRATE register controls the I^2S/TDM interfaces in subordinate nodes, which may run at a multiple of the superframe rate.

Address: 0x43

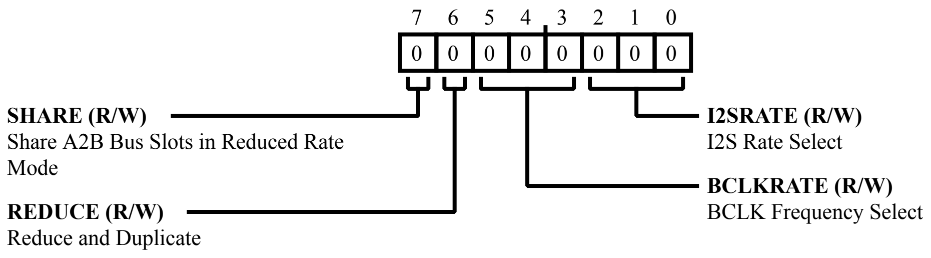


Figure 11-41: A2B_I2SRATE Register Diagram

Table 11-42: A2B_I2SRATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	SHARE	Share A2B Bus Slots in Reduced Rate Mode. The A2B_I2SRATE.SHARE bit function applies only when the local sample rate is lower than the superframe rate. When the A2B_I2SRATE.SHARE bit is set, I^2S/TDM data for the local node is time multiplexed on the A^2B bus. Only A2B_I2SRRSOFFS.RRSOFFSET values of 0 or 1 are supported when the A2B_I2SRATE.SHARE bit is enabled.	
		0	Disabled
		1	Enabled
6 (R/W)	REDUCE	Reduce and Duplicate. The A2B_I2SRATE.REDUCE bit function applies only when the local sample rate is higher than the superframe rate. When the A2B_I2SRATE.REDUCE bit is set, the number of received samples is reduced so that only one sample is used per superframe, and transmitted samples are duplicated so that only one sample is needed per superframe.	
		0	Disabled
		1	Enabled
5:3 (R/W)	BCLKRATE	BCLK Frequency Select. The A2B_I2SRATE.BCLKRATE bit field is used to select an alternate BCLK frequency for a reduced rate subordinate node. The nominal BCLK frequency is determined by the superframe frequency (SFF), settings, and reduced rate divide ratio (from A2B_I2SRRATE.RRDIV and A2B_I2SRATE.I2SRATE).	
		0	BCLK frequency as configured in I2SGCFG
		1	SYNC frequency x 2048

Table 11-42: A2B_I2SRATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	SYNC frequency x 4096
		4	SFF frequency x 64
		5	SFF frequency x 128
		6	SFF frequency x 256
2:0 (R/W)	I2SRATE	I2S Rate Select. The `A2B_I2SRATE.I2SRATE` bit sets the rate for I^2S/TDM transmit and receive operations in the local subordinate node. This sample rate is based on the superframe frequency (SFF is either 48 kHz or 44.1 kHz).	
		0	SFF x 1
		1	SFF / 2
		2	SFF / 4
		3	SFF / `A2B_I2SRRATE.RRDIV`
		5	SFF x 2
		6	SFF x 4

I2S Transmit Data Offset Register (Main Only)

The A2B_I2STXOFFSET register controls the number of I^2S transmit channels which are skipped before the node begins transmitting data. The A2B_I2STXOFFSET register must be programmed before any of the transmit data pin enable bits are set in A2B_I2SCFG register.

Address: 0x44

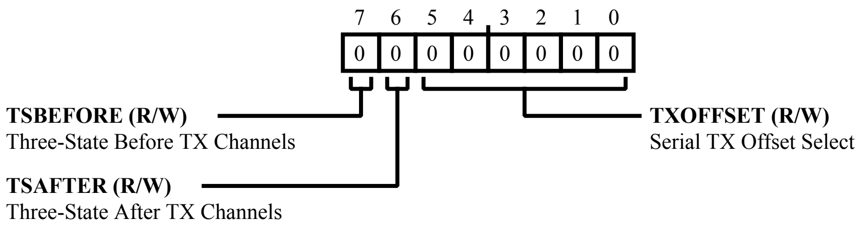


Figure 11-42: A2B_I2STXOFFSET Register Diagram

Table 11-43: A2B_I2STXOFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	TSBEFORE	Three-State Before TX Channels. When the A2B_I2STXOFFSET.TSBEFORE bit is cleared (default), the Transmit Data pins (DTXn) are driven low at the beginning of each frame for the number of data channels defined in A2B_I2STXOFFSET.TXOFFSET. When the A2B_I2STXOFFSET.TSBEFORE bit is set, the Transmit Data pins (DTXn) are instead three-stated for the number of data channels defined in A2B_I2STXOFFSET.TXOFFSET.	
		0	Disable
		1	Enable
6 (R/W)	TSAFTER	Three-State After TX Channels. When the A2B_I2STXOFFSET.TSAFTER bit is cleared (default), the Transmit Data pins (DTXn) are driven low after all valid channels have been transmitted. When the A2B_I2STXOFFSET.TSAFTER bit is set, the Transmit Data pins (DTXn) are instead three-stated after all valid channels have been transmitted.	
		0	Disable
		1	Enable

Table 11-43: A2B_I2STXOFFSET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:0 (R/W)	TXOFFSET	Serial TX Offset Select. The `A2B_I2STXOFFSET.TXOFFSET` bit field defines the number of I^{2}S/TDM channels that are skipped before the node begins transmitting data. The valid values for this field are 0-63.	
		0	No TX offset
		1	1 TDM channel
		62	62 TDM channels
		63	63 TDM channels

SYNC Offset Register (Sub Only)

The A2B_SYNCOFFSET register adjusts the A^2B bus clock ($f_{SYSBCLK}$) cycle count on which the A2B_SYNC pin indicates the start of an audio frame. The offset is an 8-bit signed value, allowing the sync to be moved anywhere between 128 cycles before the start of the superframe to 127 cycles after the start of the superframe. A^2B subordinate nodes can all sample exactly at the same time by individually compensating for their propagation delay with this register setting.

The A2B_SYNCOFFSET register should be programmed before the A2B_I2SCFG.TXPINS, A2B_I2SCFG.RXPINS, A2B_PDMCTL.PDM0EN, and A2B_PDMCTL.PDM1EN bits are set.

Address: 0x46

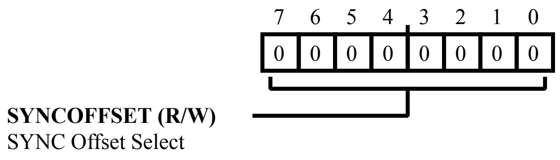


Figure 11-43: A2B_SYNCOFFSET Register Diagram

Table 11-44: A2B_SYNCOFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	SYNCOFFSET	SYNC Offset Select. The A2B_SYNCOFFSET.SYNCOFFSET bit field adjusts the system clock cycle where the A2B_SYNC pin indicates the start of an audio frame. The value programmed to the A2B_SYNCOFFSET.SYNCOFFSET field is the 8-bit signed, two's complement representation of the integer value defining the number of SYSBCLK cycles that lag the SYNC signal before the superframe begins. Valid values for this field range from no SYNC offset (0x00) to the SYNC occurring 127 cycles before the start of the superframe (0x81).	
		0	No offset
		1-128	Reserved
		129	127 SYSBCLK cycles
		130-254	126 to 2 SYSBCLK cycles (respectively)
		255	1 SYSBCLK cycle

PDM Control Register

The A2B_PDMCTL register provides enable bits for the pulse density modulators.

Address: 0x47

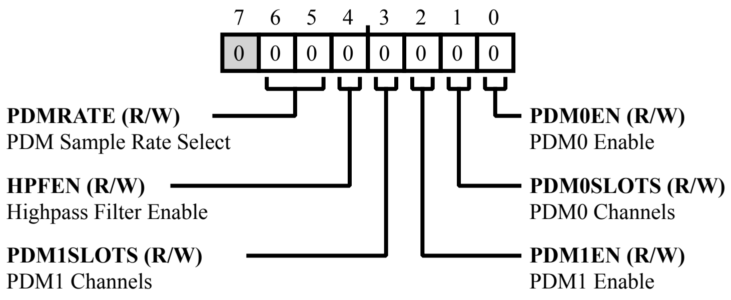


Figure 11-44: A2B_PDMCTL Register Diagram

Table 11-45: A2B_PDMCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	PDMRATE	PDM Sample Rate Select. The A2B_PDMCTL.PDMRATE bit field controls the output rate of the PDM demodulators, which is based off of the superframe rate (SFF). Changes to the A2B_PDMCTL.PDMRATE field do not change the PDM clock frequency. For a subordinate node, setting the node to a reduced rate changes the SYNC and PDM clock frequencies. Setting the subordinate node to an increased rate changes only the SYNC. The PDM clock frequency stays at 3.07MHz.	
		0	SFF
		1	SFF/2
		2	SFF/4
		3	Reserved
4 (R/W)	HPFEN	Highpass Filter Enable. The A2B_PDMCTL.HPFEN bit controls whether or not the high pass filter is used on received PDM data.	
		0	Disabled
		1	Enabled
3 (R/W)	PDM1SLOTS	PDM1 Channels. The A2B_PDMCTL.PDM1SLOTS bit controls whether the PDM signal on the A2B_SIO1 pin is one channel (mono) or two channels (stereo).	
		0	Mono
		1	Stereo

Table 11-45: A2B_PDMCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	PDM1EN	PDM1 Enable. The `A2B_PDMCTL.PDM1EN` bit enables PDM reception on the `A2B_SIO1` pin.	
		0	Disabled
		1	Enabled
1 (R/W)	PDM0SLOTS	PDM0 Channels. The `A2B_PDMCTL.PDM0SLOTS` bit controls whether the PDM signal on the A2B_SIO0 pin is one channel (mono) or two channels (stereo).	
		0	Mono
		1	Stereo
0 (R/W)	PDM0EN	PDM0 Enable. The `A2B_PDMCTL.PDM0EN` bit enables PDM reception on the `A2B_SIO0` pin.	
		0	Disabled
		1	Enabled

Error Management Register

The A2B_ERRMGMT register provides options for reporting communication errors over the I^{2}S/TDM interface.

Address: 0x48

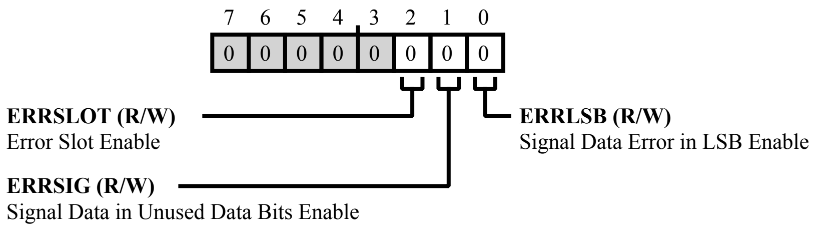


Figure 11-45: A2B_ERRMGMT Register Diagram

Table 11-46: A2B_ERRMGMT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	ERRSLOT	Error Slot Enable. Setting the A2B_ERRMGMT.ERRSLOT bit causes the transceiver to append an extra I^{2}S/TDM data channel to the TDM stream to indicate A^{2}B errors in the received data slots.	
		0	Disabled
		1	Enabled
1 (R/W)	ERRSIG	Signal Data in Unused Data Bits Enable. When the A2B_ERRMGMT.ERRSIG is set, any unused data bits in each I^{2}S/TDM channel indicate data errors.	
		0	Disabled
		1	Enabled
0 (R/W)	ERRLSB	Signal Data Error in LSB Enable. When the A2B_ERRMGMT.ERRLSB bit is set, the LSB of each I^{2}S/TDM sample is replaced with an active-high status bit indicating that there is an error in the data slot (1= error, 0 = no error).	
		0	Disabled
		1	Enabled

GPIO Output Data Register

The A2B_GPIODAT register controls output data for general-purpose I/O pins.

Address: 0x4A

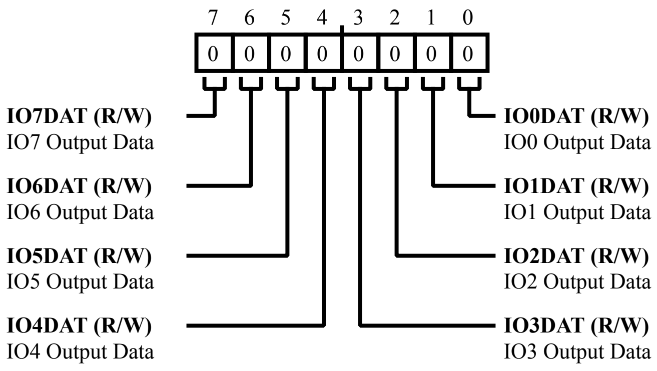


Figure 11-46: A2B_GPIODAT Register Diagram

Table 11-47: A2B_GPIODAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IO7DAT	IO7 Output Data. The value of the `A2B_GPIODAT.IO7DAT` bit is driven onto the IO7 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO7OEN`=1).	
		0	Output Low
		1	Output High
6 (R/W)	IO6DAT	IO6 Output Data. The value of the `A2B_GPIODAT.IO6DAT` bit is driven onto the GPIO6 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO6OEN`=1).	
		0	Output Low
		1	Output High
5 (R/W)	IO5DAT	IO5 Output Data. The value of the `A2B_GPIODAT.IO5DAT` bit is driven onto the GPIO5 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO5OEN`=1).	
		0	Output Low
		1	Output High

Table 11-47: A2B_GPIODAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	IO4DAT	IO4 Output Data. The value of the `A2B_GPIODAT.IO4DAT` bit is driven onto the GPIO4 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO4OEN`=1).Depending on the value of `A2B_PINCFG.GPIOSEL`, GPIO4 can be mapped to ADR1 or ADR2.	
		0	Output Low
		1	Output High
3 (R/W)	IO3DAT	IO3 Output Data. The value of the `A2B_GPIODAT.IO3DAT` bit is driven onto the IO3 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO3OEN`=1).	
		0	Output Low
		1	Output High
2 (R/W)	IO2DAT	IO2 Output Data. The value of the `A2B_GPIODAT.IO2DAT` bit is driven onto the IO2 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO2OEN`=1).	
		0	Output Low
		1	Output High
1 (R/W)	IO1DAT	IO1 Output Data. The value of the `A2B_GPIODAT.IO1DAT` bit is driven onto the IO1 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO1OEN`=1).	
		0	Output Low
		1	Output High
0 (R/W)	IO0DAT	IO0 Output Data. The value of the `A2B_GPIODAT.IO0DAT` bit is driven onto the GPIO0 pin when it is in GPIO mode with its output driver enabled (`A2B_GPIOOEN.IO0OEN`=1). Depending on the value of `A2B_PINCFG.GPIOSEL`, GPIO0 can be mapped to SCK, SIO0, or SIO4.	
		0	Output Low
		1	Output High

GPIO Output Data Set Register

The A2B_GPIODATSET register allows setting of individual GPIO output register bits (write 1 to set) without influencing the states of the other GPIO output register bits. Reads from this address return the value in the GPIO output data (A2B_GPIODAT) register.

Address: 0x4B

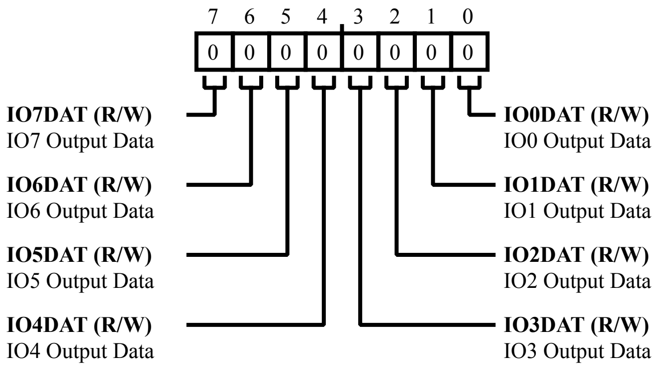


Figure 11-47: A2B_GPIODATSET Register Diagram

Table 11-48: A2B_GPIODATSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1S)	IO7DSET	IO7 Data Set. The A2B_GPIODATSET.IO7DSET bit executes a write-1-to-set action for the A2B_GPIODAT.IO7DAT bit.	
		0	No Action
		1	Set Bit
6 (R/W1S)	IO6DSET	IO6 Data Set. The A2B_GPIODATSET.IO6DSET bit executes a write-1-to-set action for the A2B_GPIODAT.IO6DAT bit.	
		0	No Action
		1	Set Bit
5 (R/W1S)	IO5DSET	IO5 Data Set. The A2B_GPIODATSET.IO5DSET bit executes a write-1-to-set action for the A2B_GPIODAT.IO5DAT bit.	
		0	No Action
		1	Set Bit

Table 11-48: A2B_GPIODATSET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1S)	IO4DSET	IO4 Data Set. The `A2B_GPIODATSET.IO4DSET` bit executes a write-1-to-set action for the `A2B_GPIODAT.IO4DAT` bit.	
		0	No Action
		1	Set Bit
3 (R/W1S)	IO3DSET	IO3 Data Set. The `A2B_GPIODATSET.IO3DSET` bit executes a write-1-to-set action for the `A2B_GPIODAT.IO3DAT` bit.	
		0	No Action
		1	Set Bit
2 (R/W1S)	IO2DSET	IO2 Data Set. The `A2B_GPIODATSET.IO2DSET` bit executes a write-1-to-set action for the `A2B_GPIODAT.IO2DAT` bit.	
		0	No Action
		1	Set Bit
1 (R/W1S)	IO1DSET	IO1 Data Set. The `A2B_GPIODATSET.IO1DSET` bit executes a write-1-to-set action for the `A2B_GPIODAT.IO1DAT` bit.	
		0	No Action
		1	Set Bit
0 (R/W1S)	IO0DSET	IO0 Data Set. The `A2B_GPIODATSET.IO0DSET` bit executes a write-1-to-set action for the `A2B_GPIODAT.IO0DAT` bit.	
		0	No Action
		1	Set Bit

GPIO Output Data Clear Register

The A2B_GPIODATCLR register allows clearing of individual GPIO output register bits to 0 (write 1 to clear) without influencing the states of the other GPIO output register bits. Reads from this address return the value in the GPIO output data (A2B_GPIODAT) register.

Address: 0x4C

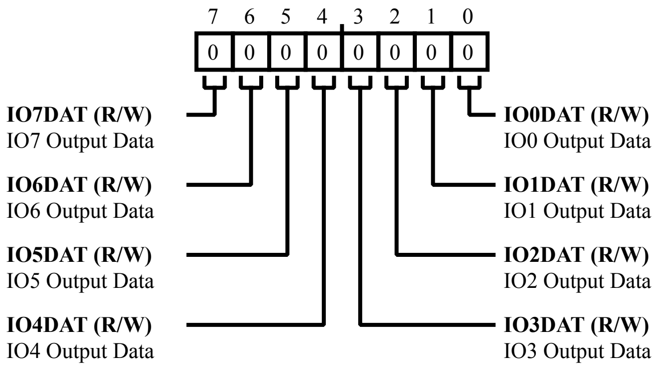


Figure 11-48: A2B_GPIODATCLR Register Diagram

Table 11-49: A2B_GPIODATCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	IO7DCLR	IO7 Data Clear. The `A2B_GPIODATCLR.IO7DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO7DAT` bit.	
		0	No Action
		1	Clear Bit
6 (R/W1C)	IO6DCLR	IO6 Data Clear. The `A2B_GPIODATCLR.IO6DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO6DAT` bit.	
		0	No Action
		1	Clear Bit
5 (R/W1C)	IO5DCLR	IO5 Data Clear. The `A2B_GPIODATCLR.IO5DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO5DAT` bit.	
		0	No Action
		1	Clear Bit

Table 11-49: A2B_GPIODATCLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	IO4DCLR	IO4 Data Clear. The `A2B_GPIODATCLR.IO4DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO4DAT` bit.	
		0	No Action
		1	Clear Bit
3 (R/W1C)	IO3DCLR	IO3 Data Clear. The `A2B_GPIODATCLR.IO3DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO3DAT` bit.	
		0	No Action
		1	Clear Bit
2 (R/W1C)	IO2DCLR	IO2 Data Clear. The `A2B_GPIODATCLR.IO2DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO2DAT` bit.	
		0	No Action
		1	Clear Bit
1 (R/W1C)	IO1DCLR	IO1 Data Clear. The `A2B_GPIODATCLR.IO1DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO1DAT` bit.	
		0	No Action
		1	Clear Bit
0 (R/W1C)	IO0DCLR	IO0 Data Clear. The `A2B_GPIODATCLR.IO0DCLR` bit executes a write-1-to-clear action for the `A2B_GPIODAT.IO0DAT` bit.	
		0	No Action
		1	Clear Bit

GPIO Output Enable Register

The A2B_GPIOOEN register controls the output enables of the general-purpose I/O pins.

Address: 0x4D

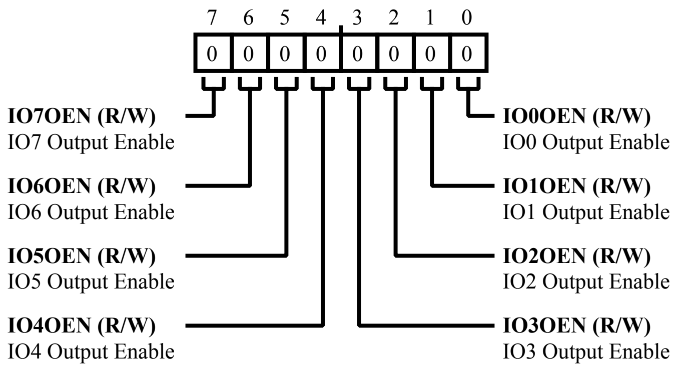


Figure 11-49: A2B_GPIOOEN Register Diagram

Table 11-50: A2B_GPIOOEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IO7OEN	IO7 Output Enable. The `A2B_GPIOOEN.IO7OEN` bit configures the IO7 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
6 (R/W)	IO6OEN	IO6 Output Enable. The `A2B_GPIOOEN.IO6OEN` bit configures the IO6 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
5 (R/W)	IO5OEN	IO5 Output Enable. The `A2B_GPIOOEN.IO5OEN` bit configures the IO5 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable

Table 11-50: A2B_GPIOOEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	IO4OEN	IO4 Output Enable. The `A2B_GPIOOEN.IO4OEN` bit configures the IO4 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
3 (R/W)	IO3OEN	IO3 Output Enable. The `A2B_GPIOOEN.IO3OEN` bit configures the IO3 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
2 (R/W)	IO2OEN	IO2 Output Enable. The `A2B_GPIOOEN.IO2OEN` bit configures the IO2 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
1 (R/W)	IO1OEN	IO1 Output Enable. The `A2B_GPIOOEN.IO1OEN` bit configures the IO1 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable
0 (R/W)	IO0OEN	IO0 Output Enable. The `A2B_GPIOOEN.IO0OEN` bit configures the IO0 pin as an output when the pin is in GPIO mode.	
		0	Disable
		1	Enable

GPIO Input Enable Register

The A2B_GPIOIEN register controls the input enables of the general purpose I/O pins.

Address: 0x4E

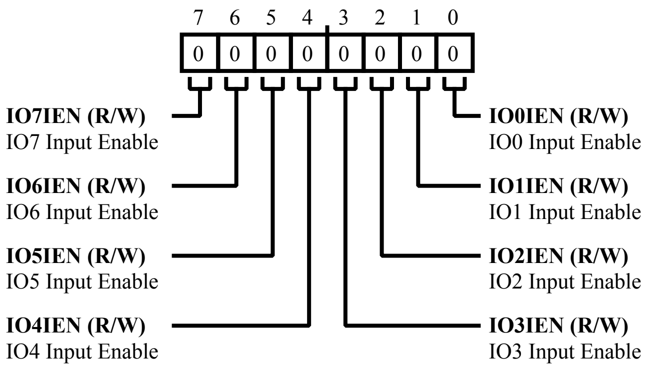


Figure 11-50: A2B_GPIOIEN Register Diagram

Table 11-51: A2B_GPIOIEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IO7IEN	IO7 Input Enable. The A2B_GPIOIEN.IO7IEN bit is the input enable for the IO7 pin.	
		0	Disable
		1	Enable
6 (R/W)	IO6IEN	IO6 Input Enable. The A2B_GPIOIEN.IO6IEN bit is the input enable for the IO6 pin.	
		0	Disable
		1	Enable
5 (R/W)	IO5IEN	IO5 Input Enable. The A2B_GPIOIEN.IO5IEN bit is the input enable for the IO5 pin.	
		0	Disable
		1	Enable
4 (R/W)	IO4IEN	IO4 Input Enable. The A2B_GPIOIEN.IO4IEN bit is the input enable for the IO4 pin.	
		0	Disable
		1	Enable

Table 11-51: A2B_GPIOIEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	IO3IEN	IO3 Input Enable. The `A2B_GPIOIEN.IO3IEN` bit is the input enable for the IO3 pin.	
		0	Disable
		1	Enable
2 (R/W)	IO2IEN	IO2 Input Enable. The `A2B_GPIOIEN.IO2IEN` bit is the input enable for the IO2 pin.	
		0	Disable
		1	Enable
1 (R/W)	IO1IEN	IO1 Input Enable. The `A2B_GPIOIEN.IO1IEN` bit is the input enable for the IO1 pin.	
		0	Disable
		1	Enable
0 (R/W)	IO0IEN	IO0 Input Enable. The `A2B_GPIOIEN.IO0IEN` bit is the input enable for the IO0 pin.	
		0	Disable
		1	Enable

GPIO Input Value Register

The A2B_GPIOIN register returns the value of enabled general-purpose I/O input pins.

Address: 0x4F

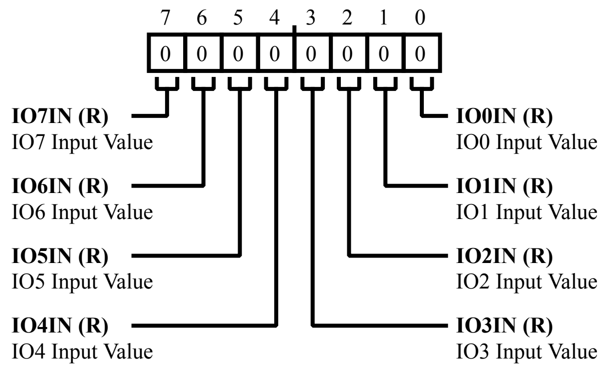


Figure 11-51: A2B_GPIOIN Register Diagram

Table 11-52: A2B_GPIOIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	IO7IN	IO7 Input Value. The A2B_GPIOIN.IO7IN bit contains the value of the IO7 pin when in input GPIO mode (A2B_GPIOIEN.IO7IEN=1). Otherwise, the bit is low.
6 (R/NW)	IO6IN	IO6 Input Value. The A2B_GPIOIN.IO6IN bit contains the value of the IO6 pin when in input GPIO mode (A2B_GPIOIEN.IO6IEN=1). Otherwise, the bit is low.
5 (R/NW)	IO5IN	IO5 Input Value. The A2B_GPIOIN.IO5IN bit contains the value of the IO5 pin when in input GPIO mode (A2B_GPIOIEN.IO5IEN=1). Otherwise, the bit is low.
4 (R/NW)	IO4IN	IO4 Input Value. The A2B_GPIOIN.IO4IN bit contains the value of the IO4 pin when in input GPIO mode (A2B_GPIOIEN.IO4IEN=1). Otherwise, the bit is low.
3 (R/NW)	IO3IN	IO3 Input Value. The A2B_GPIOIN.IO3IN bit contains the value of the IO3 pin when in input GPIO mode (A2B_GPIOIEN.IO3IEN=1). Otherwise, the bit is low.
2 (R/NW)	IO2IN	IO2 Input Value. The A2B_GPIOIN.IO2IN bit contains the value of the IO2 pin when in input GPIO mode (A2B_GPIOIEN.IO2IEN=1). Otherwise, the bit is low.

Table 11-52: A2B_GPIOIN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	IO1IN	IO1 Input Value. The `A2B_GPIOIN.IO1IN` bit contains the value of the IO1 pin when the `A2B_GPIOIEN.IO1IEN` bit is high. Otherwise the bit is low.
0 (R/NW)	IO0IN	IO0 Input Value. The `A2B_GPIOIN.IO0IN` bit contains the value of the IO0 pin when in input GPIO mode (`A2B_GPIOIEN.IO0IEN=1`). Otherwise, the bit is low.

Pin Interrupt Enable Register

The A2B_PINTEN register enables input-enabled GPIO pins to generate an interrupt.

Address: 0x50

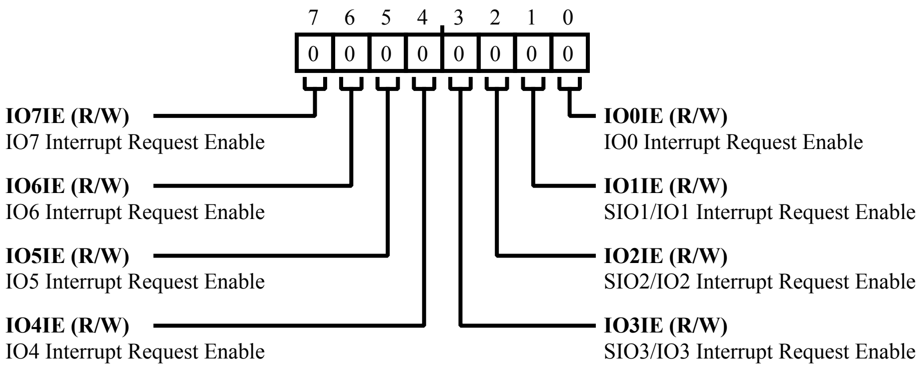

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
IO7IE (R/W)
IO7 Interrupt Request Enable
IO6IE (R/W)
IO6 Interrupt Request Enable
IO5IE (R/W)
IO5 Interrupt Request Enable
IO4IE (R/W)
IO4 Interrupt Request Enable
IO0IE (R/W)
IO0 Interrupt Request Enable
IO1IE (R/W)
SIO1/IO1 Interrupt Request Enable
IO2IE (R/W)
SIO2/IO2 Interrupt Request Enable
IO3IE (R/W)
SIO3/IO3 Interrupt Request Enable

Figure 11-52: A2B_PINTEN Register Diagram

Table 11-53: A2B_PINTEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IO7IE	IO7 Interrupt Request Enable. The A2B_PINTEN.IO7IE bit enables the IO7 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
6 (R/W)	IO6IE	IO6 Interrupt Request Enable. The A2B_PINTEN.IO6IE bit enables the IO6 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
5 (R/W)	IO5IE	IO5 Interrupt Request Enable. The A2B_PINTEN.IO5IE bit enables the IO5 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled

Table 11-53: A2B_PINTEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	IO4IE	IO4 Interrupt Request Enable. The `A2B_PINTEN.IO4IE` bit enables the IO4 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
3 (R/W)	IO3IE	SIO3/IO3 Interrupt Request Enable. The `A2B_PINTEN.IO3IE` bit enables the IO3 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
2 (R/W)	IO2IE	SIO2/IO2 Interrupt Request Enable. The `A2B_PINTEN.IO2IE` bit enables the IO2 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
1 (R/W)	IO1IE	SIO1/IO1 Interrupt Request Enable. The `A2B_PINTEN.IO1IE` bit enables bit enables the IO1 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled
0 (R/W)	IO0IE	IO0 Interrupt Request Enable. The `A2B_PINTEN.IO0IE` bit enables the IO0 input to generate an interrupt request when a rising edge is sensed.	
		0	Disabled
		1	Enabled

Pin Interrupt Invert Register

The A2B_PINTINV register is used to invert pin inputs in the path to interrupt generation.

Address: 0x51

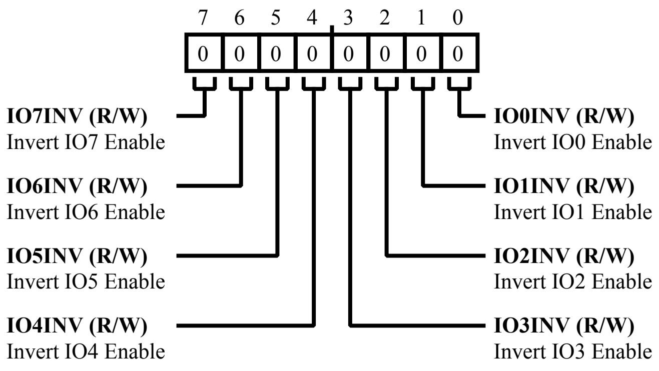


Figure 11-53: A2B_PINTINV Register Diagram

Table 11-54: A2B_PINTINV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IO7INV	Invert IO7 Enable. Setting the A2B_PINTINV.IO7INV bit inverts the polarity of the IO7 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
6 (R/W)	IO6INV	Invert IO6 Enable. Setting the A2B_PINTINV.IO6INV bit inverts the polarity of the IO6 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
5 (R/W)	IO5INV	Invert IO5 Enable. Setting the A2B_PINTINV.IO5INV bit inverts the polarity of the IO5 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled

Table 11-54: A2B_PINTINV Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	IO4INV	Invert IO4 Enable. Setting the `A2B_PINTINV.IO4INV` bit inverts the polarity of the IO4 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
3 (R/W)	IO3INV	Invert IO3 Enable. Setting the `A2B_PINTINV.IO3INV` bit inverts the polarity of the IO3 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
2 (R/W)	IO2INV	Invert IO2 Enable. Setting the `A2B_PINTINV.IO2INV` bit inverts the polarity of the IO2 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
1 (R/W)	IO1INV	Invert IO1 Enable. Setting the `A2B_PINTINV.IO1INV` bit inverts the polarity of the IO1 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled
0 (R/W)	IO0INV	Invert IO0 Enable. Setting the `A2B_PINTINV.IO0INV` bit inverts the polarity of the IO0 pin interrupt request input such that a falling edge sensed on the pin generates the interrupt rather than the rising edge (default).	
		0	Disabled
		1	Enabled

Pin Configuration Register

The A2B_PINCFG register configures various digital pin characteristics.

Address: 0x52

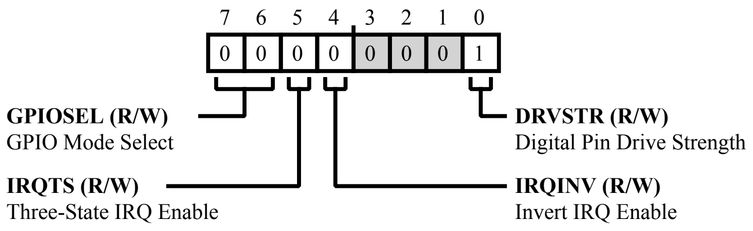


Figure 11-54: A2B_PINCFG Register Diagram

Table 11-55: A2B_PINCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/W)	GPIOSEL	GPIO Mode Select.	
		0	GPIO on SPI Pins
		2	GPIO on I2C, IO0 on SIO0
		3	GPIO on I2C, IO0 on SIO4
5 (R/W)	IRQTS	Three-State IRQ Enable. When the A2B_PINCFG.IRQTS bit is cleared (default), the IRQ pin is always actively driven. Setting the A2B_PINCFG.IRQTS bit causes the transceiver to drive the IRQ pin when the interrupt is active and to three-state the IRQ pin when inactive.	
		0	Disabled
		1	Enabled
4 (R/W)	IRQINV	Invert IRQ Enable. When the A2B_PINCFG.IRQINV bit is cleared (default), the IRQ pin is active high. Setting the A2B_PINCFG.IRQINV bit makes the IRQ pin active low.	
		0	Disabled
		1	Enabled
0 (R/W)	DRVSTR	Digital Pin Drive Strength. The A2B_PINCFG.DRVSTR bit controls the drive strength of non-I^{2}C digital output pins. The A2B_SCL and A2B_SDA pins always have a high drive strength.	
		0	Low Drive Strength
		1	High Drive Strength

I2S Test Register

The A2B_I2STEST register enables a test mode to verify and debug the I^{2}S/TDM interface.

Address: 0x53

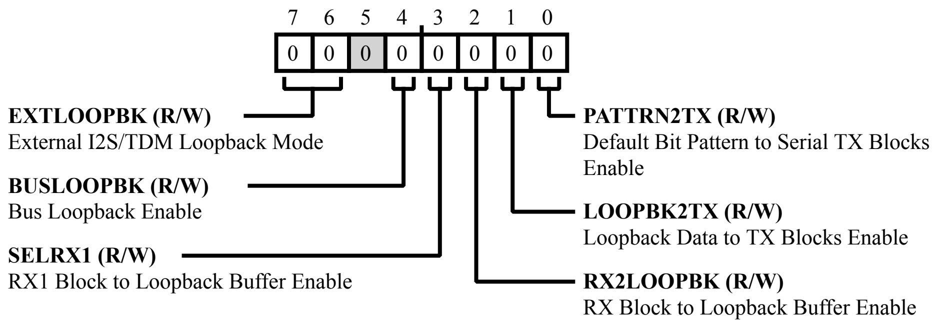

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
EXTLOOPBK (R/W)
External I2S/TDM Loopback Mode
BUSLOOPBK (R/W)
Bus Loopback Enable
SELRX1 (R/W)
RX1 Block to Loopback Buffer Enable
PATTRN2TX (R/W)
Default Bit Pattern to Serial TX Blocks Enable
LOOPBK2TX (R/W)
Loopback Data to TX Blocks Enable
RX2LOOPBK (R/W)
RX Block to Loopback Buffer Enable

Figure 11-55: A2B_I2STEST Register Diagram

Table 11-56: A2B_I2STEST Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/W)	EXTLOOPBK	External I2S/TDM Loopback Mode. The A2B_I2STEST.EXTLOOPBK bit field controls external loopback modes, where the value of an I^{2}S/TDM RX pin is driven directly onto an I^{2}S/TDM TX pin. This loopback may be applied even if the PLL is not locked. When the A2B_I2STEST.EXTLOOPBK bit field is non-zero, the values of A2B_I2STEST.PATTRN2TX, A2B_I2STEST.LOOPBK2TX, A2B_I2STEST.RX2LOOPBK, and A2B_I2STEST.SELRX1 are ignored.	
		0	External Loopback Disabled
		1	External Loopback SIO0 to SIO3
		2	External Loopback SIO0 to SIO3 and SIO1 to SIO4
4 (R/W)	BUSLOOPBK	Bus Loopback Enable. The A2B_I2STEST.BUSLOOPBK bit enables data loop back from the A2B_SIO4 pin to the A2B_SIO0 pin and from the A2B_SIO3 pin to the A2B_SIO1 pin. The A2B_I2STEST.LOOPBK2TX, A2B_I2STEST.RX2LOOPBK, and A2B_I2STEST.SELRX1 are ignored when this bit is set.	
		0	Disabled
		1	Enabled

Table 11-56: A2B_I2STEST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	SELRX1	RX1 Block to Loopback Buffer Enable. When the `A2B_I2STEST.SELRX1` bit is cleared (default), the RX0 block is used for the loopback test when the `A2B_I2STEST.RX2LOOPBK` bit is set.When the `A2B_I2STEST.SELRX1` bit is set, data from the SIO1 block is used instead.	
		0	Disabled
		1	Enabled
2 (R/W)	RX2LOOPBK	RX Block to Loopback Buffer Enable. When the `A2B_I2STEST.RX2LOOPBK` bit is set, the receive bit pattern on either the `A2B_SIO0` or `A2B_SIO1` pins (as controlled by the `A2B_I2STEST.SELRX1` bit) is stored in the TX frame buffer. The `A2B_I2SCFG.RXPINS` bit field is ignored when this bit is set. The RXMASKn registers should contain default values while this bit is set.	
		0	Disabled
		1	Enabled
1 (R/W)	LOOPBK2TX	Loopback Data to TX Blocks Enable. When the `A2B_I2STEST.LOOPBK2TX` bit is set, data received on the `A2B_SIO0` or `A2B_SIO1` pin (as controlled by the `A2B_I2STEST.SELRX1` bit) is sent out on the `A2B_SIO4` and `A2B_SIO3` pins. If the `A2B_I2STEST.RX2LOOPBK` bit is not set when the `A2B_I2STEST.LOOPBK2TX` bit is set, the default bit pattern is sent in all channels. If the `A2B_I2STEST.RX2LOOPBK` bit is cleared while the `A2B_I2STEST.LOOPBK2TX` bit is set, the last received frame is repeated. The value of `A2B_I2SCFG.TXPINS` is ignored when this bit is set.	
		0	Disabled
		1	Enabled
0 (R/W)	PATTRN2TX	Default Bit Pattern to Serial TX Blocks Enable. When the `A2B_I2STEST.PATTRN2TX` bit is set, a default bit pattern (up to 32 bits) is sent in all channels on the `A2B_SIO4` and `A2B_SIO3` pins. The TX0 and TX1 blocks are both enabled if this bit is set. The values of the `A2B_I2SCFG.TXPINS` bit field is not used if this bit is set. This bit is ignored if the `A2B_I2STEST.LOOPBK2TX` bit is set.	
		0	Disabled
		1	Enabled

Raise Interrupt Register

The `A2B_RAISE` register allows the host to generate an interrupt in any node in the system through software. This register must be written over the A^{2}B bus, as writes to this register from the local I^{2}C port have no effect.

Address: 0x54

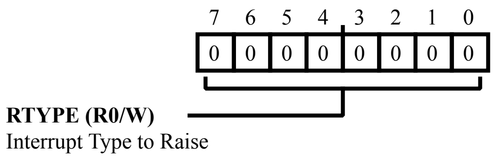


Figure 11-56: A2B_RAISE Register Diagram

Table 11-57: A2B_RAISE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R0/W)	RTYPE	Interrupt Type to Raise. The `A2B_RAISE.RTYPE` bit field is written with the type of interrupt to raise. Any valid interrupt type may be generated in any node in the system. If the RTYPE field does not match a valid interrupt type for the node being written, no action will be taken.	
		0	HDCNTERR
		1	DDERR
		2	CRCERR
		3	DPERR
		4	BECOVF
		5	SRFMISSERR
		6	SRFCRCERR
		9	PWRERR - Shorted to GND
		10	PWRERR - Shorted to VBUS
		11	PWRERR - Short of Wires
		14	PWRERR - Cable is Disconnected (Open Circuit) or Reverse Connected or Wrong Port
		15	PWRERR - Indeterminate Fault
		16	IO0PND
		17	IO1PND
		18	IO2PND
		19	IO3PND

Table 11-57: A2B_RAISE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		20	IO4PND
		21	IO5PND
		22	IO6PND
		23	IO7PND
		24	DSCDONE - Main Only
		25	I2CERR - Main Only
		26	ICRCERR - Main Only
		41	PWRERR - Shorted to GND
		42	PWRERR - Shorted to VBUS
		47	Reserved
		48	Mailbox 0 Full (Sub Only)
		49	Mailbox 0 Empty (Sub Only)
		50	Mailbox 1 Full (Sub Only)
		51	Mailbox 1 Empty (Sub Only)
		64	SPI Done
		65	SPI Remote Register Access Error (Main Only)
		66	SPI Remote I2C Access Error (Main Only)
		67	SPI Data Tunnel Access Error
		68	SPI Bad Command
		69	SPI FIFO Overflow
		70	SPI FIFO Underflow
		80	VMTR Interrupt
		252	Startup Error - Return to Factory
		253	Slave INTTYPE Read Error - Main Only
		254	Standby Done - Main Only
		255	MSTR_RUNNING - Main Only

Generate Bus Error

The A2B_GENERR register allows the host to generate bus errors from any node in the system through software. This register must be written over the A^{2}B bus, as writes to this register from the local I^{2}C port have no effect.

Address: 0x55

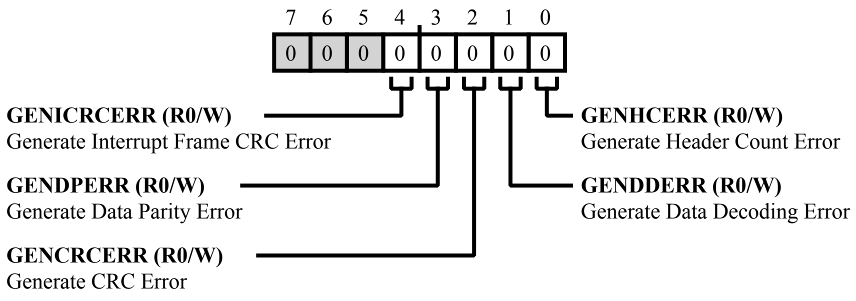


Figure 11-57: A2B_GENERR Register Diagram

Table 11-58: A2B_GENERR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/W)	GENICRCERR	Generate Interrupt Frame CRC Error. A write of 1 to the A2B_GENERR.GENICRCERR bit instructs a subordinate node to generate an interrupt frame CRC error on the A^{2}B bus. A write of 1 to this bit in a main node has no effect.	
		0	No Action
		1	Generate Error
3 (R0/W)	GENDPERR	Generate Data Parity Error. A write of 1 to the A2B_GENERR.GENDPERR bit instructs the node to generate a data parity error on the A^{2}B bus.	
		0	No Action
		1	Generate Error
2 (R0/W)	GENCRCERR	Generate CRC Error. A write of 1 to the A2B_GENERR.GENCRCERR bit instructs the node to generate a CRC error on the A^{2}B bus.	
		0	No Action
		1	Generate Error

Table 11-58: A2B_GENERR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R0/W)	GENDDERR	Generate Data Decoding Error. A write of 1 to the `A2B_GENERR.GENDDERR` bit instructs the node to generate a Data Decode Error on the A^{2}B bus.	
		0	No Action
		1	Generate Error
0 (R0/W)	GENHCERR	Generate Header Count Error. A write of 1 to the `A2B_GENERR.GENHCERR` bit instructs the node to generate a header count error on the A^{2}B bus.	
		0	No Action
		1	Generate Error

I2S Reduced Rate Register (Main Only, Auto-Broadcast)

The A2B_I2SRRATE register provides a means of reducing the A^2B bus data rate by delivering data on a subset of superframes rather than on each superframe, thereby reducing the overall bus power.

When the A2B_I2SRRATE register is written in the main node, the new setting is automatically broadcast over the A^2B bus to all discovered subordinate nodes. Local host writes to this register in a subordinate node have no effect.

Address: 0x56

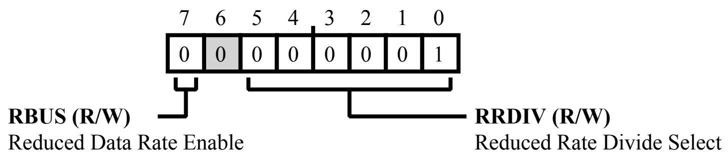


Figure 11-58: A2B_I2SRRATE Register Diagram

Table 11-59: A2B_I2SRRATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RBUS	Reduced Data Rate Enable. When the A2B_I2SRRATE.RBUS bit is set, the bus is configured for reduced-rate data slots where downstream data and upstream data are only delivered once every A2B_I2SRRATE.RRDIV superframes.	
		0	Disabled
		1	Enabled
5:0 (R/W)	RRDIV	Reduced Rate Divide Select. The A2B_I2SRRATE.RRDIV bit field configures the superframe rate at which the I^2S/TDM data is active on the bus. For example, when A2B_I2SRRATE.RRDIV =16, I^2S/TDM data is active every 16th superframe rather than every superframe. Valid settings for this field are only those listed.	
		1	Superframe frequency (SFF)
		2	SFF/2
		4	SFF/4
		8	SFF/8
		12	SFF/12
		16	SFF/16
		20	SFF/20
		24	SFF/24
		28	SFF/28

Table 11-59: A2B_I2SRRATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		32	SFF/32

I2S Reduced Rate Control Register

The A2B_I2SRRCTL register provides bits for controlling the I^{2}S reduced rate strobe.

Address: 0x57

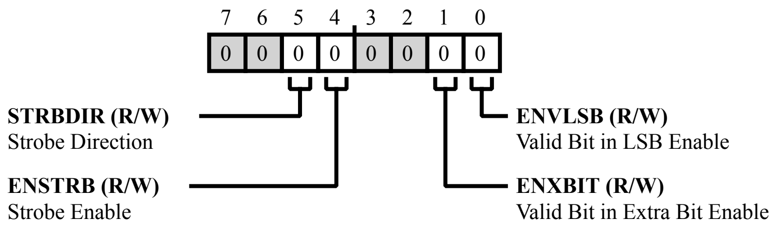


Figure 11-59: A2B_I2SRRCTL Register Diagram

Table 11-60: A2B_I2SRRCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	STRBDIR	Strobe Direction. When the strobe signal is configured as an input, it influences the timing of frames on the bus. For a divide-by-N reduced rate, the strobe must be high once every N frames.	
		0	Input
		1	Output
4 (R/W)	ENSTRB	Strobe Enable. When the A2B_I2SRRCTL.ENSTRB bit is set, the IO7 pin is used as a strobe, indicating the audio frame where the reduced-rate data is updated.	
1 (R/W)	ENXBIT	Valid Bit in Extra Bit Enable. The A2B_I2SRRCTL.ENXBIT bit is only meaningful in a full-rate subordinate node that is receiving reduced rate data from the bus. It does not affect data going over the bus. When the A2B_I2SRRCTL.ENXBIT bit is set, the bit after the LSB in each I^{2}S/TDM channel is high for the superframe with new data and low for the other superframes.	
0 (R/W)	ENVLSB	Valid Bit in LSB Enable. If the A2B_I2SRRCTL.ENVLSB bit is set in a reduced-rate subordinate node, the LSB of the data field is high for a new piece of data and low for a repeated piece of data. The A2B_I2SRRCTL.ENVLSB bit is applicable in the subordinate node only. If the reduced-rate subordinate node sets A2B_I2SRRCTL.ENVLSB and the receiving main node's A2B_I2SRRCTL.ENXBIT bit is set, the output of the TDM data channel is xxxx11 for the first sampled word and xxxx00 for any repeated samples. Additionally, if the A2B_I2SRATE.SHARE bit is set in the reduced-rate subordinate node, the LSB (additional bit) is high for the first data sample and low for the other samples.	

I2S Reduced Rate SYNC Offset Register (Sub Only)

The A2B_I2SRRSOFFS register controls the SYNC offset for subordinate nodes.

Address: 0x58

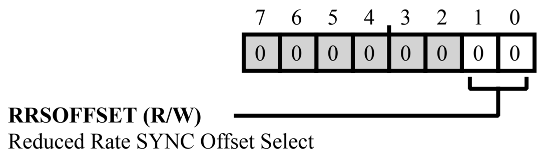


Figure 11-60: A2B_I2SRRSOFFS Register Diagram

Table 11-61: A2B_I2SRRSOFFS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	RRSOFFSET	Reduced Rate SYNC Offset Select. A write of N to the A2B_I2SRRSOFFS.RRSOFFSET bit field instructs a subordinate node, using a reduced I^2S/TDM rate, to offset the SYNC edge to the left by N superframes. When the reduced-rate subordinate node's A2B_I2SRATE.SHARE bit is set, this field can only be programmed to 0 or 1.	
		0	No Offset
		1	1 Superframe Earlier
		2	2 Superframes Earlier
		3	3 Superframes Earlier

CLKOUT1 Configuration Register

The A2B_CLK1CFG register enables an output clock on the A2B_ADR1 pin and sets its frequency.

Address: 0x59

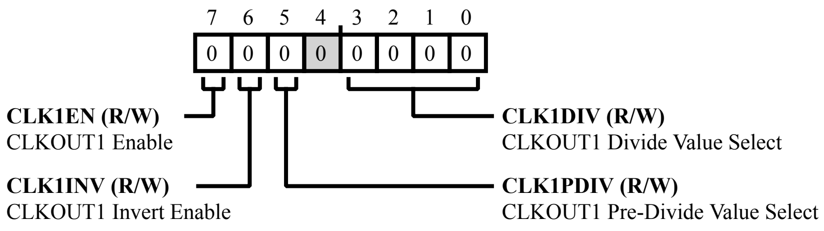


Figure 11-61: A2B_CLK1CFG Register Diagram

Table 11-62: A2B_CLK1CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	CLK1EN	CLKOUT1 Enable. When the A2B_CLK1CFG.CLK1EN bit is set, the ADR1 pin is configured as a clock output, and GPIO functionality for the ADR1 pin is disabled.	
		0	Disabled
		1	Enabled
6 (R/W)	CLK1INV	CLKOUT1 Invert Enable. When the A2B_CLK1CFG.CLK1INV bit is set, the clock output to the ADR1 pin is inverted (moved 180 degrees out of phase) with respect to CLKOUT2. This feature helps to generate CLKOUT clocks with the same frequency but opposite polarity on ADR1 and ADR2.	
		0	Disabled
		1	Enabled
5 (R/W)	CLK1PDIV	CLKOUT1 Pre-Divide Value Select. The A2B_CLK1CFG.CLK1PDIV bit selects a pre-divide of either 2 or 32 from the PLL clock. At a 48 kHz sample frequency, the PLL clock has a frequency of 98.304 MHz. The PLL clock is 2048 times the sample frequency.	
		0	Pre-divide is 2
		1	Pre-divide is 32
3:0 (R/W)	CLK1DIV	CLKOUT1 Divide Value Select. The A2B_CLK1CFG.CLK1DIV bit field selects a divisor between 2 and 32 that is applied to the pre-divided clock before going to the pin. The divide ratio is 2 * (CLK1DIV + 1).	

CLKOUT2 Configuration Register

The A2B_CLK2CFG register enables an output clock on the A2B_ADR2 pin and sets its frequency.

Address: 0x5A

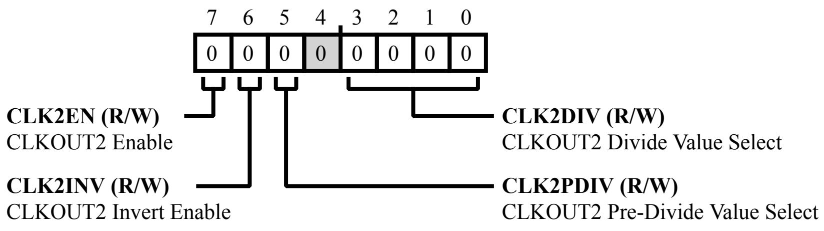


Figure 11-62: A2B_CLK2CFG Register Diagram

Table 11-63: A2B_CLK2CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	CLK2EN	CLKOUT2 Enable. When the A2B_CLK2CFG.CLK2EN bit is set, the ADR2 pin is configured as a clock output, and GPIO functionality for the ADR2 pin is disabled.	
		0	Disabled
		1	Enabled
6 (R/W)	CLK2INV	CLKOUT2 Invert Enable. When the A2B_CLK2CFG.CLK2INV bit is set, the clock output to the ADR2 pin is inverted (moved 180 degrees out of phase).	
		0	Disabled
		1	Enabled
5 (R/W)	CLK2PDIV	CLKOUT2 Pre-Divide Value Select. The A2B_CLK2CFG.CLK2PDIV bit selects a pre-divide of either 2 or 32 from the PLL clock.	
		0	Pre-Divide is 2
		1	Pre-Divide is 32
3:0 (R/W)	CLK2DIV	CLKOUT2 Divide Value Select. The A2B_CLK2CFG.CLK2DIV bit field selects a divisor between 2 and 32 that is applied to the pre-divided clock before going to the pin. The divide ratio is 2 * (CLK2DIV + 1).	

Bus Monitor Mode Configuration Register

The A2B_BMMCFG register configures settings for bus monitor mode.

Address: 0x5B

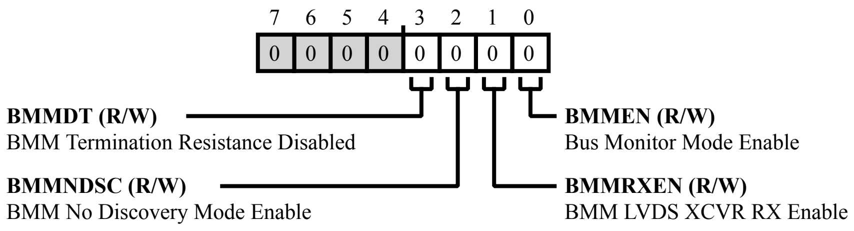


Figure 11-63: A2B_BMMCFG Register Diagram

Table 11-64: A2B_BMMCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	BMMDT	BMM Termination Resistance Disabled. The A2B_BMMCFG.BMMDT bit is used to disable LVDS Termination Resistance in Bus Monitor Mode.	
		0	BMM LVDS Termination Resistance Enabled
		1	BMM LVDS Termination Resistance Disabled
2 (R/W)	BMMNDSC	BMM No Discovery Mode Enable. The A2B_BMMCFG.BMMNDSC bit is used to enable No Discovery Mode in Bus Monitor Mode.	
		0	Disabled
		1	Enabled
1 (R/W)	BMMRXEN	BMM LVDS XCVR RX Enable. The A2B_BMMCFG.BMMRXEN bit is used to enable LVDS RX in Bus Monitor Mode.	
		0	Disabled
		1	Enabled
0 (R/W)	BMMEN	Bus Monitor Mode Enable. The A2B_BMMCFG.BMMEN bit is used to enable Bus Monitor Mode.	
		0	Disabled
		1	Enabled

Sustain Configuration Register (Sub Only)

The A2B_SUSCFG register is used to configure sustain functionality in a subordinate node.

Address: 0x5C

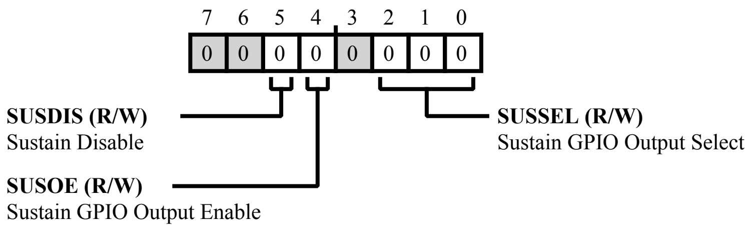


Figure 11-64: A2B_SUSCFG Register Diagram

Table 11-65: A2B_SUSCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SUSDIS	Sustain Disable.	
		0	Enabled
		1	Disabled
4 (R/W)	SUSOE	Sustain GPIO Output Enable.	
		0	Disabled
		1	Enabled
2:0 (R/W)	SUSSEL	Sustain GPIO Output Select.	
		0	Sustain output on IO0
		1	Sustain output on IO1
		2	Sustain output on IO2
		3	Sustain output on IO3
		4	Sustain output on IO4
		5	Sustain output on IO5
		6	Sustain output on IO6
		7	Sustain output on IO7

PDM Control 2 Register

The A2B_PDMCTL2 register provides a means of routing and handling PDM clock and data signals differently to accommodate various PDM configurations.

Address: 0x5D

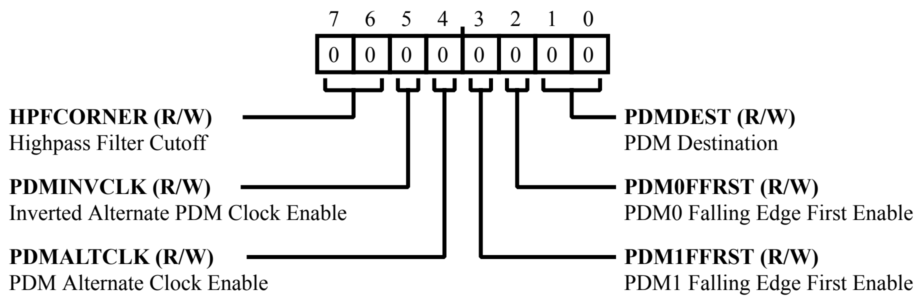


Figure 11-65: A2B_PDMCTL2 Register Diagram

Table 11-66: A2B_PDMCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/W)	HPFCORNER	Highpass Filter Cutoff. The A2B_PDMCTL2.HPFCORNER bit field controls the corner frequency of a high pass filter which can be applied to received PDM data.	
		0	HPF Corner Frequency 1 Hz (Default)
		1	HPF Corner Frequency 60 Hz
		2	HPF Corner Frequency 120 Hz
		3	HPF Corner Frequency 240 Hz
5 (R/W)	PDMINVCLK	Inverted Alternate PDM Clock Enable. When the A2B_PDMCTL2.PDMINVCLK bit is set in a subordinate node, and the A2B_PDMCTL2.PDMALTCLK bit is set, an inverted version of the PDMCLK on the IO7 pin is driven on the BCLK pin. I^2S/TDM is still supported in this mode, but the BCLK frequency is constrained to 64x the SYNC frequency.	
4 (R/W)	PDMALTCLK	PDM Alternate Clock Enable. When the A2B_PDMCTL2.PDMALTCLK bit is set and at least one PDM input pin is enabled, the IO7 pin is used as the PDMCLK clock output pin. For a subordinate node, this allows the BCLK frequency to be set from the I^2S/TDM configuration even when PDM functions are enabled. For a main node, this allows the PDM clock to be a different frequency than the input BCLK. The frequency of the PDM clock on IO7 is 64x the SYNC frequency. If both PDM input pins are disabled (A2B_PDMCTL.PDM0EN = A2B_PDMCTL.PDM1EN = 0), the A2B_PDMCTL2.PDMALTCLK bit is ignored.	

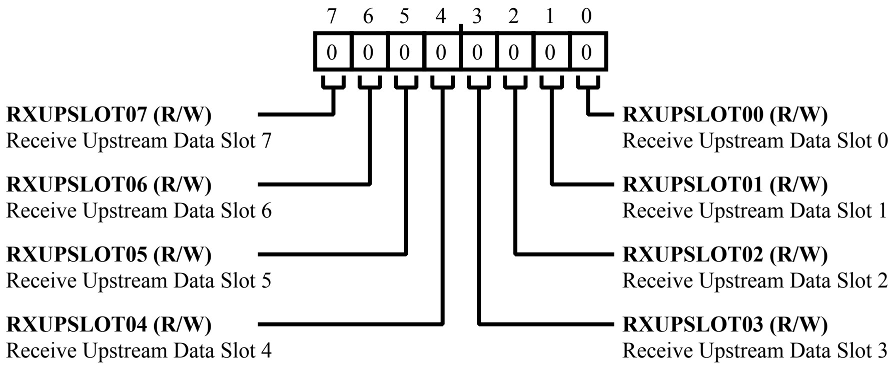
Table 11-66: A2B_PDMCTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	PDM1FFRST	PDM1 Falling Edge First Enable. When the `A2B_PDMCTL2.PDM1FFRST` bit is cleared (default), PDM data on the SIO1 pin is sampled rising edge first. When the `A2B_PDMCTL2.PDM1FFRST` bit is set, the SIO1 pin is sampled falling edge first.	
2 (R/W)	PDM0FFRST	PDM0 Falling Edge First Enable. When the `A2B_PDMCTL2.PDM0FFRST` bit is cleared (default), PDM data on the SIO0 pin is sampled rising edge first. When the `A2B_PDMCTL2.PDM0FFRST` bit is set, the SIO0 pin is sampled falling edge first.	
1:0 (R/W)	PDMDEST	PDM Destination. The `A2B_PDMCTL2.PDMDEST` bit field selects how PDM data is routed. By default, PDM data received by the SIO0 and SIO1 pins goes to the A^{2}B bus after demodulation. The demodulated data can instead or also be routed over the I^{2}S/TDM port to the local node using the SIOn pins.	
		0	(Default) A^{2}B bus only
		1	SIOn pin(s) only
		2	A^{2}B bus and SIOn pin(s)
		3	Reserved

Upstream Data RX Mask 0 Register (Sub Only)

The `A2B_UPMASK0` register identifies which upstream data slots (from 0 to 7) are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM and follow any downstream slots which were received by the subordinate node during the downstream portion of the superframe (defined by the `A2B_LDNSLOTS` register). Changes to this register only take effect after setting the `A2B_CONTROL.NEWSTRCT` bit of the main node.

Address: 0x60



7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXUPSLOT07 (R/W)
Receive Upstream Data Slot 7
RXUPSLOT06 (R/W)
Receive Upstream Data Slot 6
RXUPSLOT05 (R/W)
Receive Upstream Data Slot 5
RXUPSLOT04 (R/W)
Receive Upstream Data Slot 4
RXUPSLOT00 (R/W)
Receive Upstream Data Slot 0
RXUPSLOT01 (R/W)
Receive Upstream Data Slot 1
RXUPSLOT02 (R/W)
Receive Upstream Data Slot 2
RXUPSLOT03 (R/W)
Receive Upstream Data Slot 3

Figure 11-66: A2B_UPMASK0 Register Diagram

Table 11-67: A2B_UPMASK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXUPSLOT07	Receive Upstream Data Slot 7. The `A2B_UPMASK0.RXUPSLOT07` bit defines whether or not upstream data slot 7 is received by the local subordinate node.	
		0	Upstream Data Slot 7 RX Disabled
		1	Upstream Data Slot 7 RX Enabled
6 (R/W)	RXUPSLOT06	Receive Upstream Data Slot 6. The `A2B_UPMASK0.RXUPSLOT06` bit defines whether or not upstream data slot 6 is received by the local subordinate node.	
		0	Upstream Data Slot 6 RX Disabled
		1	Upstream Data Slot 6 RX Enabled
5 (R/W)	RXUPSLOT05	Receive Upstream Data Slot 5. The `A2B_UPMASK0.RXUPSLOT05` bit defines whether or not upstream data slot 5 is received by the local subordinate node.	
		0	Upstream Data Slot 5 RX Disabled
		1	Upstream Data Slot 5 RX Enabled

Table 11-67: A2B_UPMASK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXUPSLOT04	Receive Upstream Data Slot 4. The `A2B_UPMASK0.RXUPSLOT04` bit defines whether or not upstream data slot 4 is received by the local subordinate node.	
		0	Upstream Data Slot 4 RX Disabled
		1	Upstream Data Slot 4 RX Enabled
3 (R/W)	RXUPSLOT03	Receive Upstream Data Slot 3. The `A2B_UPMASK0.RXUPSLOT03` bit defines whether or not upstream data slot 3 is received by the local subordinate node.	
		0	Upstream Data Slot 3 RX Disabled
		1	Upstream Data Slot 3 RX Enabled
2 (R/W)	RXUPSLOT02	Receive Upstream Data Slot 2. The `A2B_UPMASK0.RXUPSLOT02` bit defines whether or not upstream data slot 2 is received by the local subordinate node.	
		0	Upstream Data Slot 2 RX Disabled
		1	Upstream Data Slot 2 RX Enabled
1 (R/W)	RXUPSLOT01	Receive Upstream Data Slot 1. The `A2B_UPMASK0.RXUPSLOT01` bit defines whether or not upstream data slot 1 is received by the local subordinate node.	
		0	Upstream Data Slot 1 RX Disabled
		1	Upstream Data Slot 1 RX Enabled
0 (R/W)	RXUPSLOT00	Receive Upstream Data Slot 0. The `A2B_UPMASK0.RXUPSLOT00` bit defines whether or not upstream data slot 0 is received by the local subordinate node.	
		0	Upstream Data Slot 0 RX Disabled
		1	Upstream Data Slot 0 RX Enabled

Upstream Data RX Mask 1 Register (Sub Only)

The A2B_UPMASK1 register identifies which upstream data slots (from 8 to 15) are received from the A^2B bus. These data slots may be transmitted via I^2S/TDM and follow any downstream slots which were received by the subordinate node during the downstream portion of the superframe (defined by the A2B_LDNSLOTS register). Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x61

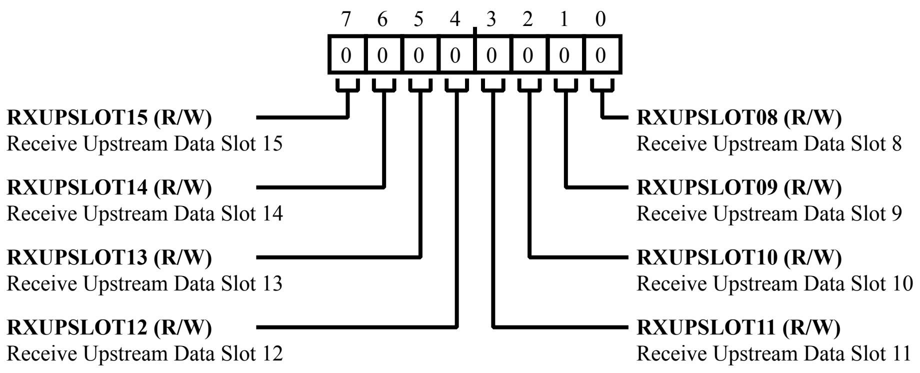

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXUPSLOT15 (R/W)
Receive Upstream Data Slot 15
RXUPSLOT14 (R/W)
Receive Upstream Data Slot 14
RXUPSLOT13 (R/W)
Receive Upstream Data Slot 13
RXUPSLOT12 (R/W)
Receive Upstream Data Slot 12
RXUPSLOT08 (R/W)
Receive Upstream Data Slot 8
RXUPSLOT09 (R/W)
Receive Upstream Data Slot 9
RXUPSLOT10 (R/W)
Receive Upstream Data Slot 10
RXUPSLOT11 (R/W)
Receive Upstream Data Slot 11

Figure 11-67: A2B_UPMASK1 Register Diagram

Table 11-68: A2B_UPMASK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXUPSLOT15	Receive Upstream Data Slot 15. The A2B_UPMASK1.RXUPSLOT15 bit defines whether or not upstream data slot 15 is received by the local subordinate node.	
		0	Upstream Data Slot 15 RX Disabled
		1	Upstream Data Slot 15 RX Enabled
6 (R/W)	RXUPSLOT14	Receive Upstream Data Slot 14. The A2B_UPMASK1.RXUPSLOT14 bit defines whether or not upstream data slot 14 is received by the local subordinate node.	
		0	Upstream Data Slot 14 RX Disabled
		1	Upstream Data Slot 14 RX Enabled
5 (R/W)	RXUPSLOT13	Receive Upstream Data Slot 13. The A2B_UPMASK1.RXUPSLOT13 bit defines whether or not upstream data slot 13 is received by the local subordinate node.	
		0	Upstream Data Slot 13 RX Disabled
		1	Upstream Data Slot 13 RX Enabled

Table 11-68: A2B_UPMASK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXUPSLOT12	Receive Upstream Data Slot 12. The `A2B_UPMASK1.RXUPSLOT12` bit defines whether or not upstream data slot 12 is received by the local subordinate node.	
		0	Upstream Data Slot 12 RX Disabled
		1	Upstream Data Slot 12 RX Enabled
3 (R/W)	RXUPSLOT11	Receive Upstream Data Slot 11. The `A2B_UPMASK1.RXUPSLOT11` bit defines whether or not upstream data slot 11 is received by the local subordinate node.	
		0	Upstream Data Slot 11 RX Disabled
		1	Upstream Data Slot 11 RX Enabled
2 (R/W)	RXUPSLOT10	Receive Upstream Data Slot 10. The `A2B_UPMASK1.RXUPSLOT10` bit defines whether or not upstream data slot 10 is received by the local subordinate node.	
		0	Upstream Data Slot 10 RX Disabled
		1	Upstream Data Slot 10 RX Enabled
1 (R/W)	RXUPSLOT09	Receive Upstream Data Slot 9. The `A2B_UPMASK1.RXUPSLOT09` bit defines whether or not upstream data slot 9 is received by the local subordinate node.	
		0	Upstream Data Slot 9 RX Disabled
		1	Upstream Data Slot 9 RX Enabled
0 (R/W)	RXUPSLOT08	Receive Upstream Data Slot 8. The `A2B_UPMASK1.RXUPSLOT08` bit defines whether or not upstream data slot 8 is received by the local subordinate node.	
		0	Upstream Data Slot 8 RX Disabled
		1	Upstream Data Slot 8 RX Enabled

Upstream Data RX Mask 2 Register (Sub Only)

The `A2B_UPMASK2` register identifies which upstream data slots (from 16 to 23) are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM and follow any downstream slots which were received by the subordinate node during the downstream portion of the superframe (defined by the `A2B_LDNSLOTS` register). Changes to this register only take effect after setting the `A2B_CONTROL.NEWSTRCT` bit of the main node.

Address: 0x62

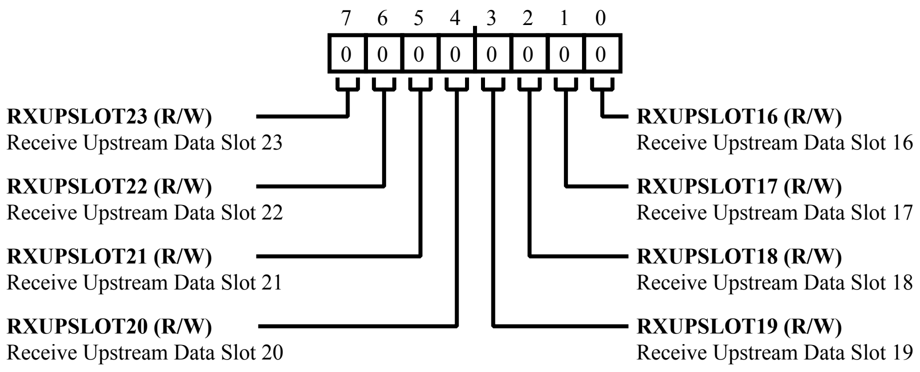

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXUPSLOT23 (R/W)
Receive Upstream Data Slot 23
RXUPSLOT22 (R/W)
Receive Upstream Data Slot 22
RXUPSLOT21 (R/W)
Receive Upstream Data Slot 21
RXUPSLOT20 (R/W)
Receive Upstream Data Slot 20
RXUPSLOT16 (R/W)
Receive Upstream Data Slot 16
RXUPSLOT17 (R/W)
Receive Upstream Data Slot 17
RXUPSLOT18 (R/W)
Receive Upstream Data Slot 18
RXUPSLOT19 (R/W)
Receive Upstream Data Slot 19

Figure 11-68: A2B_UPMASK2 Register Diagram

Table 11-69: A2B_UPMASK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXUPSLOT23	Receive Upstream Data Slot 23. The `A2B_UPMASK2.RXUPSLOT23` bit defines whether or not upstream data slot 23 is received by the local subordinate node.	
		0	Upstream Data Slot 23 RX Disabled
		1	Upstream Data Slot 23 RX Enabled
6 (R/W)	RXUPSLOT22	Receive Upstream Data Slot 22. The `A2B_UPMASK2.RXUPSLOT22` bit defines whether or not upstream data slot 22 is received by the local subordinate node.	
		0	Upstream Data Slot 22 RX Disabled
		1	Upstream Data Slot 22 RX Enabled
5 (R/W)	RXUPSLOT21	Receive Upstream Data Slot 21. The `A2B_UPMASK2.RXUPSLOT21` bit defines whether or not upstream data slot 21 is received by the local subordinate node.	
		0	Upstream Data Slot 21 RX Disabled
		1	Upstream Data Slot 21 RX Enabled

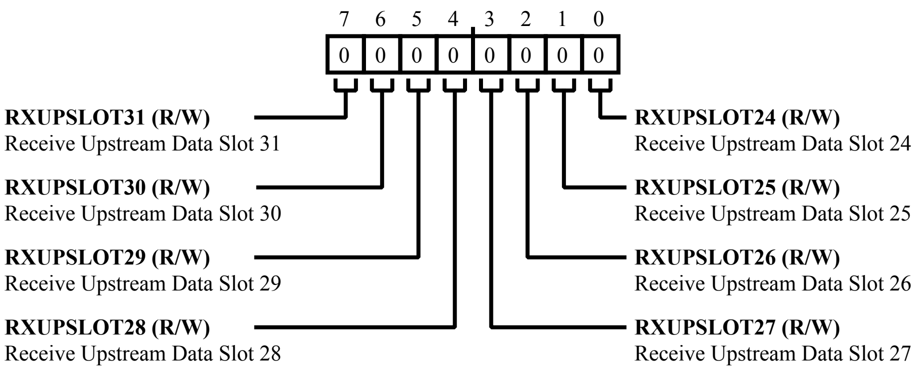
Table 11-69: A2B_UPMASK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXUPSLOT20	Receive Upstream Data Slot 20. The `A2B_UPMASK2.RXUPSLOT20` bit defines whether or not upstream data slot 20 is received by the local subordinate node.	
		0	Upstream Data Slot 20 RX Disabled
		1	Upstream Data Slot 20 RX Enabled
3 (R/W)	RXUPSLOT19	Receive Upstream Data Slot 19. The `A2B_UPMASK2.RXUPSLOT19` bit defines whether or not upstream data slot 19 is received by the local subordinate node.	
		0	Upstream Data Slot 19 RX Disabled
		1	Upstream Data Slot 19 RX Enabled
2 (R/W)	RXUPSLOT18	Receive Upstream Data Slot 18. The `A2B_UPMASK2.RXUPSLOT18` bit defines whether or not upstream data slot 18 is received by the local subordinate node.	
		0	Upstream Data Slot 18 RX Disabled
		1	Upstream Data Slot 18 RX Enabled
1 (R/W)	RXUPSLOT17	Receive Upstream Data Slot 17. The `A2B_UPMASK2.RXUPSLOT17` bit defines whether or not upstream data slot 17 is received by the local subordinate node.	
		0	Upstream Data Slot 17 RX Disabled
		1	Upstream Data Slot 17 RX Enabled
0 (R/W)	RXUPSLOT16	Receive Upstream Data Slot 16. The `A2B_UPMASK2.RXUPSLOT16` bit defines whether or not upstream data slot 16 is received by the local subordinate node.	
		0	Upstream Data Slot 16 RX Disabled
		1	Upstream Data Slot 16 RX Enabled

Upstream Data RX Mask 3 Register (Sub Only)

The `A2B_UPMASK3` register identifies which upstream data slots (from 24 to 31) are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM and follow any downstream slots which were received by the subordinate node during the downstream portion of the superframe (defined by the `A2B_LDNSLOTS` register). Changes to this register only take effect after setting the `A2B_CONTROL.NEWSTRCT` bit of the main node.

Address: 0x63



7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXUPSLOT31 (R/W)
Receive Upstream Data Slot 31
RXUPSLOT30 (R/W)
Receive Upstream Data Slot 30
RXUPSLOT29 (R/W)
Receive Upstream Data Slot 29
RXUPSLOT28 (R/W)
Receive Upstream Data Slot 28
RXUPSLOT24 (R/W)
Receive Upstream Data Slot 24
RXUPSLOT25 (R/W)
Receive Upstream Data Slot 25
RXUPSLOT26 (R/W)
Receive Upstream Data Slot 26
RXUPSLOT27 (R/W)
Receive Upstream Data Slot 27

Figure 11-69: A2B_UPMASK3 Register Diagram

Table 11-70: A2B_UPMASK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXUPSLOT31	Receive Upstream Data Slot 31. The `A2B_UPMASK3.RXUPSLOT31` bit defines whether or not upstream data slot 31 is received by the local subordinate node.	
		0	Upstream Data Slot 31 RX Disabled
		1	Upstream Data Slot 31 RX Enabled
6 (R/W)	RXUPSLOT30	Receive Upstream Data Slot 30. The `A2B_UPMASK3.RXUPSLOT30` bit defines whether or not upstream data slot 30 is received by the local subordinate node.	
		0	Upstream Data Slot 30 RX Disabled
		1	Upstream Data Slot 30 RX Enabled
5 (R/W)	RXUPSLOT29	Receive Upstream Data Slot 29. The `A2B_UPMASK3.RXUPSLOT29` bit defines whether or not upstream data slot 29 is received by the local subordinate node.	
		0	Upstream Data Slot 29 RX Disabled
		1	Upstream Data Slot 29 RX Enabled

Table 11-70: A2B_UPMASK3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXUPSLOT28	Receive Upstream Data Slot 28. The `A2B_UPMASK3.RXUPSLOT28` bit defines whether or not upstream data slot 28 is received by the local subordinate node.	
		0	Upstream Data Slot 28 RX Disabled
		1	Upstream Data Slot 28 RX Enabled
3 (R/W)	RXUPSLOT27	Receive Upstream Data Slot 27. The `A2B_UPMASK3.RXUPSLOT27` bit defines whether or not upstream data slot 27 is received by the local subordinate node.	
		0	Upstream Data Slot 27 RX Disabled
		1	Upstream Data Slot 27 RX Enabled
2 (R/W)	RXUPSLOT26	Receive Upstream Data Slot 26. The `A2B_UPMASK3.RXUPSLOT26` bit defines whether or not upstream data slot 26 is received by the local subordinate node.	
		0	Upstream Data Slot 26 RX Disabled
		1	Upstream Data Slot 26 RX Enabled
1 (R/W)	RXUPSLOT25	Receive Upstream Data Slot 25. The `A2B_UPMASK3.RXUPSLOT25` bit defines whether or not upstream data slot 25 is received by the local subordinate node.	
		0	Upstream Data Slot 25 RX Disabled
		1	Upstream Data Slot 25 RX Enabled
0 (R/W)	RXUPSLOT24	Receive Upstream Data Slot 24. The `A2B_UPMASK3.RXUPSLOT24` bit defines whether or not upstream data slot 24 is received by the local subordinate node.	
		0	Upstream Data Slot 24 RX Disabled
		1	Upstream Data Slot 24 RX Enabled

Local Upstream Channel Offset Register (Sub Only)

In a subordinate node, the A2B_UPOFFSET register defines the number of data channels received via I^{2}S/TDM/PDM that are skipped before data slots are transmitted upstream on the A^{2}B bus. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x64

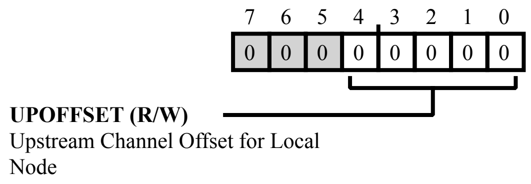


Figure 11-70: A2B_UPOFFSET Register Diagram

Table 11-71: A2B_UPOFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	UPOFFSET	Upstream Channel Offset for Local Node. The A2B_UPOFFSET.UPOFFSET bit field defines the number of data channels received via I^{2}S/TDM/PDM that are skipped before data slots are transmitted upstream on the A^{2}B bus.

Downstream Data RX Mask 0 Register (Sub Only)

The `A2B_DNMASK0` register identifies the downstream data slots (from 0 to 7) that are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM. If none of the bits in this register are set, the `A2B_LDNSLOTS` register defines the number of downstream data slots taken by the local node, as in the AD2410. Changes to this register only take effect after setting the `A2B_CONTROL.NEWSTRCT` bit of the main node.

Address: 0x65

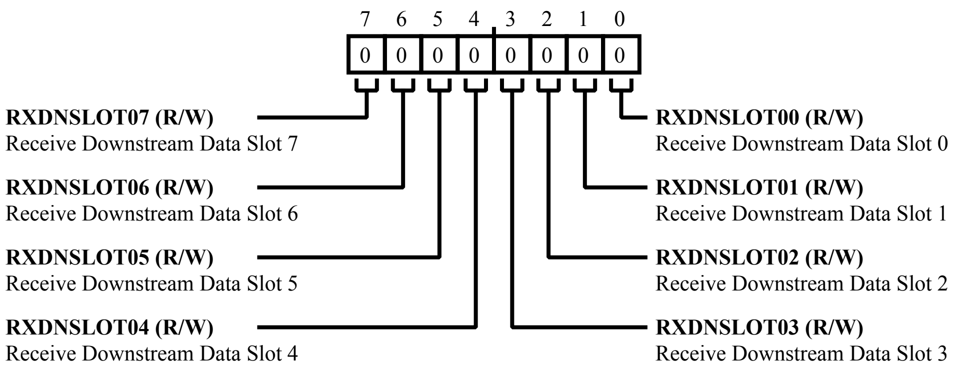

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXDNSLOT07 (R/W)
Receive Downstream Data Slot 7
RXDNSLOT06 (R/W)
Receive Downstream Data Slot 6
RXDNSLOT05 (R/W)
Receive Downstream Data Slot 5
RXDNSLOT04 (R/W)
Receive Downstream Data Slot 4
RXDNSLOT00 (R/W)
Receive Downstream Data Slot 0
RXDNSLOT01 (R/W)
Receive Downstream Data Slot 1
RXDNSLOT02 (R/W)
Receive Downstream Data Slot 2
RXDNSLOT03 (R/W)
Receive Downstream Data Slot 3

Figure 11-71: A2B_DNMASK0 Register Diagram

Table 11-72: A2B_DNMASK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXDNSLOT07	Receive Downstream Data Slot 7. The `A2B_DNMASK0.RXDNSLOT07` bit defines whether or not downstream data slot 7 is received by the local subordinate node.	
		0	Downstream Data Slot 7 RX Disabled
		1	Downstream Data Slot 7 RX Enabled
6 (R/W)	RXDNSLOT06	Receive Downstream Data Slot 6. The `A2B_DNMASK0.RXDNSLOT06` bit defines whether or not downstream data slot 6 is received by the local subordinate node.	
		0	Downstream Data Slot 6 RX Disabled
		1	Downstream Data Slot 6 RX Enabled
5 (R/W)	RXDNSLOT05	Receive Downstream Data Slot 5. The `A2B_DNMASK0.RXDNSLOT05` bit defines whether or not downstream data slot 5 is received by the local subordinate node.	
		0	Downstream Data Slot 5 RX Disabled
		1	Downstream Data Slot 5 RX Enabled

Table 11-72: A2B_DNMASK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXDNSLOT04	Receive Downstream Data Slot 4. The `A2B_DNMASK0.RXDNSLOT04` bit defines whether or not downstream data slot 4 is received by the local subordinate node.	
		0	Downstream Data Slot 4 RX Disabled
		1	Downstream Data Slot 4 RX Enabled
3 (R/W)	RXDNSLOT03	Receive Downstream Data Slot 3. The `A2B_DNMASK0.RXDNSLOT03` bit defines whether or not downstream data slot 3 is received by the local subordinate node.	
		0	Downstream Data Slot 3 RX Disabled
		1	Downstream Data Slot 3 RX Enabled
2 (R/W)	RXDNSLOT02	Receive Downstream Data Slot 2. The `A2B_DNMASK0.RXDNSLOT02` bit defines whether or not downstream data slot 2 is received by the local subordinate node.	
		0	Downstream Data Slot 2 RX Disabled
		1	Downstream Data Slot 2 RX Enabled
1 (R/W)	RXDNSLOT01	Receive Downstream Data Slot 1. The `A2B_DNMASK0.RXDNSLOT01` bit defines whether or not downstream data slot 1 is received by the local subordinate node.	
		0	Downstream Data Slot 1 RX Disabled
		1	Downstream Data Slot 1 RX Enabled
0 (R/W)	RXDNSLOT00	Receive Downstream Data Slot 0. The `A2B_DNMASK0.RXDNSLOT00` bit defines whether or not downstream data slot 0 is received by the local subordinate node.	
		0	Downstream Data Slot 0 RX Disabled
		1	Downstream Data Slot 0 RX Enabled

Downstream Data RX Mask 1 Register (Sub Only)

The A2B_DNMASK1 register identifies the downstream data slots (from 8 to 15) that are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM. If none of the bits in this register are set, the A2B_LDNSLOTS register defines the number of downstream data slots taken by the local node, as in the AD2410. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x66

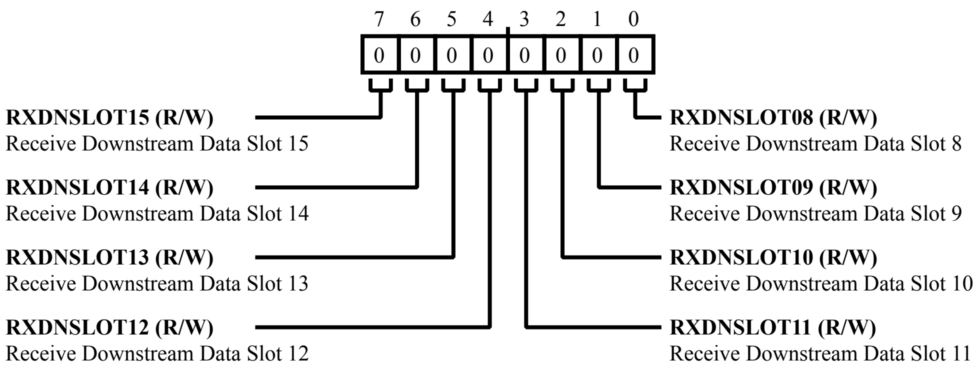

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXDNSLOT15 (R/W)
Receive Downstream Data Slot 15
RXDNSLOT14 (R/W)
Receive Downstream Data Slot 14
RXDNSLOT13 (R/W)
Receive Downstream Data Slot 13
RXDNSLOT12 (R/W)
Receive Downstream Data Slot 12
RXDNSLOT08 (R/W)
Receive Downstream Data Slot 8
RXDNSLOT09 (R/W)
Receive Downstream Data Slot 9
RXDNSLOT10 (R/W)
Receive Downstream Data Slot 10
RXDNSLOT11 (R/W)
Receive Downstream Data Slot 11

Figure 11-72: A2B_DNMASK1 Register Diagram

Table 11-73: A2B_DNMASK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXDNSLOT15	Receive Downstream Data Slot 15. The A2B_DNMASK1.RXDNSLOT15 bit defines whether or not downstream data slot 15 is received by the local subordinate node.	
		0	Downstream Data Slot 15 RX Disabled
		1	Downstream Data Slot 15 RX Enabled
6 (R/W)	RXDNSLOT14	Receive Downstream Data Slot 14. The A2B_DNMASK1.RXDNSLOT14 bit defines whether or not downstream data slot 14 is received by the local subordinate node.	
		0	Downstream Data Slot 14 RX Disabled
		1	Downstream Data Slot 14 RX Enabled
5 (R/W)	RXDNSLOT13	Receive Downstream Data Slot 13. The A2B_DNMASK1.RXDNSLOT13 bit defines whether or not downstream data slot 13 is received by the local subordinate node.	
		0	Downstream Data Slot 13 RX Disabled
		1	Downstream Data Slot 13 RX Enabled

Table 11-73: A2B_DNMASK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXDNSLOT12	Receive Downstream Data Slot 12. The `A2B_DNMASK1.RXDNSLOT12` bit defines whether or not downstream data slot 12 is received by the local subordinate node.	
		0	Downstream Data Slot 12 RX Disabled
		1	Downstream Data Slot 12 RX Enabled
3 (R/W)	RXDNSLOT11	Receive Downstream Data Slot 11. The `A2B_DNMASK1.RXDNSLOT11` bit defines whether or not downstream data slot 11 is received by the local subordinate node.	
		0	Downstream Data Slot 11 RX Disabled
		1	Downstream Data Slot 11 RX Enabled
2 (R/W)	RXDNSLOT10	Receive Downstream Data Slot 10. The `A2B_DNMASK1.RXDNSLOT10` bit defines whether or not downstream data slot 10 is received by the local subordinate node.	
		0	Downstream Data Slot 10 RX Disabled
		1	Downstream Data Slot 10 RX Enabled
1 (R/W)	RXDNSLOT09	Receive Downstream Data Slot 9. The `A2B_DNMASK1.RXDNSLOT09` bit defines whether or not downstream data slot 9 is received by the local subordinate node.	
		0	Downstream Data Slot 9 RX Disabled
		1	Downstream Data Slot 9 RX Enabled
0 (R/W)	RXDNSLOT08	Receive Downstream Data Slot 8. The `A2B_DNMASK1.RXDNSLOT08` bit defines whether or not downstream data slot 8 is received by the local subordinate node.	
		0	Downstream Data Slot 8 RX Disabled
		1	Downstream Data Slot 8 RX Enabled

Downstream Data RX Mask 2 Register (Sub Only)

The A2B_DNMASK2 register identifies the downstream data slots (from 16 to 23) that are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM. If none of the bits in this register are set, the A2B_LDNSLOTS register defines the number of downstream data slots taken by the local node. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x67

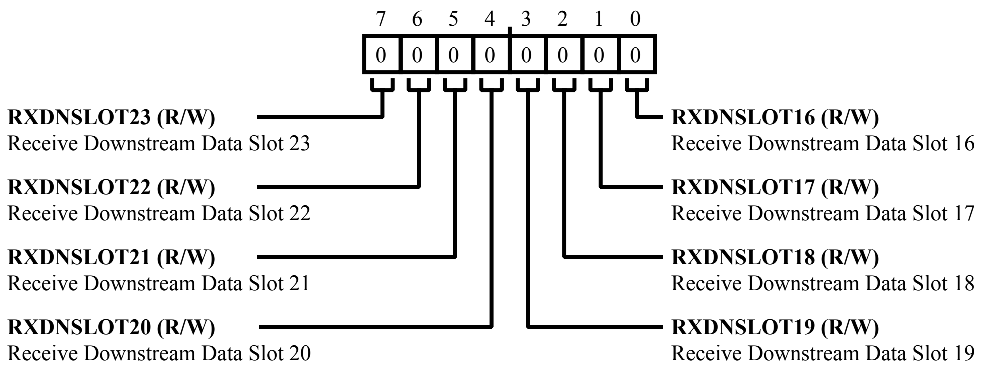

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXDNSLOT23 (R/W)
Receive Downstream Data Slot 23
RXDNSLOT22 (R/W)
Receive Downstream Data Slot 22
RXDNSLOT21 (R/W)
Receive Downstream Data Slot 21
RXDNSLOT20 (R/W)
Receive Downstream Data Slot 20
RXDNSLOT16 (R/W)
Receive Downstream Data Slot 16
RXDNSLOT17 (R/W)
Receive Downstream Data Slot 17
RXDNSLOT18 (R/W)
Receive Downstream Data Slot 18
RXDNSLOT19 (R/W)
Receive Downstream Data Slot 19

Figure 11-73: A2B_DNMASK2 Register Diagram

Table 11-74: A2B_DNMASK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXDNSLOT23	Receive Downstream Data Slot 23. The A2B_DNMASK2.RXDNSLOT23 bit defines whether or not downstream data slot 23 is received by the local subordinate node.	
		0	Downstream Data Slot 23 RX Disabled
		1	Downstream Data Slot 23 RX Enabled
6 (R/W)	RXDNSLOT22	Receive Downstream Data Slot 22. The A2B_DNMASK2.RXDNSLOT22 bit defines whether or not downstream data slot 22 is received by the local subordinate node.	
		0	Downstream Data Slot 22 RX Disabled
		1	Downstream Data Slot 22 RX Enabled
5 (R/W)	RXDNSLOT21	Receive Downstream Data Slot 21. The A2B_DNMASK2.RXDNSLOT21 bit defines whether or not downstream data slot 21 is received by the local subordinate node.	
		0	Downstream Data Slot 21 RX Disabled
		1	Downstream Data Slot 21 RX Enabled

Table 11-74: A2B_DNMASK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXDNSLOT20	Receive Downstream Data Slot 20. The `A2B_DNMASK2.RXDNSLOT20` bit defines whether or not downstream data slot 20 is received by the local subordinate node.	
		0	Downstream Data Slot 20 RX Disabled
		1	Downstream Data Slot 20 RX Enabled
3 (R/W)	RXDNSLOT19	Receive Downstream Data Slot 19. The `A2B_DNMASK2.RXDNSLOT19` bit defines whether or not downstream data slot 19 is received by the local subordinate node.	
		0	Downstream Data Slot 19 RX Disabled
		1	Downstream Data Slot 19 RX Enabled
2 (R/W)	RXDNSLOT18	Receive Downstream Data Slot 18. The `A2B_DNMASK2.RXDNSLOT18` bit defines whether or not downstream data slot 18 is received by the local subordinate node.	
		0	Downstream Data Slot 18 RX Disabled
		1	Downstream Data Slot 18 RX Enabled
1 (R/W)	RXDNSLOT17	Receive Downstream Data Slot 17. The `A2B_DNMASK2.RXDNSLOT17` bit defines whether or not downstream data slot 17 is received by the local subordinate node.	
		0	Downstream Data Slot 17 RX Disabled
		1	Downstream Data Slot 17 RX Enabled
0 (R/W)	RXDNSLOT16	Receive Downstream Data Slot 16. The `A2B_DNMASK2.RXDNSLOT16` bit defines whether or not downstream data slot 16 is received by the local subordinate node.	
		0	Downstream Data Slot 16 RX Disabled
		1	Downstream Data Slot 16 RX Enabled

Downstream Data RX Mask 3 Register (Sub Only)

The A2B_DNMASK3 register identifies the downstream data slots (from 24 to 31) that are received from the A^{2}B bus. These data slots may be transmitted via I^{2}S/TDM. If none of the bits in this register are set, the A2B_LDNSLOTS register defines the number of downstream data slots taken by the local node, as in the AD2410. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x68

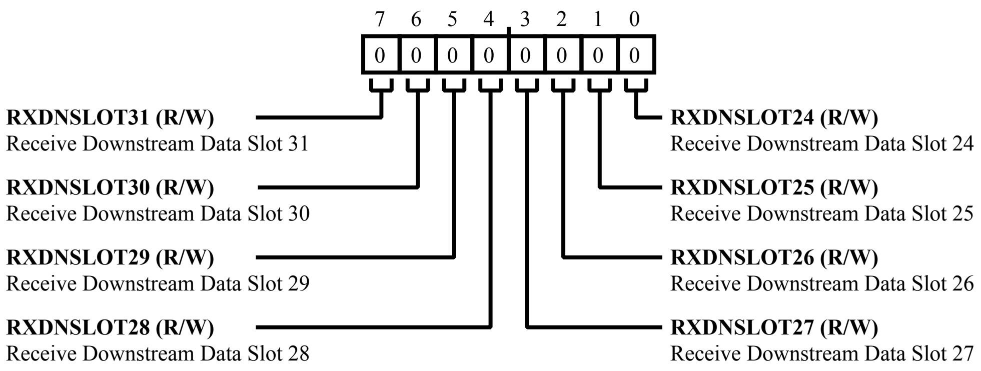

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
RXDNSLOT31 (R/W)
Receive Downstream Data Slot 31
RXDNSLOT30 (R/W)
Receive Downstream Data Slot 30
RXDNSLOT29 (R/W)
Receive Downstream Data Slot 29
RXDNSLOT28 (R/W)
Receive Downstream Data Slot 28
RXDNSLOT24 (R/W)
Receive Downstream Data Slot 24
RXDNSLOT25 (R/W)
Receive Downstream Data Slot 25
RXDNSLOT26 (R/W)
Receive Downstream Data Slot 26
RXDNSLOT27 (R/W)
Receive Downstream Data Slot 27

Figure 11-74: A2B_DNMASK3 Register Diagram

Table 11-75: A2B_DNMASK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXDNSLOT31	Receive Downstream Data Slot 31. The A2B_DNMASK3.RXDNSLOT31 bit defines whether or not downstream data slot 31 is received by the local subordinate node.	
		0	Downstream Data Slot 31 RX Disabled
		1	Downstream Data Slot 31 RX Enabled
6 (R/W)	RXDNSLOT30	Receive Downstream Data Slot 30. The A2B_DNMASK3.RXDNSLOT30 bit defines whether or not downstream data slot 30 is received by the local subordinate node.	
		0	Downstream Data Slot 30 RX Disabled
		1	Downstream Data Slot 30 RX Enabled
5 (R/W)	RXDNSLOT29	Receive Downstream Data Slot 29. The A2B_DNMASK3.RXDNSLOT29 bit defines whether or not downstream data slot 29 is received by the local subordinate node.	
		0	Downstream Data Slot 29 RX Disabled
		1	Downstream Data Slot 29 RX Enabled

Table 11-75: A2B_DNMASK3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	RXDNSLOT28	Receive Downstream Data Slot 28. The `A2B_DNMASK3.RXDNSLOT28` bit defines whether or not downstream data slot 28 is received by the local subordinate node.	
		0	Downstream Data Slot 28 RX Disabled
		1	Downstream Data Slot 28 RX Enabled
3 (R/W)	RXDNSLOT27	Receive Downstream Data Slot 27. The `A2B_DNMASK3.RXDNSLOT27` bit defines whether or not downstream data slot 27 is received by the local subordinate node.	
		0	Downstream Data Slot 27 RX Disabled
		1	Downstream Data Slot 27 RX Enabled
2 (R/W)	RXDNSLOT26	Receive Downstream Data Slot 26. The `A2B_DNMASK3.RXDNSLOT26` bit defines whether or not downstream data slot 26 is received by the local subordinate node.	
		0	Downstream Data Slot 26 RX Disabled
		1	Downstream Data Slot 26 RX Enabled
1 (R/W)	RXDNSLOT25	Receive Downstream Data Slot 25. The `A2B_DNMASK3.RXDNSLOT25` bit defines whether or not downstream data slot 25 is received by the local subordinate node.	
		0	Downstream Data Slot 25 RX Disabled
		1	Downstream Data Slot 25 RX Enabled
0 (R/W)	RXDNSLOT24	Receive Downstream Data Slot 24. The `A2B_DNMASK3.RXDNSLOT24` bit defines whether or not downstream data slot 24 is received by the local subordinate node.	
		0	Downstream Data Slot 24 RX Disabled
		1	Downstream Data Slot 24 RX Enabled

Local Downstream Channel Offset Register (Sub Only)

In a subordinate node, the A2B_DNOFFSET register defines the number of data channels received via I^2S/TDM/PDM that are skipped before data slots are transmitted downstream on the A^2B bus. The value in the A2B_DNOFFSET register is used only if any of the bits in the A2B_DNMASK0 through A2B_DNMASK3 registers are set and the A2B_LDNSLOTS register is non-zero. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x69

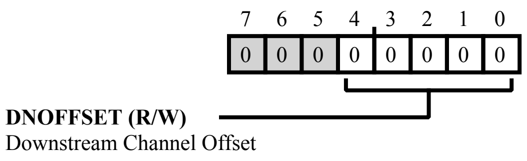


Figure 11-75: A2B_DNOFFSET Register Diagram

Table 11-76: A2B_DNOFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	DNOFFSET	Downstream Channel Offset. The A2B_DNOFFSET.DNOFFSET bit field defines the number of data channels received via I^2S/TDM/PDM that are skipped before data slots are transmitted downstream on the A^2B bus.

Chip ID Register 0

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6A

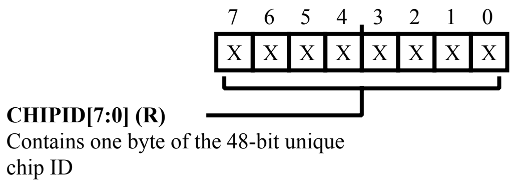


Figure 11-76: A2B_CHIPID0 Register Diagram

Table 11-77: A2B_CHIPID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Chip ID Register 1

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6B

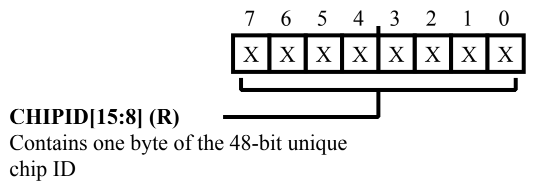


Figure 11-77: A2B_CHIPID1 Register Diagram

Table 11-78: A2B_CHIPID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Chip ID Register 2

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6C

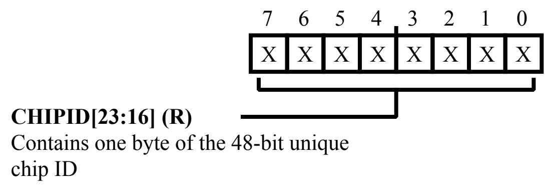


Figure 11-78: A2B_CHIPID2 Register Diagram

Table 11-79: A2B_CHIPID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Chip ID Register 3

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6D

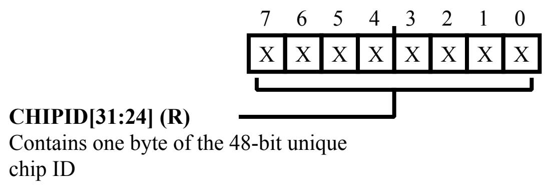


Figure 11-79: A2B_CHIPID3 Register Diagram

Table 11-80: A2B_CHIPID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Chip ID Register 4

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6E

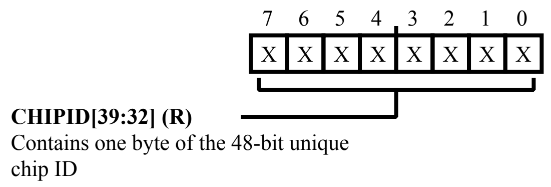


Figure 11-80: A2B_CHIPID4 Register Diagram

Table 11-81: A2B_CHIPID4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Chip ID Register 5

The A2B_CHIPID0 through A2B_CHIPID5 registers concatenate to form a unique 48-bit ID for the transceiver, where A2B_CHIPID0 contains the LSB (bits 7:0) and A2B_CHIPID5 contains the MSB (bits 47:40).

Address: 0x6F

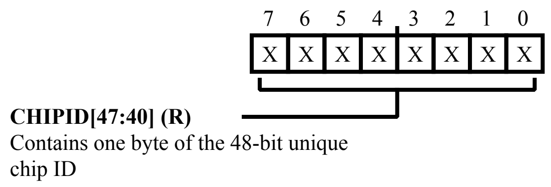


Figure 11-81: A2B_CHIPID5 Register Diagram

Table 11-82: A2B_CHIPID5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CHIPID	Contains one byte of the 48-bit unique chip ID.

Data Tunnel Configuration Register

The A2B_DTCFG register is used to configure a node to participate in a data tunnel. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x7C

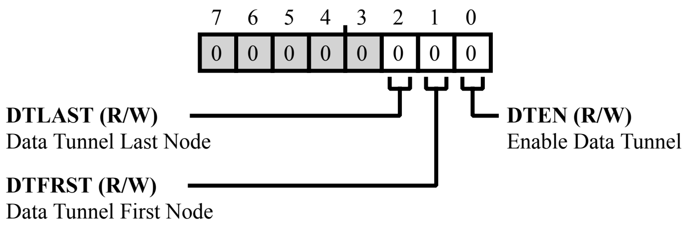


Figure 11-82: A2B_DTCFG Register Diagram

Table 11-83: A2B_DTCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	DTLAST	Data Tunnel Last Node. The A2B_DTCFG.DTLAST bit should be set in the last (most downstream) node in a data tunnel.
1 (R/W)	DTFRST	Data Tunnel First Node. The A2B_DTCFG.DTFRST bit should be set in the first (most upstream) node in a data tunnel.
0 (R/W)	DTEN	Enable Data Tunnel. The A2B_DTCFG.DTEN bit enables a data tunnel in the current node assuming DNS and UPS are both set. The width of the data tunnel is set by the DTSLOTS register.

Data Tunnel Slots Register

The A2B_DTSLOTS register is used to configure the number of upstream and downstream slots used in a data tunnel. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x7D

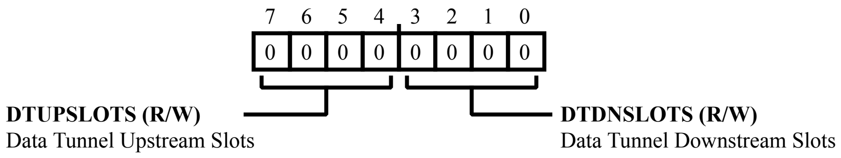


Figure 11-83: A2B_DTSLOTS Register Diagram

Table 11-84: A2B_DTSLOTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/W)	DTUPSLOTS	Data Tunnel Upstream Slots.	
		0	0 Upstream Data Slots
		1	1 Upstream Data Slot
		2	2 Upstream Data Slots
		3	3 Upstream Data Slots
		4	4 Upstream Data Slots
		5	5 Upstream Data Slots
		6	6 Upstream Data Slots
		7	7 Upstream Data Slots
		8	8 Upstream Data Slots
		9	9 Upstream Data Slots
		10	10 Upstream Data Slots
		11	11 Upstream Data Slots
		12	12 Upstream Data Slots
3:0 (R/W)	DTDNSLOTS	Data Tunnel Downstream Slots.	
		0	0 Downstream Data Slots
		1	1 Downstream Data Slot
		2	2 Downstream Data Slots
		3	3 Downstream Data Slots
		4	4 Downstream Data Slots

Table 11-84: A2B_DTSLOTS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	5 Downstream Data Slots
		6	6 Downstream Data Slots
		7	7 Downstream Data Slots
		8	8 Downstream Data Slots
		9	9 Downstream Data Slots
		10	10 Downstream Data Slots
		11	11 Downstream Data Slots
		12	12 Downstream Data Slots

Data Tunnel Downstream Offset Register

The A2B_DTDNOFFS register is used to configure the downstream offset of the data tunnel. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x7E

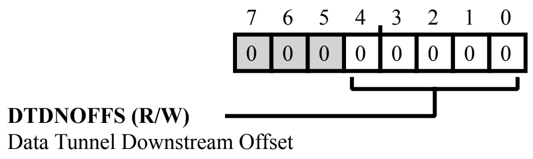


Figure 11-84: A2B_DTDNOFFS Register Diagram

Table 11-85: A2B_DTDNOFFS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	DTDNOFFS	Data Tunnel Downstream Offset.

Data Tunnel Upstream Offset Register

The A2B_DTUPOFFS register is used to configure the upstream offset of the data tunnel. Changes to this register only take effect after setting the A2B_CONTROL.NEWSTRCT bit of the main node.

Address: 0x7F

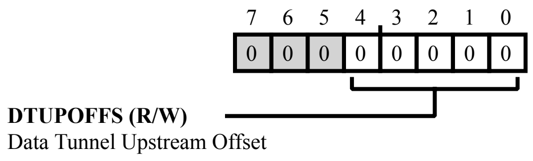


Figure 11-85: A2B_DTUPOFFS Register Diagram

Table 11-86: A2B_DTUPOFFS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	DTUPOFFS	Data Tunnel Upstream Offset.

GPIO Over Distance Enable Register

The A2B_GPIODEN register controls the general-purpose I/O pins for use in GPIO Over Distance.

Address: 0x80

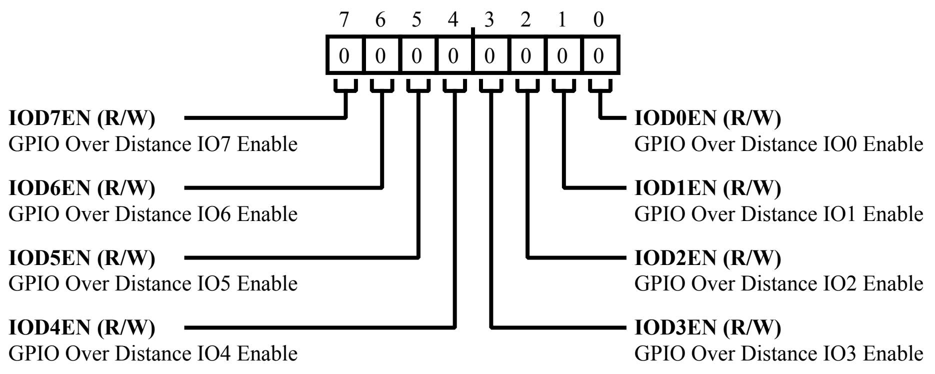

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
IOD7EN (R/W)
GPIO Over Distance IO7 Enable
IOD6EN (R/W)
GPIO Over Distance IO6 Enable
IOD5EN (R/W)
GPIO Over Distance IO5 Enable
IOD4EN (R/W)
GPIO Over Distance IO4 Enable
IOD0EN (R/W)
GPIO Over Distance IO0 Enable
IOD1EN (R/W)
GPIO Over Distance IO1 Enable
IOD2EN (R/W)
GPIO Over Distance IO2 Enable
IOD3EN (R/W)
GPIO Over Distance IO3 Enable

Figure 11-86: A2B_GPIODEN Register Diagram

Table 11-87: A2B_GPIODEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	IOD7EN	GPIO Over Distance IO7 Enable. The A2B_GPIODEN.IOD7EN bit enables GPIO Over Distance for IO7.	
		0	GPIO Over Distance for IO7 Disabled
		1	GPIO Over Distance for IO7 Enabled
6 (R/W)	IOD6EN	GPIO Over Distance IO6 Enable. The A2B_GPIODEN.IOD6EN bit enables GPIO Over Distance for IO6.	
		0	GPIO Over Distance for IO6 Disabled
		1	GPIO Over Distance for IO6 Enabled
5 (R/W)	IOD5EN	GPIO Over Distance IO5 Enable. The A2B_GPIODEN.IOD5EN bit enables GPIO Over Distance for IO5.	
		0	GPIO Over Distance for IO5 Disabled
		1	GPIO Over Distance for IO5 Enabled
4 (R/W)	IOD4EN	GPIO Over Distance IO4 Enable. The A2B_GPIODEN.IOD4EN bit enables GPIO Over Distance for IO4.	
		0	GPIO Over Distance for IO4 Disabled
		1	GPIO Over Distance for IO4 Enabled

Table 11-87: A2B_GPIODEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	IOD3EN	GPIO Over Distance IO3 Enable. The `A2B_GPIODEN.IOD3EN` bit enables GPIO Over Distance for IO3.	
		0	GPIO Over Distance for IO3 Disabled
		1	GPIO Over Distance for IO3 Enabled
2 (R/W)	IOD2EN	GPIO Over Distance IO2 Enable. The `A2B_GPIODEN.IOD2EN` bit enables GPIO Over Distance for IO2.	
		0	GPIO Over Distance for IO2 Disabled
		1	GPIO Over Distance for IO2 Enabled
1 (R/W)	IOD1EN	GPIO Over Distance IO1 Enable. The `A2B_GPIODEN.IOD1EN` bit enables GPIO Over Distance for IO1.	
		0	GPIO Over Distance for IO1 Disabled
		1	GPIO Over Distance for IO1 Enabled
0 (R/W)	IOD0EN	GPIO Over Distance IO0 Enable. The `A2B_GPIODEN.IOD0EN` bit enables GPIO Over Distance for IO0.	
		0	GPIO Over Distance for IO0 Disabled
		1	GPIO Over Distance for IO0 Enabled

GPIO Over Distance Mask 0 Register

The A2B_GPIOD0MSK register controls the mapping between the GPIO0 pin and the Bus GPIO Ports.

Address: 0x81

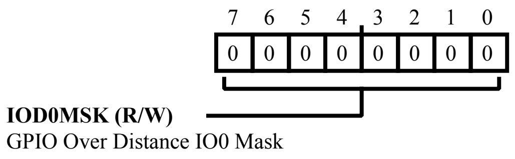


Figure 11-87: A2B_GPIOD0MSK Register Diagram

Table 11-88: A2B_GPIOD0MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD0MSK	GPIO Over Distance IO0 Mask.

GPIO Over Distance Mask 1 Register

The A2B_GPIOD1MSK register controls the mapping between the GPIO1 pin and the Bus GPIO Ports.

Address: 0x82

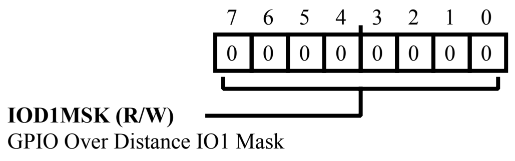


Figure 11-88: A2B_GPIOD1MSK Register Diagram

Table 11-89: A2B_GPIOD1MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD1MSK	GPIO Over Distance IO1 Mask.

GPIO Over Distance Mask 2 Register

The `A2B_GPIOD2MSK` register controls the mapping between the GPIO2 pin and the Bus GPIO Ports.

Address: 0x83

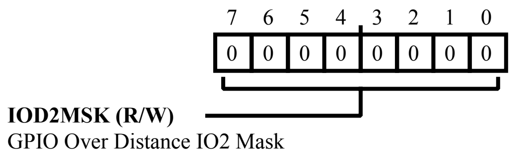


Figure 11-89: A2B_GPIOD2MSK Register Diagram

Table 11-90: A2B_GPIOD2MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD2MSK	GPIO Over Distance IO2 Mask.

GPIO Over Distance Mask 3 Register

The A2B_GPIOD3MSK register controls the mapping between the GPIO3 pin and the Bus GPIO Ports.

Address: 0x84

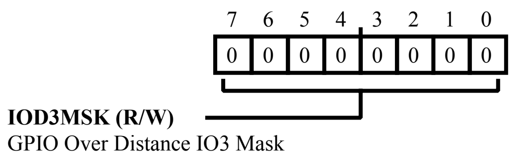


Figure 11-90: A2B_GPIOD3MSK Register Diagram

Table 11-91: A2B_GPIOD3MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD3MSK	GPIO Over Distance IO3 Mask.

GPIO Over Distance Mask 4 Register

The A2B_GPIOD4MSK register controls the mapping between the GPIO4 pin and the Bus GPIO Ports.

Address: 0x85

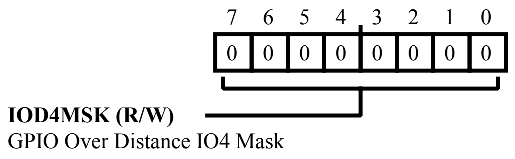


Figure 11-91: A2B_GPIOD4MSK Register Diagram

Table 11-92: A2B_GPIOD4MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD4MSK	GPIO Over Distance IO4 Mask.

GPIO Over Distance Mask 5 Register

The A2B_GPIOD5MSK register controls the mapping between the GPIO5 pin and the Bus GPIO Ports.

Address: 0x86

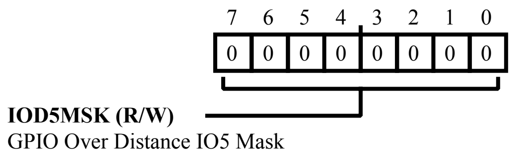


Figure 11-92: A2B_GPIOD5MSK Register Diagram

Table 11-93: A2B_GPIOD5MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD5MSK	GPIO Over Distance IO5 Mask.

GPIO Over Distance Mask 6 Register

The A2B_GPIOD6MSK register controls the mapping between the GPIO6 pin and the Bus GPIO Ports.

Address: 0x87

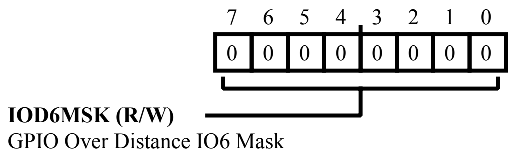


Figure 11-93: A2B_GPIOD6MSK Register Diagram

Table 11-94: A2B_GPIOD6MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD6MSK	GPIO Over Distance IO6 Mask.

GPIO Over Distance Mask 7 Register

The A2B_GPIOD7MSK register controls the mapping between the GPIO7 pin and the Bus GPIO Ports.

Address: 0x88

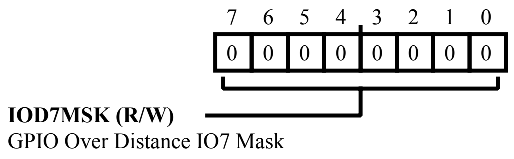


Figure 11-94: A2B_GPIOD7MSK Register Diagram

Table 11-95: A2B_GPIOD7MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	IOD7MSK	GPIO Over Distance IO7 Mask.

GPIO Over Distance Data Register

Address: 0x89

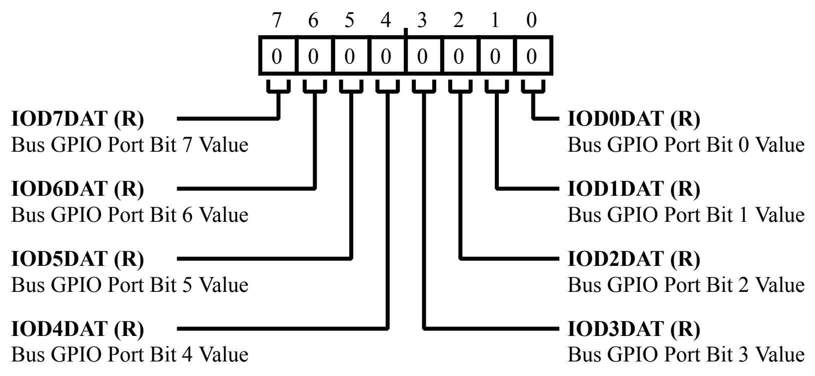

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
IOD7DAT (R)
Bus GPIO Port Bit 7 Value
IOD6DAT (R)
Bus GPIO Port Bit 6 Value
IOD5DAT (R)
Bus GPIO Port Bit 5 Value
IOD4DAT (R)
Bus GPIO Port Bit 4 Value
IOD0DAT (R)
Bus GPIO Port Bit 0 Value
IOD1DAT (R)
Bus GPIO Port Bit 1 Value
IOD2DAT (R)
Bus GPIO Port Bit 2 Value
IOD3DAT (R)
Bus GPIO Port Bit 3 Value

Figure 11-95: A2B_GPIODDAT Register Diagram

Table 11-96: A2B_GPIODDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	IOD7DAT	Bus GPIO Port Bit 7 Value.
6 (R/NW)	IOD6DAT	Bus GPIO Port Bit 6 Value.
5 (R/NW)	IOD5DAT	Bus GPIO Port Bit 5 Value.
4 (R/NW)	IOD4DAT	Bus GPIO Port Bit 4 Value.
3 (R/NW)	IOD3DAT	Bus GPIO Port Bit 3 Value.
2 (R/NW)	IOD2DAT	Bus GPIO Port Bit 2 Value.
1 (R/NW)	IOD1DAT	Bus GPIO Port Bit 1 Value.
0 (R/NW)	IOD0DAT	Bus GPIO Port Bit 0 Value.

GPIO Over Distance Invert Register

Address: 0x8A

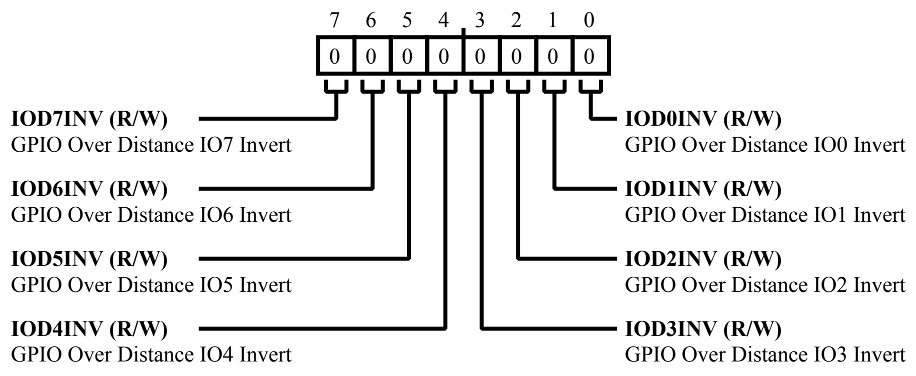

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
IOD7INV (R/W)
GPIO Over Distance IO7 Invert
IOD6INV (R/W)
GPIO Over Distance IO6 Invert
IOD5INV (R/W)
GPIO Over Distance IO5 Invert
IOD4INV (R/W)
GPIO Over Distance IO4 Invert
IOD0INV (R/W)
GPIO Over Distance IO0 Invert
IOD1INV (R/W)
GPIO Over Distance IO1 Invert
IOD2INV (R/W)
GPIO Over Distance IO2 Invert
IOD3INV (R/W)
GPIO Over Distance IO3 Invert

Figure 11-96: A2B_GPIODINV Register Diagram

Table 11-97: A2B_GPIODINV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	IOD7INV	GPIO Over Distance IO7 Invert.
6 (R/W)	IOD6INV	GPIO Over Distance IO6 Invert.
5 (R/W)	IOD5INV	GPIO Over Distance IO5 Invert.
4 (R/W)	IOD4INV	GPIO Over Distance IO4 Invert.
3 (R/W)	IOD3INV	GPIO Over Distance IO3 Invert.
2 (R/W)	IOD2INV	GPIO Over Distance IO2 Invert.
1 (R/W)	IOD1INV	GPIO Over Distance IO1 Invert.
0 (R/W)	IOD0INV	GPIO Over Distance IO0 Invert.

Mailbox 0 Control Register (Sub Only)

The A2B_MBOX0CTL register contains bits that control direction, message length and interrupts.

Address: 0x90

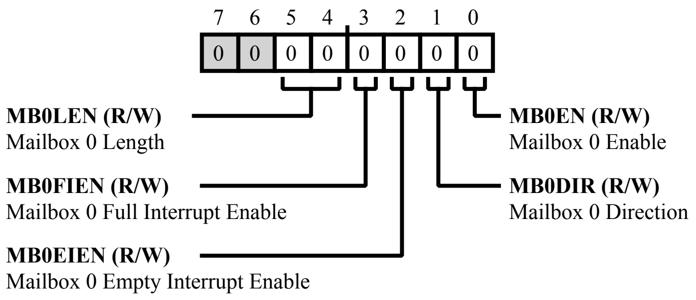


Figure 11-97: A2B_MBOX0CTL Register Diagram

Table 11-98: A2B_MBOX0CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/W)	MB0LEN	Mailbox 0 Length. The A2B_MBOX0CTL.MB0LEN bit field controls the length of Mailbox 0.	
		0	1 Byte
		1	2 Bytes
		2	3 Bytes
		3	4 Bytes
3 (R/W)	MB0FIEN	Mailbox 0 Full Interrupt Enable. The A2B_MBOX0CTL.MB0FIEN bit enables an interrupt which is generated when Mailbox 0 becomes full.	
		0	Mailbox 0 Interrupt on Full Disabled
		1	Mailbox 0 Interrupt on Full Enabled
2 (R/W)	MB0EIEN	Mailbox 0 Empty Interrupt Enable. The A2B_MBOX0CTL.MB0EIEN bit enables an interrupt which is generated when Mailbox 0 becomes empty.	
		0	Mailbox 0 Interrupt on Empty Disabled
		1	Mailbox 0 Interrupt on Empty Enabled
1 (R/W)	MB0DIR	Mailbox 0 Direction. The A2B_MBOX0CTL.MB0DIR bit controls the direction of Mailbox 0.	
		0	Mailbox 0 is Receive Mailbox
		1	Mailbox 0 is Transmit Mailbox

Table 11-98: A2B_MBOX0CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	MB0EN	Mailbox 0 Enable. Setting the `A2B_MBOX0CTL.MB0EN` bit enables Mailbox 0.	
		0	Mailbox 0 Disabled
		1	Mailbox 0 Enabled

Mailbox 0 Status Register (Sub Only)

The A2B_MBOX0STAT register reports the status of the configured mailbox interrupts.

Address: 0x91

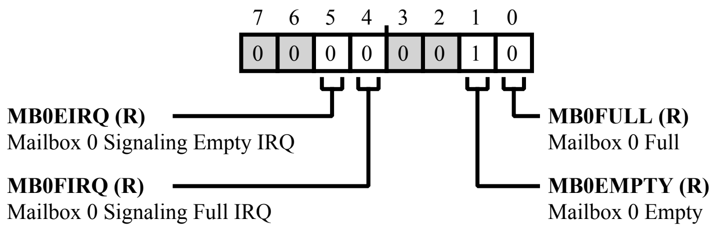


Figure 11-98: A2B_MBOX0STAT Register Diagram

Table 11-99: A2B_MBOX0STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	MB0EIRQ	Mailbox 0 Signaling Empty IRQ. The A2B_MBOX0STAT.MB0EIRQ bit indicates whether or not the Mailbox 0 empty interrupt is active.	
		0	Mailbox 0 Empty Interrupt Inactive
		1	Mailbox 0 Empty Interrupt Active
4 (R/NW)	MB0FIRQ	Mailbox 0 Signaling Full IRQ. The A2B_MBOX0STAT.MB0FIRQ bit indicates whether or not the Mailbox 0 full interrupt is active.	
		0	Mailbox 0 Full Interrupt Inactive
		1	Mailbox 0 Full Interrupt Active
1 (R/NW)	MB0EMPTY	Mailbox 0 Empty. The A2B_MBOX0STAT.MB0EMPTY bit indicates whether or not Mailbox 0 is empty.	
		0	Mailbox 0 Currently Not Empty
		1	Mailbox 0 Currently Empty
0 (R/NW)	MB0FULL	Mailbox 0 Full. The A2B_MBOX0STAT.MB0FULL bit indicates whether or not Mailbox 0 is full.	
		0	Mailbox 0 Currently Not Full
		1	Mailbox 0 Currently Full

Mailbox 0 Byte 0 Register (Sub Only)

The A2B_MBOX0B0 register represents byte 0 of Mailbox 0.

Address: 0x92

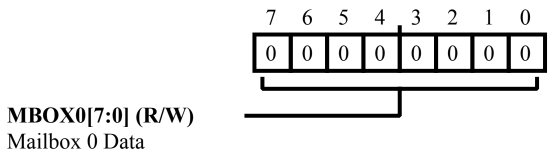


Figure 11-99: A2B_MBOX0B0 Register Diagram

Table 11-100: A2B_MBOX0B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX0	Mailbox 0 Data.

Mailbox 0 Byte 1 Register (Sub Only)

The `A2B_MBOX0B1` register represents byte 1 of Mailbox 0.

Address: 0x93

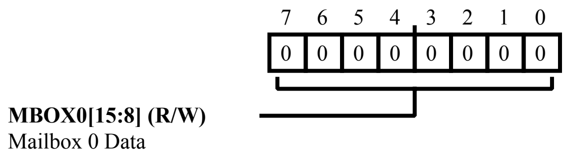


Figure 11-100: A2B_MBOX0B1 Register Diagram

Table 11-101: A2B_MBOX0B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX0	Mailbox 0 Data.

Mailbox 0 Byte 2 Register (Sub Only)

The A2B_MBOX0B2 register represents byte 2 of Mailbox 0.

Address: 0x94

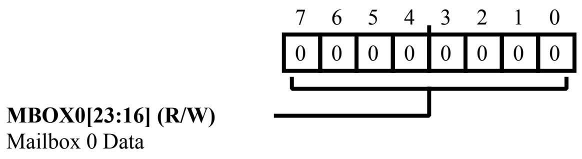


Figure 11-101: A2B_MBOX0B2 Register Diagram

Table 11-102: A2B_MBOX0B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX0	Mailbox 0 Data.

Mailbox 0 Byte 3 Register (Sub Only)

The A2B_MBOX0B3 register represents byte 3 of Mailbox 0.

Address: 0x95

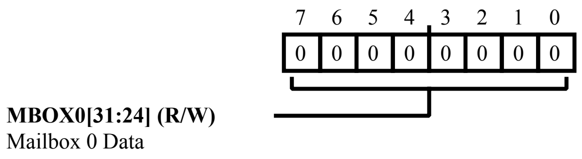


Figure 11-102: A2B_MBOX0B3 Register Diagram

Table 11-103: A2B_MBOX0B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX0	Mailbox 0 Data.

Mailbox 1 Control Register (Sub Only)

The A2B_MBOX1CTL register contains bits that control direction, message length and interrupts.

Address: 0x96

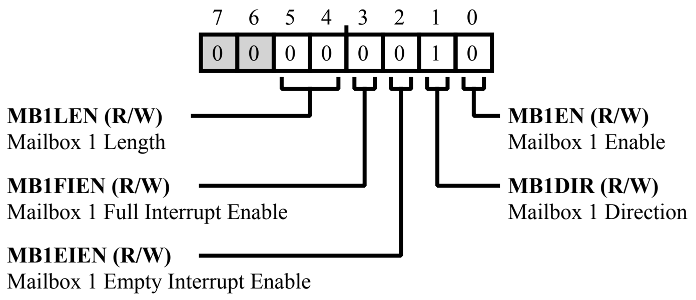


Figure 11-103: A2B_MBOX1CTL Register Diagram

Table 11-104: A2B_MBOX1CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/W)	MB1LEN	Mailbox 1 Length. The A2B_MBOX1CTL.MB1LEN bit field controls the length of Mailbox 1.	
		0	1 Byte
		1	2 Bytes
		2	3 Bytes
		3	4 Bytes
3 (R/W)	MB1FIEN	Mailbox 1 Full Interrupt Enable. The A2B_MBOX1CTL.MB1FIEN bit enables an interrupt which is generated when Mailbox 1 becomes full.	
		0	Mailbox 1 Interrupt on Full Disabled
		1	Mailbox 1 Interrupt on Full Enabled
2 (R/W)	MB1EIEN	Mailbox 1 Empty Interrupt Enable. The A2B_MBOX1CTL.MB1EIEN bit enables an interrupt which is generated when Mailbox 1 becomes empty.	
		0	Mailbox 1 Interrupt on Empty Disabled
		1	Mailbox 1 Interrupt on Empty Enabled
1 (R/W)	MB1DIR	Mailbox 1 Direction. The A2B_MBOX1CTL.MB1DIR bit controls the direction of Mailbox 1.	
		0	Mailbox 1 is Receive Mailbox
		1	Mailbox 1 is Transmit Mailbox

Table 11-104: A2B_MBOX1CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	MB1EN	Mailbox 1 Enable. Setting the `A2B_MBOX1CTL.MB1EN` bit enables Mailbox 1.	
		0	Mailbox 1 Disabled
		1	Mailbox 1 Enabled

Mailbox 1 Status Register (Sub Only)

The A2B_MBOX1STAT register reports the status of the configured mailbox interrupts.

Address: 0x97

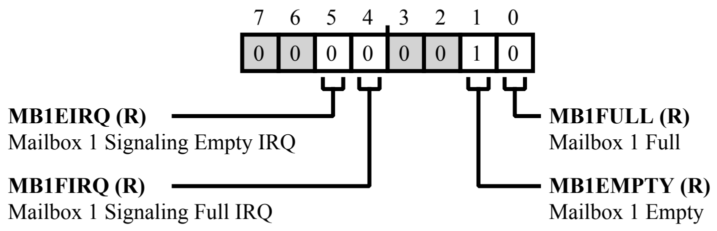


Figure 11-104: A2B_MBOX1STAT Register Diagram

Table 11-105: A2B_MBOX1STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	MB1EIRQ	Mailbox 1 Signaling Empty IRQ. The A2B_MBOX1STAT.MB1EIRQ bit indicates whether or not the Mailbox 1 empty interrupt is active.	
		0	Mailbox 1 Empty Interrupt Inactive
		1	Mailbox 1 Empty Interrupt Active
4 (R/NW)	MB1FIRQ	Mailbox 1 Signaling Full IRQ. The A2B_MBOX1STAT.MB1FIRQ bit indicates whether or not the Mailbox 1 full interrupt is active.	
		0	Mailbox 1 Full Interrupt Inactive
		1	Mailbox 1 Full Interrupt Active
1 (R/NW)	MB1EMPTY	Mailbox 1 Empty. The A2B_MBOX1STAT.MB1EMPTY bit indicates whether or not Mailbox 1 is empty.	
		0	Mailbox 1 Currently Not Empty
		1	Mailbox 1 Currently Empty
0 (R/NW)	MB1FULL	Mailbox 1 Full. The A2B_MBOX1STAT.MB1FULL bit indicates whether or not Mailbox 1 is full.	
		0	Mailbox 1 Currently Not Full
		1	Mailbox 1 Currently Full

Mailbox 1 Byte 0 Register (Sub Only)

The A2B_MBOX1B0 register represents byte 0 of Mailbox 1.

Address: 0x98

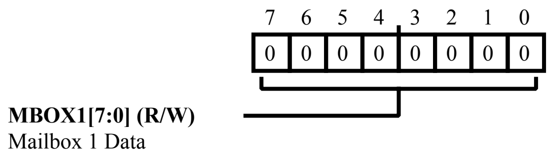


Figure 11-105: A2B_MBOX1B0 Register Diagram

Table 11-106: A2B_MBOX1B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX1	Mailbox 1 Data.

Mailbox 1 Byte 1 Register (Sub Only)

The A2B_MBOX1B1 register represents byte 1 of Mailbox 1.

Address: 0x99

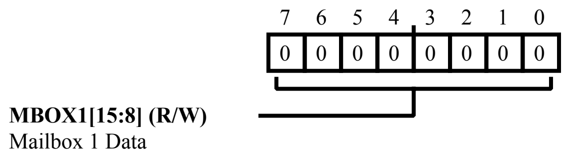


Figure 11-106: A2B_MBOX1B1 Register Diagram

Table 11-107: A2B_MBOX1B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX1	Mailbox 1 Data.

Mailbox 1 Byte 2 Register (Sub Only)

The `A2B_MBOX1B2` register represents byte 2 of Mailbox 1.

Address: 0x9A

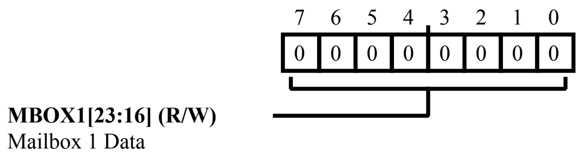


Figure 11-107: A2B_MBOX1B2 Register Diagram

Table 11-108: A2B_MBOX1B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX1	Mailbox 1 Data.

Mailbox 1 Byte 3 Register (Sub Only)

The A2B_MBOX1B3 register represents byte 3 of Mailbox 1.

Address: 0x9B

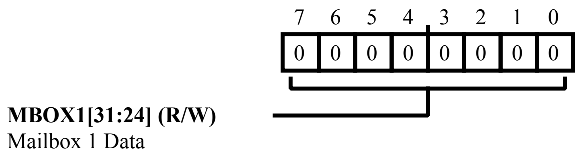


Figure 11-108: A2B_MBOX1B3 Register Diagram

Table 11-109: A2B_MBOX1B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MBOX1	Mailbox 1 Data.

Switch Control Register 2

The A2B_SWCTL2 register provides further control bits for switching of A^2B bus power onto the downstream B-side of the A^2B bus.

Address: 0xA0

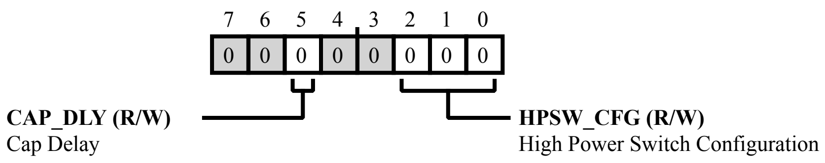


Figure 11-109: A2B_SWCTL2 Register Diagram

Table 11-110: A2B_SWCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	CAP_DLY	Cap Delay. The A2B_SWCTL2.CAP_DLY bit indicates the DC bias charging time.	
		0	Default DC Bias. Suitable for all nodes.
		1	Reduced Delay. Used to reduce DC bias charging ramp time when discovering nodes with low (<100 uF) capacitance.
2:0 (R/W)	HPSW_CFG	High Power Switch Configuration. The A2B_SWCTL2.HPSW_CFG field indicates the power configuration.	
		4	High Power. AD2437

Switch Status Register 2

The A2B_SWSTAT2 register provides further status bits for switching of A^{2}B bus power onto the downstream B-side of the A^{2}B bus.

Address: 0xA5

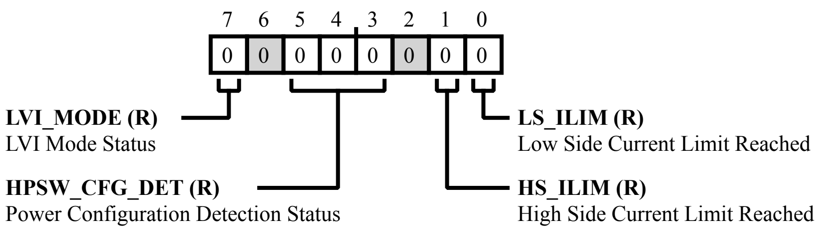

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
LVI_MODE (R)
LVI Mode Status
HPSW_CFG_DET (R)
Power Configuration Detection Status
LS_ILIM (R)
Low Side Current Limit Reached
HS_ILIM (R)
High Side Current Limit Reached

Figure 11-110: A2B_SWSTAT2 Register Diagram

Table 11-111: A2B_SWSTAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	LVI_MODE	LVI Mode Status. The A2B_SWSTAT2.LVI_MODE bit indicates whether the Low Voltage Input (LVI) mode is detected.	
		0	Mode Not Detected
		1	Mode Detected
5:3 (R/NW)	HPSW_CFG_DET	Power Configuration Detection Status. The A2B_SWSTAT2.HPSW_CFG_DET field indicates the detected power configuration.	
		0	Standard Power Configuration
		1	Reserved
		4	High Power Configuration
1 (R/NW)	HS_ILIM	High Side Current Limit Reached. The A2B_SWSTAT2.HS_ILIM bit indicates when the current limit on the high side FET is reached.	
0 (R/NW)	LS_ILIM	Low Side Current Limit Reached. The A2B_SWSTAT2.LS_ILIM bit indicates when the current limit on the low side FET is reached.	

SPI Data Tunnel Last Command Register

The A2B_SPIDTLCMD register provides the last command seen on the data tunnel.

Address: 0xAF

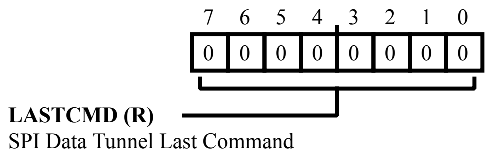


Figure 11-111: A2B_SPIDTLCMD Register Diagram

Table 11-112: A2B_SPIDTLCMD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	LASTCMD	SPI Data Tunnel Last Command.

SPI Configuration Register

The A2B_SPICFG register is used to configure the SPI peripheral.

Address: 0xB0

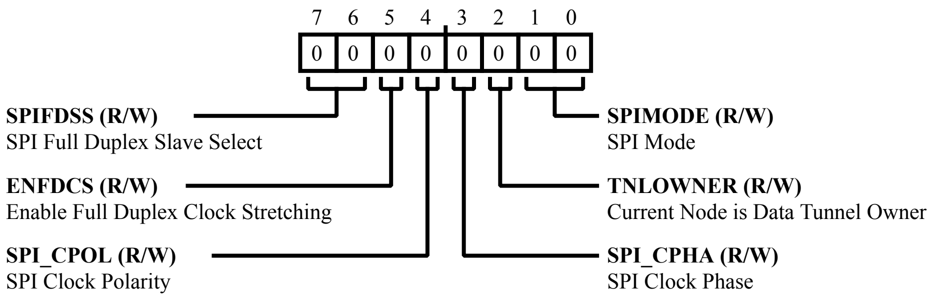


Figure 11-112: A2B_SPICFG Register Diagram

Table 11-113: A2B_SPICFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/W)	SPIFDSS	SPI Full Duplex Slave Select. The `A2B_SPICFG.SPIFDSS` bit field controls how full duplex transactions are delivered to a data tunnel owner via the SPI interface.	
		0	Use SPISS0n (ADR1) and command bytes for full duplex transaction
		1	Use SPISS1n (SIO2) and no command bytes for full duplex transaction
		2	Use SPISS2n (ADR2) and no command bytes for full duplex transaction
5 (R/W)	ENFDCS	Enable Full Duplex Clock Stretching.	
4 (R/W)	SPI_CPOL	SPI Clock Polarity.	
3 (R/W)	SPI_CPHA	SPI Clock Phase.	
2 (R/W)	TNLOWNER	Current Node is Data Tunnel Owner.	
1:0 (R/W)	SPIMODE	SPI Mode.	
		0	SPI Slave Mode
		1	Data Tunnel Target (SPI Master Mode)
		2	SPI Module Disabled

SPI Status Register

The A2B_SPISTAT register provides status from the SPI peripheral.

Address: 0xB1

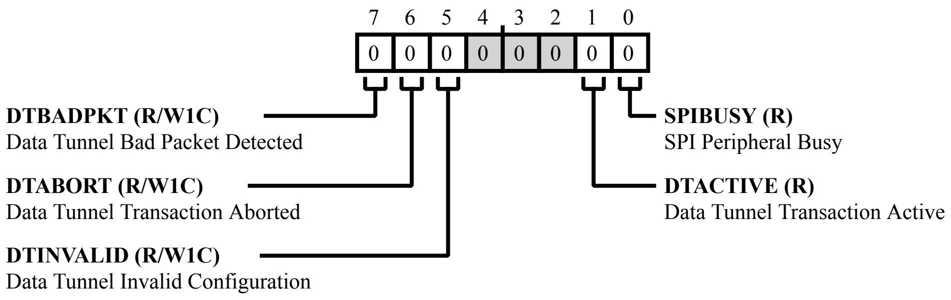

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
DTBADPKT (R/W1C)
Data Tunnel Bad Packet Detected
DTABORT (R/W1C)
Data Tunnel Transaction Aborted
DTINVALID (R/W1C)
Data Tunnel Invalid Configuration
SPIBUSY (R)
SPI Peripheral Busy
DTACTIVE (R)
Data Tunnel Transaction Active

Figure 11-113: A2B_SPISTAT Register Diagram

Table 11-114: A2B_SPISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	DTBADPKT	Data Tunnel Bad Packet Detected.
6 (R/W1C)	DTABORT	Data Tunnel Transaction Aborted.
5 (R/W1C)	DTINVALID	Data Tunnel Invalid Configuration. High if the data tunnel is enabled with an invalid configuration.
1 (R/NW)	DTACTIVE	Data Tunnel Transaction Active.
0 (R/NW)	SPIBUSY	SPI Peripheral Busy.

SPI Clock Divide Register

The A2B_SPICKDIV register is used to configure frequency of the clock generated on SCK when the SPI peripheral is operating as an SPI master.

Address: 0xB2

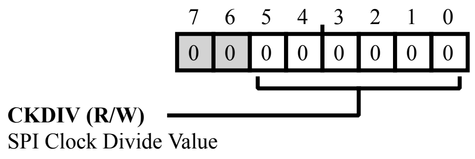


Figure 11-114: A2B_SPICKDIV Register Diagram

Table 11-115: A2B_SPICKDIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5:0 (R/W)	CKDIV	SPI Clock Divide Value.	
		0	Reserved
		1	Reserved
		2	Reserved
		3	SPI Master SCK period is bit time times 4 (nominally 12.288MHz)
		4	SPI Master SCK period is bit time times 5 (nominally 9.8304MHz)
		63	SPI Master SCK period is bit time times 64 (nominally 0.768MHz)

SPI Full Duplex Size Register

The A2B_SPIFDSIZE register configures the size of a full duplex transaction in the SPI tunnel owner when A2B_SPICFG.SPIFDSS is non-zero.

Address: 0xB3

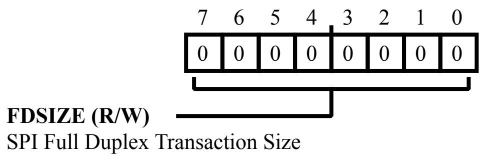


Figure 11-115: A2B_SPIFDSIZE Register Diagram

Table 11-116: A2B_SPIFDSIZE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	FDSIZE	SPI Full Duplex Transaction Size.	
		0	Full Duplex Transaction Size of 1 Byte
		1	Full Duplex Transaction Size of 2 Bytes
		2	Full Duplex Transaction Size of 3 Bytes
		3	Full Duplex Transaction Size of 4 Bytes
		254	Full Duplex Transaction Size of 255 Bytes
		255	Full Duplex Transaction Size of 256 Bytes

SPI Full Duplex Target Register

The A2B_SPIFDTARG register configures the target of a full duplex transaction in the SPI tunnel owner when A2B_SPICFG.SPIFDSS is non-zero.

Address: 0xB4

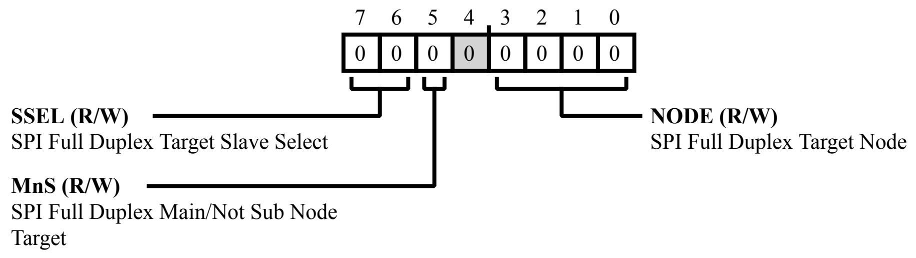

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
SSEL (R/W)
SPI Full Duplex Target Slave Select
MnS (R/W)
SPI Full Duplex Main/Not Sub Node Target
NODE (R/W)
SPI Full Duplex Target Node

Figure 11-116: A2B_SPIFDTARG Register Diagram

Table 11-117: A2B_SPIFDTARG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/W)	SSEL	SPI Full Duplex Target Slave Select. Indicates the slave select to be used in the data tunnel target during a register based full duplex transaction.	
		0	ADR1 is the target slave select for register based full duplex transaction
		1	SIO2 is the target slave select for register based full duplex transaction
		2	ADR2 is the target slave select for register based full duplex transaction
5 (R/W)	MNS	SPI Full Duplex Main/Not Sub Node Target. Indicates whether the data tunnel target is the main node or a subordinate node for a register based full duplex transaction.	
3:0 (R/W)	NODE	SPI Full Duplex Target Node. Provides the subordinate node ID of the data tunnel target for a register based full duplex transaction. Unused if MnS is one.	

SPI Pin Configuration Register

The A2B_SPIPINCFG register configures pin mapping for the SPI peripheral.

Address: 0xB5

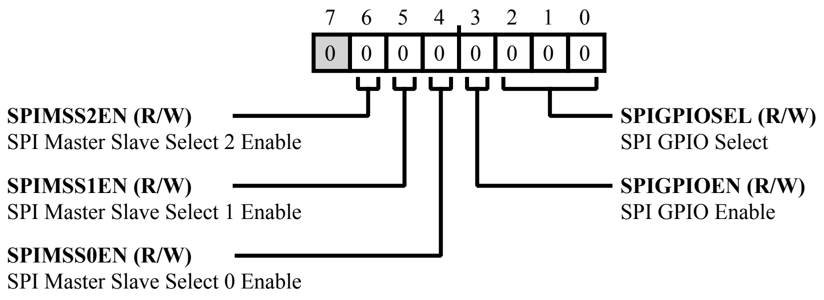


Figure 11-117: A2B_SPIPINCFG Register Diagram

Table 11-118: A2B_SPIPINCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	SPIMSS2EN	SPI Master Slave Select 2 Enable.	
5 (R/W)	SPIMSS1EN	SPI Master Slave Select 1 Enable.	
4 (R/W)	SPIMSS0EN	SPI Master Slave Select 0 Enable.	
3 (R/W)	SPIGPIOEN	SPI GPIO Enable. The A2B_SPIPINCFG.SPIGPIOEN bit enables the value of A2B_SPISTAT.SPIBUSY to be driven to a GPIO pin in a node where A2B_SPICFG.SPIMODE ==0. This feature has lower priority on the pin than sustain to GPIO or GPIO over distance.	
2:0 (R/W)	SPIGPIOSEL	SPI GPIO Select. The A2B_SPIPINCFG.SPIGPIOSEL bit field determines the pin used for SPIBUSY on GPIO assuming A2B_SPIPINCFG.SPIGPIOEN is set and A2B_SPICFG.SPIMODE==0.	
		0	Use GPIO0 for SPIBUSY when enabled
		1	Use GPIO1 for SPIBUSY when enabled
		2	Use GPIO2 for SPIBUSY when enabled
		3	Use GPIO3 for SPIBUSY when enabled
		4	Use GPIO4 for SPIBUSY when enabled
		5	Use GPIO5 for SPIBUSY when enabled

Table 11-118: A2B_SPIPINCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Use GPIO6 for SPIBUSY when enabled
		7	Use GPIO7 for SPIBUSY when enabled

SPI Interrupt Register

The A2B_SPIINT register indicates pending interrupts from the SPI peripheral.

Address: 0xB6

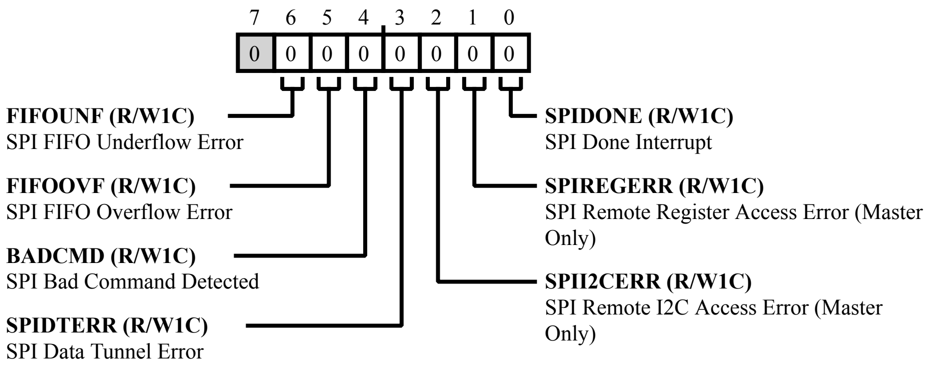


Figure 11-118: A2B_SPIINT Register Diagram

Table 11-119: A2B_SPIINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W1C)	FIFOUNF	SPI FIFO Underflow Error.
5 (R/W1C)	FIFOOVF	SPI FIFO Overflow Error.
4 (R/W1C)	BADCMD	SPI Bad Command Detected.
3 (R/W1C)	SPIDTERR	SPI Data Tunnel Error.
2 (R/W1C)	SPII2CERR	SPI Remote I2C Access Error (Master Only).
1 (R/W1C)	SPIREGERR	SPI Remote Register Access Error (Master Only).
0 (R/W1C)	SPIDONE	SPI Done Interrupt.

SPI Interrupt Mask Register

The A2B_SPIMSK register contains mask bits to enable pending SPI interrupts to be processed.

Address: 0xB7

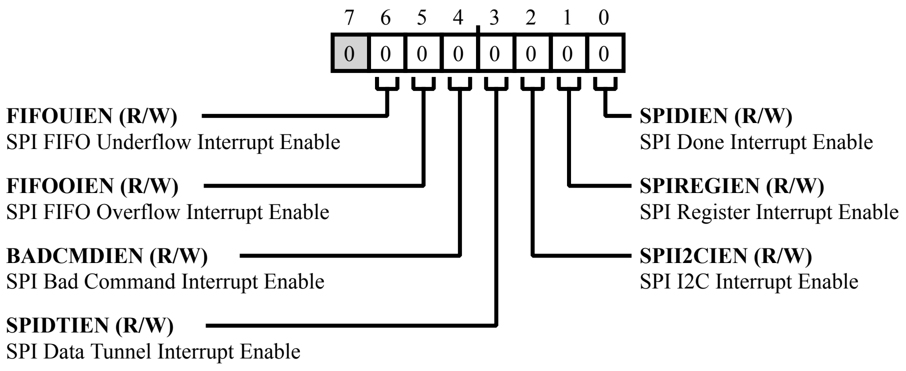

7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0
FIFOUIEN (R/W)
SPI FIFO Underflow Interrupt Enable
FIFOOIEN (R/W)
SPI FIFO Overflow Interrupt Enable
BADCMDIEN (R/W)
SPI Bad Command Interrupt Enable
SPIDTIEN (R/W)
SPI Data Tunnel Interrupt Enable
SPIDIEN (R/W)
SPI Done Interrupt Enable
SPIREGIEN (R/W)
SPI Register Interrupt Enable
SPII2CIEN (R/W)
SPI I2C Interrupt Enable

Figure 11-119: A2B_SPIMSK Register Diagram

Table 11-120: A2B_SPIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W)	FIFOUIEN	SPI FIFO Underflow Interrupt Enable.
5 (R/W)	FIFOOIEN	SPI FIFO Overflow Interrupt Enable.
4 (R/W)	BADCMDIEN	SPI Bad Command Interrupt Enable.
3 (R/W)	SPIDTIEN	SPI Data Tunnel Interrupt Enable.
2 (R/W)	SPII2CIEN	SPI I2C Interrupt Enable.
1 (R/W)	SPIREGIEN	SPI Register Interrupt Enable.
0 (R/W)	SPIDIEN	SPI Done Interrupt Enable.

I2S/TDM RX Mask 0 Register

The A2B_RXMASK0 register defines bits 7 to 0 of the 32-bit RXMASK field. The RXMASK field defines the I2S/TDM data channels which are received by the local node.

Address: 0xB8

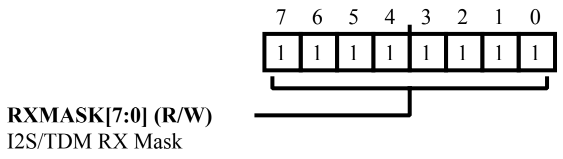


Figure 11-120: A2B_RXMASK0 Register Diagram

Table 11-121: A2B_RXMASK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 1 Register

The A2B_RXMASK1 register defines bits 15 to 8 of the 32-bit RXMASK field. The RXMASK field defines the I2S/TDM data channels which are received by the local node.

Address: 0xB9

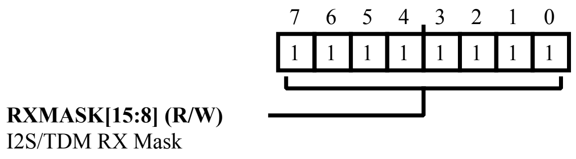


Figure 11-121: A2B_RXMASK1 Register Diagram

Table 11-122: A2B_RXMASK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 2 Register

The A2B_RXMASK2 register defines bits 23 to 16 of the 32-bit RXMASK field. The RXMASK field defines the I2S/TDM data channels which are received by the local node.

Address: 0xBA

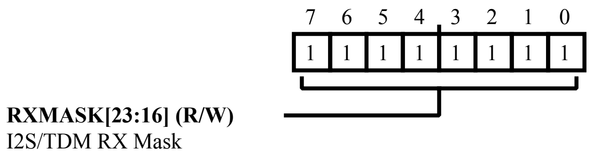


Figure 11-122: A2B_RXMASK2 Register Diagram

Table 11-123: A2B_RXMASK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 3 Register

The A2B_RXMASK3 register defines bits 31 to 24 of the 32-bit RXMASK field. The RXMASK field defines the I2S/TDM data channels which are received by the local node.

Address: 0xBB

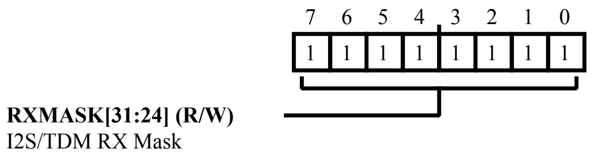


Figure 11-123: A2B_RXMASK3 Register Diagram

Table 11-124: A2B_RXMASK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 4 Register

Address: 0xBC

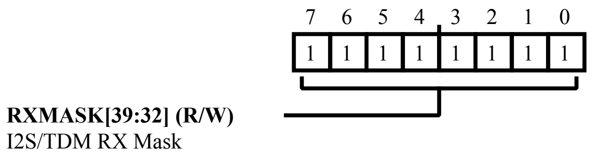


Figure 11-124: A2B_RXMASK4 Register Diagram

Table 11-125: A2B_RXMASK4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 5 Register

Address: 0xBD

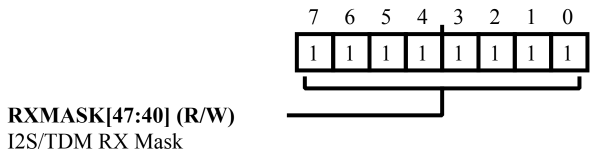


Figure 11-125: A2B_RXMASK5 Register Diagram

Table 11-126: A2B_RXMASK5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 6 Register

Address: 0xBE

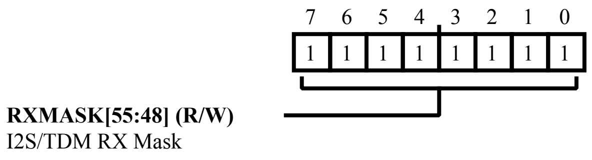


Figure 11-126: A2B_RXMASK6 Register Diagram

Table 11-127: A2B_RXMASK6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

I2S/TDM RX Mask 7 Register

Address: 0xBF

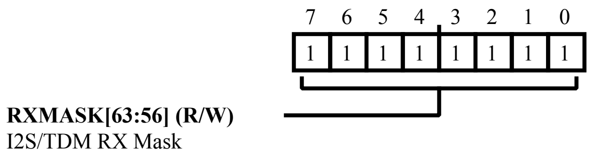


Figure 11-127: A2B_RXMASK7 Register Diagram

Table 11-128: A2B_RXMASK7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RXMASK	I2S/TDM RX Mask.

Serial TX Crossbar Register 0

The TXXBARn registers provide a mapping from received bus slots to I2S/TDM TX channels. TXXBAR0 controls the mapping of the first received bus slot, TXXBAR1 of the second received bus slot, and so on. In an intermediate subordinate node, downstream slots are received before upstream slots.

Address: 0xC0

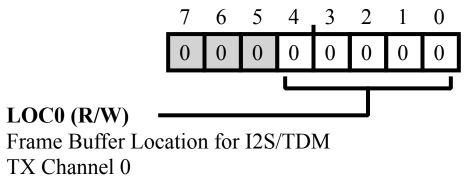


Figure 11-128: A2B_TXXBAR0 Register Diagram

Table 11-129: A2B_TXXBAR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC0	Frame Buffer Location for I2S/TDM TX Channel 0. The `A2B_TXXBAR0.LOC0` bit field defines the frame buffer location used for I2S/TDM TX channel 0.

Serial TX Crossbar Register 1

Address: 0xC1

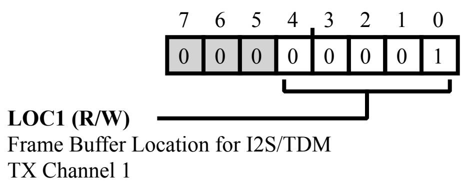


Figure 11-129: A2B_TXXBAR1 Register Diagram

Table 11-130: A2B_TXXBAR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC1	Frame Buffer Location for I2S/TDM TX Channel 1. The `A2B_TXXBAR1.LOC1` bit field defines the frame buffer location used for I2S/TDM TX channel 1.

Serial TX Crossbar Register 2

Address: 0xC2

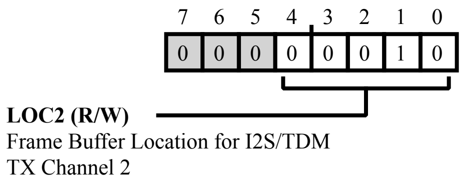


Figure 11-130: A2B_TXXBAR2 Register Diagram

Table 11-131: A2B_TXXBAR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC2	Frame Buffer Location for I2S/TDM TX Channel 2. The `A2B_TXXBAR2.LOC2` bit field defines the frame buffer location used for I2S/TDM TX channel 2.

Serial TX Crossbar Register 3

Address: 0xC3

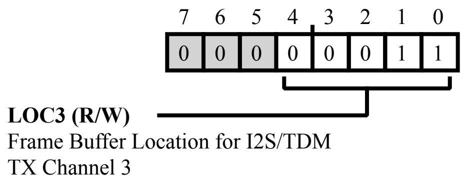


Figure 11-131: A2B_TXXBAR3 Register Diagram

Table 11-132: A2B_TXXBAR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC3	Frame Buffer Location for I2S/TDM TX Channel 3. The `A2B_TXXBAR3.LOC3` bit field defines the frame buffer location used for I2S/TDM TX channel 3.

Serial TX Crossbar Register 4

Address: 0xC4

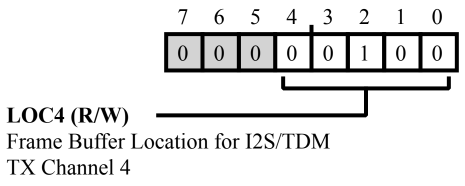


Figure 11-132: A2B_TXXBAR4 Register Diagram

Table 11-133: A2B_TXXBAR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC4	Frame Buffer Location for I2S/TDM TX Channel 4. The `A2B_TXXBAR4.LOC4` bit field defines the frame buffer location used for I2S/TDM TX channel 4.

Serial TX Crossbar Register 5

Address: 0xC5

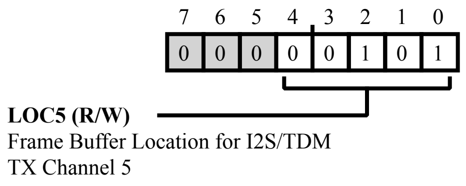


Figure 11-133: A2B_TXXBAR5 Register Diagram

Table 11-134: A2B_TXXBAR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC5	Frame Buffer Location for I2S/TDM TX Channel 5. The `A2B_TXXBAR5.LOC5` bit field defines the frame buffer location used for I2S/TDM TX channel 5.

Serial TX Crossbar Register 6

Address: 0xC6

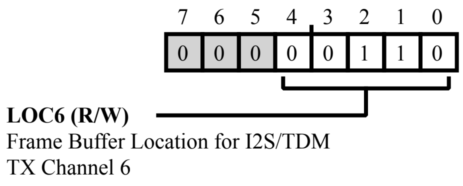


Figure 11-134: A2B_TXXBAR6 Register Diagram

Table 11-135: A2B_TXXBAR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC6	Frame Buffer Location for I2S/TDM TX Channel 6. The `A2B_TXXBAR6.LOC6` bit field defines the frame buffer location used for I2S/TDM TX channel 6.

Serial TX Crossbar Register 7

Address: 0xC7

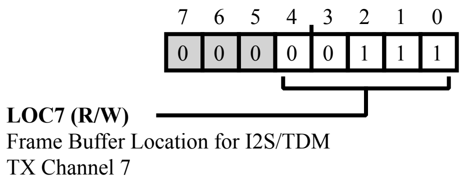


Figure 11-135: A2B_TXXBAR7 Register Diagram

Table 11-136: A2B_TXXBAR7 Register Fields

Bit No. (Access)	**Bit Name**	**Description/Enumeration**
4:0 (R/W)	LOC7	Frame Buffer Location for I2S/TDM TX Channel 7. The `A2B_TXXBAR7.LOC7` bit field defines the frame buffer location used for I2S/TDM TX channel 7.

Serial TX Crossbar Register 8

Address: 0xC8

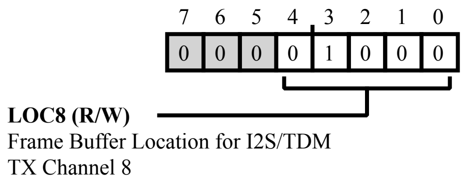


Figure 11-136: A2B_TXXBAR8 Register Diagram

Table 11-137: A2B_TXXBAR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC8	Frame Buffer Location for I2S/TDM TX Channel 8. The `A2B_TXXBAR8.LOC8` bit field defines the frame buffer location used for I2S/TDM TX channel 8.

Serial TX Crossbar Register 9

Address: 0xC9

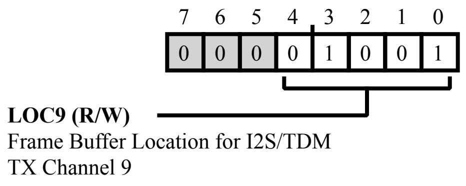


Figure 11-137: A2B_TXXBAR9 Register Diagram

Table 11-138: A2B_TXXBAR9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC9	Frame Buffer Location for I2S/TDM TX Channel 9. The `A2B_TXXBAR9.LOC9` bit field defines the frame buffer location used for I2S/TDM TX channel 9.

Serial TX Crossbar Register 10

Address: 0xCA

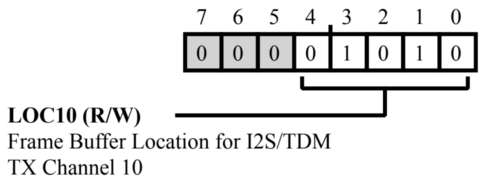


Figure 11-138: A2B_TXXBAR10 Register Diagram

Table 11-139: A2B_TXXBAR10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC10	Frame Buffer Location for I2S/TDM TX Channel 10. The `A2B_TXXBAR10.LOC10` bit field defines the frame buffer location used for I2S/TDM TX channel 10.

Serial TX Crossbar Register 11

Address: 0xCB

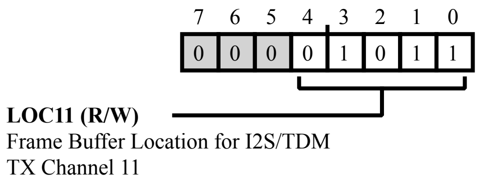


Figure 11-139: A2B_TXXBAR11 Register Diagram

Table 11-140: A2B_TXXBAR11 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC11	Frame Buffer Location for I2S/TDM TX Channel 11. The `A2B_TXXBAR11.LOC11` bit field defines the frame buffer location used for I2S/TDM TX channel 11.

Serial TX Crossbar Register 12

Address: 0xCC

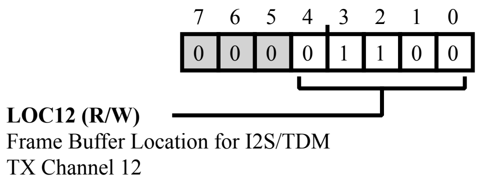


Figure 11-140: A2B_TXXBAR12 Register Diagram

Table 11-141: A2B_TXXBAR12 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC12	Frame Buffer Location for I2S/TDM TX Channel 12. The `A2B_TXXBAR12.LOC12` bit field defines the frame buffer location used for I2S/TDM TX channel 12.

Serial TX Crossbar Register 13

Address: 0xCD

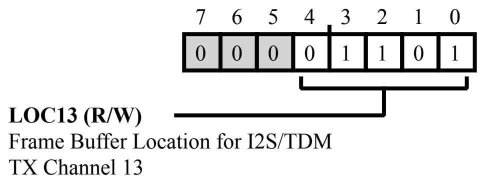


Figure 11-141: A2B_TXXBAR13 Register Diagram

Table 11-142: A2B_TXXBAR13 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC13	Frame Buffer Location for I2S/TDM TX Channel 13. The `A2B_TXXBAR13.LOC13` bit field defines the frame buffer location used for I2S/TDM TX channel 13.

Serial TX Crossbar Register 14

Address: 0xCE

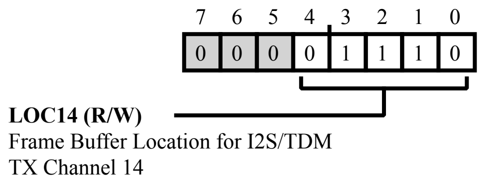


Figure 11-142: A2B_TXXBAR14 Register Diagram

Table 11-143: A2B_TXXBAR14 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC14	Frame Buffer Location for I2S/TDM TX Channel 14. The `A2B_TXXBAR14.LOC14` bit field defines the frame buffer location used for I2S/TDM TX channel 14.

Serial TX Crossbar Register 15

Address: 0xCF

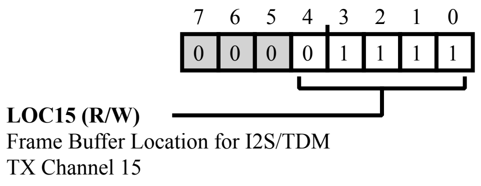


Figure 11-143: A2B_TXXBAR15 Register Diagram

Table 11-144: A2B_TXXBAR15 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC15	Frame Buffer Location for I2S/TDM TX Channel 15. The `A2B_TXXBAR15.LOC15` bit field defines the frame buffer location used for I2S/TDM TX channel 15.

Serial TX Crossbar Register 16

Address: 0xD0

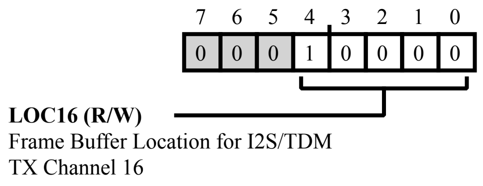


Figure 11-144: A2B_TXXBAR16 Register Diagram

Table 11-145: A2B_TXXBAR16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC16	Frame Buffer Location for I2S/TDM TX Channel 16. The `A2B_TXXBAR16.LOC16` bit field defines the frame buffer location used for I2S/TDM TX channel 16.

Serial TX Crossbar Register 17

Address: 0xD1

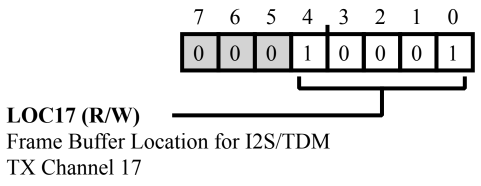


Figure 11-145: A2B_TXXBAR17 Register Diagram

Table 11-146: A2B_TXXBAR17 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC17	Frame Buffer Location for I2S/TDM TX Channel 17. The `A2B_TXXBAR17.LOC17` bit field defines the frame buffer location used for I2S/TDM TX channel 17.

Serial TX Crossbar Register 18

Address: 0xD2

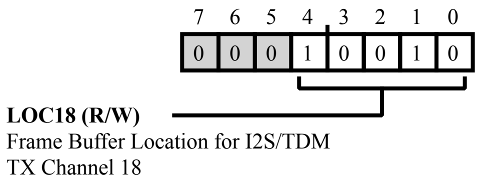


Figure 11-146: A2B_TXXBAR18 Register Diagram

Table 11-147: A2B_TXXBAR18 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC18	Frame Buffer Location for I2S/TDM TX Channel 18. The `A2B_TXXBAR18.LOC18` bit field defines the frame buffer location used for I2S/TDM TX channel 18.

Serial TX Crossbar Register 19

Address: 0xD3

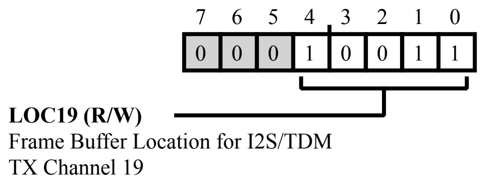


Figure 11-147: A2B_TXXBAR19 Register Diagram

Table 11-148: A2B_TXXBAR19 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC19	Frame Buffer Location for I2S/TDM TX Channel 19. The `A2B_TXXBAR19.LOC19` bit field defines the frame buffer location used for I2S/TDM TX channel 19.

Serial TX Crossbar Register 20

Address: 0xD4

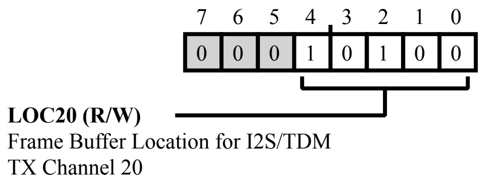


Figure 11-148: A2B_TXXBAR20 Register Diagram

Table 11-149: A2B_TXXBAR20 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC20	Frame Buffer Location for I2S/TDM TX Channel 20. The `A2B_TXXBAR20.LOC20` bit field defines the frame buffer location used for I2S/TDM TX channel 20.

Serial TX Crossbar Register 21

Address: 0xD5

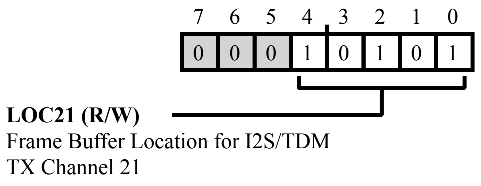


Figure 11-149: A2B_TXXBAR21 Register Diagram

Table 11-150: A2B_TXXBAR21 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC21	Frame Buffer Location for I2S/TDM TX Channel 21. The `A2B_TXXBAR21.LOC21` bit field defines the frame buffer location used for I2S/TDM TX channel 21.

Serial TX Crossbar Register 22

Address: 0xD6

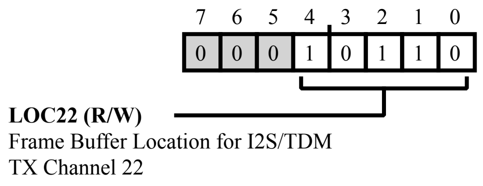


Figure 11-150: A2B_TXXBAR22 Register Diagram

Table 11-151: A2B_TXXBAR22 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC22	Frame Buffer Location for I2S/TDM TX Channel 22. The `A2B_TXXBAR22.LOC22` bit field defines the frame buffer location used for I2S/TDM TX channel 22.

Serial TX Crossbar Register 23

Address: 0xD7

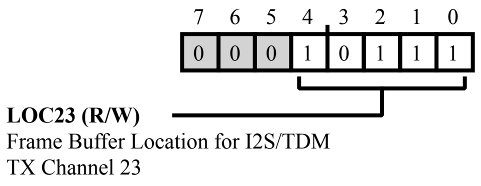


Figure 11-151: A2B_TXXBAR23 Register Diagram

Table 11-152: A2B_TXXBAR23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC23	Frame Buffer Location for I2S/TDM TX Channel 23. The `A2B_TXXBAR23.LOC23` bit field defines the frame buffer location used for I2S/TDM TX channel 23.

Serial TX Crossbar Register 24

Address: 0xD8

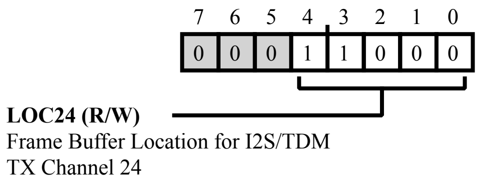


Figure 11-152: A2B_TXXBAR24 Register Diagram

Table 11-153: A2B_TXXBAR24 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC24	Frame Buffer Location for I2S/TDM TX Channel 24. The `A2B_TXXBAR24.LOC24` bit field defines the frame buffer location used for I2S/TDM TX channel 24.

Serial TX Crossbar Register 25

Address: 0xD9

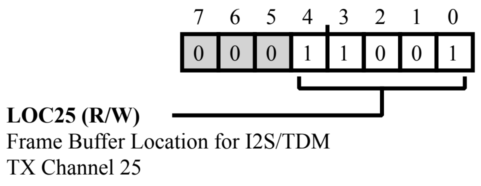


Figure 11-153: A2B_TXXBAR25 Register Diagram

Table 11-154: A2B_TXXBAR25 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC25	Frame Buffer Location for I2S/TDM TX Channel 25. The `A2B_TXXBAR25.LOC25` bit field defines the frame buffer location used for I2S/TDM TX channel 25.

Serial TX Crossbar Register 26

Address: 0xDA

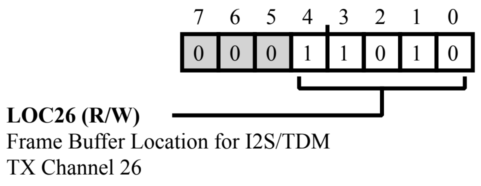


Figure 11-154: A2B_TXXBAR26 Register Diagram

Table 11-155: A2B_TXXBAR26 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC26	Frame Buffer Location for I2S/TDM TX Channel 26. The `A2B_TXXBAR26.LOC26` bit field defines the frame buffer location used for I2S/TDM TX channel 26.

Serial TX Crossbar Register 27

Address: 0xDB

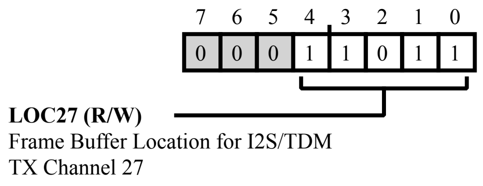


Figure 11-155: A2B_TXXBAR27 Register Diagram

Table 11-156: A2B_TXXBAR27 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC27	Frame Buffer Location for I2S/TDM TX Channel 27. The `A2B_TXXBAR27.LOC27` bit field defines the frame buffer location used for I2S/TDM TX channel 27.

Serial TX Crossbar Register 28

Address: 0xDC

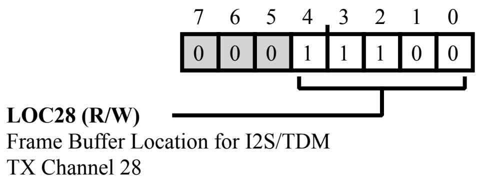


Figure 11-156: A2B_TXXBAR28 Register Diagram

Table 11-157: A2B_TXXBAR28 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC28	Frame Buffer Location for I2S/TDM TX Channel 28. The `A2B_TXXBAR28.LOC28` bit field defines the frame buffer location used for I2S/TDM TX channel 28.

Serial TX Crossbar Register 29

Address: 0xDD

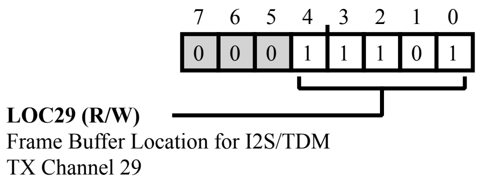


Figure 11-157: A2B_TXXBAR29 Register Diagram

Table 11-158: A2B_TXXBAR29 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC29	Frame Buffer Location for I2S/TDM TX Channel 29. The `A2B_TXXBAR29.LOC29` bit field defines the frame buffer location used for I2S/TDM TX channel 29.

Serial TX Crossbar Register 30

Address: 0xDE

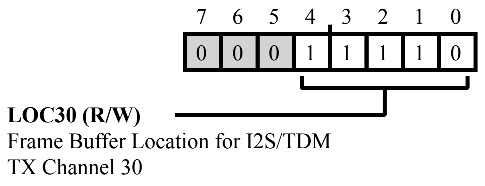


Figure 11-158: A2B_TXXBAR30 Register Diagram

Table 11-159: A2B_TXXBAR30 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC30	Frame Buffer Location for I2S/TDM TX Channel 30. The `A2B_TXXBAR30.LOC30` bit field defines the frame buffer location used for I2S/TDM TX channel 30.

Serial TX Crossbar Register 31

Address: 0xDF

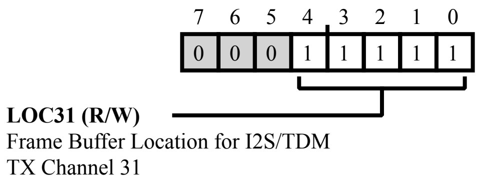


Figure 11-159: A2B_TXXBAR31 Register Diagram

Table 11-160: A2B_TXXBAR31 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	LOC31	Frame Buffer Location for I2S/TDM TX Channel 31. The `A2B_TXXBAR31.LOC31` bit field defines the frame buffer location used for I2S/TDM TX channel 31.

MMR Page Register

The A2B_MMRPAGE register provides a page address for MMR register accesses. This register is written to 0x01 to access the VMTR and PWM registers. All other registers are in MMR page 0. This register should be 0 while accessing the page 0 registers. Regardless of the value of this register, a write to address 0xE0 always updates its value. MMR page values other than 0 and 1 are not supported and must not be written.

Address: 0xE0

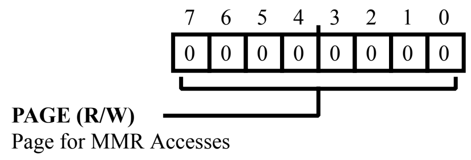


Figure 11-160: A2B_MMRPAGE Register Diagram

Table 11-161: A2B_MMRPAGE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PAGE	Page for MMR Accesses.

Enable Voltage Measurement

The A2B_VMTR_VEN register enables voltage monitoring based on pin input.

Address: 0x100

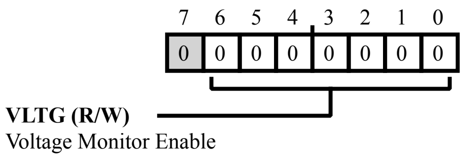


Figure 11-161: A2B_VMTR_VEN Register Diagram

Table 11-162: A2B_VMTR_VEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	VLTG	Voltage Monitor Enable. The A2B_VMTR_VEN.VLTG bit field enables voltage monitoring when A2B_VMTR_VEN is set. If enabled, the VMTR block samples the voltage level present on the associated pins and updates the A2B_VMTR_VTLGn register with the measured value in each superframe. Each bit is associated with a different input as follows: Bit 0 enables the VIN voltage (VIN-GND) measurement. Bit 1 enables the VBUS voltage (VBUS-GND) measurement. Bit 2 enables the IOVDD voltage (IOVDD-GND) measurement. Bit 3 enables the TRXVDD voltage (TRXVDD-GND) measurement. Bit 4 enables the DVDD voltage (DVDD-GND) measurement. Bit 5 enables the low-side current (ISENSEN-VSENSEN) measurement. Bit 6 enables the high-side current (VBUS-ISENSEP) measurement.

Min / Max Error Interrupt Enable

The `A2B_VMTR_INTEN` register enables an interrupt for corresponding pin inputs based on maximum and minimum voltage thresholds.

Address: 0x101

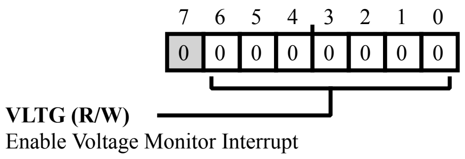


Figure 11-162: A2B_VMTR_INTEN Register Diagram

Table 11-163: A2B_VMTR_INTEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	VLTG	Enable Voltage Monitor Interrupt. Interrupts are generated if the measured voltage is less than the threshold configured in the A2B_VMTR_VMINn register or higher than the threshold configured in the A2B_VMTR_VMAXn register.. Each bit enables the interrupt for a different input as follows: Bit 0 enables the interrupt for VIN voltage (VIN-GND) measurement. Bit 1 enables the interrupt for VBUS voltage (VBUS-GND) measurement. Bit 2 enables the interrupt for IOVDD voltage (IOVDD-GND) measurement. Bit 3 enables the interrupt for TRXVDD voltage (TRXVDD-GND) measurement. Bit 4 enables the interrupt for DVDD voltage (DVDD-GND) measurement. Bit 5 enables the interrupt for the low-side current (ISENSEN-VSENSEN) measurement. Bit 6 enables the interrupt for the high-side current (VBUS-ISENSEP) measurement.

VMAX Check Result

When enabled, the VMAX[n] check updates the MXERR[n] bit at the same time A2B_VMTR_VLTGn is loaded.

Address: 0x102

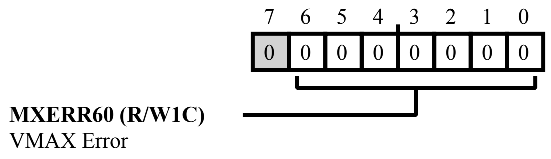


Figure 11-163: A2B_VMTR_MXSTAT Register Diagram

Table 11-164: A2B_VMTR_MXSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W1C)	MXERR60	VMAX Error. The MXERR[6:0] bit field is set when the measured voltage in A2B_VMTR_VLTG[n] is higher than the threshold configured in the A2B_VMTR_VMAX[n] register. Each bit indicates the active MXERR interrupt for a different input as follows: Bit 0 is set (=1) when the measured VIN voltage (`A2B_VMTR_VLTG0.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX0` register. Bit 1 is set (=1) when the measured VBUS voltage (`A2B_VMTR_VLTG1.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX1` register. Bit 2 is set (=1) when the measured IOVDD voltage (`A2B_VMTR_VLTG2.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX2` register. Bit 3 is set (=1) when the measured TRXVDD voltage (`A2B_VMTR_VLTG3.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX3` register. Bit 4 is set (=1) when the measured DVDD voltage (`A2B_VMTR_VLTG4.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX4` register. Bit 5 is set (=1) when the measured low-side current (`A2B_VMTR_VLTG5.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX5` register. Bit 6 is set (=1) when the measured high-side current (`A2B_VMTR_VLTG6.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMAX6`register.

VMIN Check Result

When enabled, the VMIN[n] check updates the MNERR[n] bit at the same time VMTR_VLTGn is loaded.

Address: 0x103

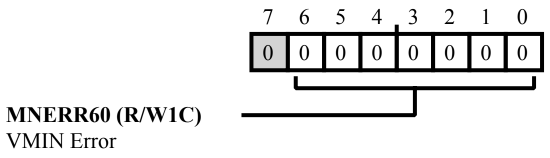


Figure 11-164: A2B_VMTR_MNSTAT Register Diagram

Table 11-165: A2B_VMTR_MNSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W1C)	MNERR60	VMIN Error. The MNERR[6:0] bit field is set when the measured voltage A2B_VMTR_VLTG[n] is lower than the threshold configured in the A2B_VMTR_VMIN[n] register. Each bit indicates the active MXERR interrupt for a different input as follows: Bit 0 is set (=1) when the measured VIN voltage (`A2B_VMTR_VLTG0.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN0` register. Bit 1 is set (=1) when the measured VBUS voltage (`A2B_VMTR_VLTG1.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN1` register. Bit 2 is set (=1) when the measured IOVDD voltage (`A2B_VMTR_VLTG2.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN2` register. Bit 3 is set (=1) when the measured TRXVDD voltage (`A2B_VMTR_VLTG3.VLTG`) is higher than the threshold configured in the `A2B_VMTR_VMIN3` register. Bit 4 is set (=1) when the measured DVDD voltage (`A2B_VMTR_VLTG4.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN4` register. Bit 5 is set (=1) when the measured low-side current (`A2B_VMTR_VLTG5.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN5` register. Bit 6 is set (=1) when the measured high-side current (`A2B_VMTR_VLTG6.VLTG`) is lower than the threshold configured in the `A2B_VMTR_VMIN6` register.

Measured Voltage 0

The A2B_VMTR_VLTG0 register contains the measured voltage for monitor 0 (if enabled).

Address: 0x120

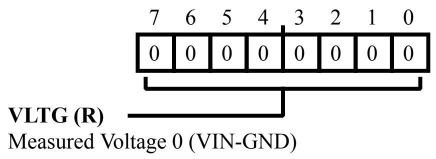


Figure 11-165: A2B_VMTR_VLTG0 Register Diagram

Table 11-166: A2B_VMTR_VLTG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 0 (VIN-GND). The A2B_VMTR_VLTG0 register is updated with the measured voltage in the next superframe when A2B_VMTR_VEN is enabled.

MAX Voltage Threshold

The A2B_VMTR_VMAX0 register is used to configure the maximum allowed voltage for monitor 1.

Address: 0x121

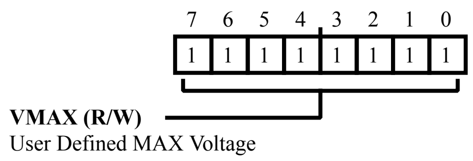


Figure 11-166: A2B_VMTR_VMAX0 Register Diagram

Table 11-167: A2B_VMTR_VMAX0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX0 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 0

The A2B_VMTR_VMIN0 register is used to configure the minimum allowed voltage for monitor 0.

Address: 0x122

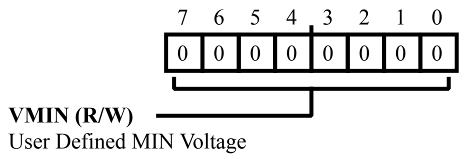


Figure 11-167: A2B_VMTR_VMIN0 Register Diagram

Table 11-168: A2B_VMTR_VMIN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN0 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 1

The A2B_VMTR_VLTG1 register contains the measured voltage for monitor 1 (if enabled).

Address: 0x123

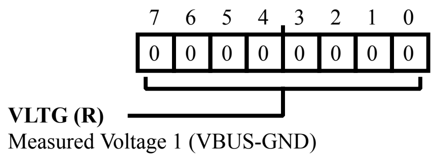


Figure 11-168: A2B_VMTR_VLTG1 Register Diagram

Table 11-169: A2B_VMTR_VLTG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 1 (VBUS-GND). The A2B_VMTR_VLTG1 register is updated with the measured voltage in the next superframe when A2B_VMTR_VEN is enabled.

VMAX Register 1

The A2B_VMTR_VMAX1 register is used to configure the maximum allowed voltage for monitor 1.

Address: 0x124

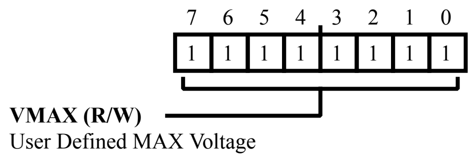


Figure 11-169: A2B_VMTR_VMAX1 Register Diagram

Table 11-170: A2B_VMTR_VMAX1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX1 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 1

The A2B_VMTR_VMIN1 register is used to configure the minimum allowed voltage for monitor 1.

Address: 0x125

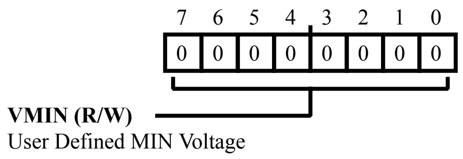


Figure 11-170: A2B_VMTR_VMIN1 Register Diagram

Table 11-171: A2B_VMTR_VMIN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN1 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 2

The A2B_VMTR_VLTG2 register contains the measured voltage for monitor 2 (if enabled).

Address: 0x126

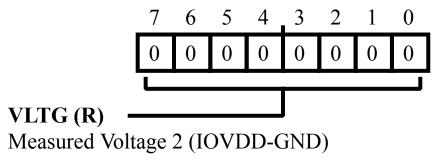


Figure 11-171: A2B_VMTR_VLTG2 Register Diagram

Table 11-172: A2B_VMTR_VLTG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 2 (IOVDD-GND). The A2B_VMTR_VLTG2 register is updated with the measured voltage in the next superframe when A2B_VMTR_VEN is enabled.

VMAX Register 2

The A2B_VMTR_VMAX2 register is used to configure the maximum allowed voltage for monitor 2.

Address: 0x127

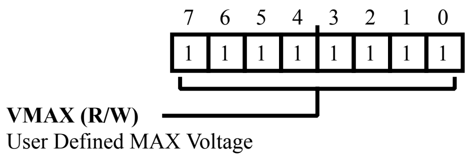


Figure 11-172: A2B_VMTR_VMAX2 Register Diagram

Table 11-173: A2B_VMTR_VMAX2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX2 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 2

The A2B_VMTR_VMIN2 register is used to configure the minimum allowed voltage for monitor 2.

Address: 0x128

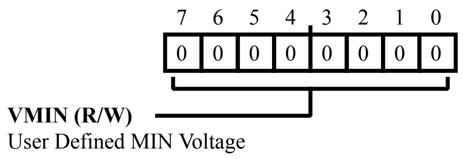


Figure 11-173: A2B_VMTR_VMIN2 Register Diagram

Table 11-174: A2B_VMTR_VMIN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN2 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 3

The A2B_VMTR_VLTG3 register contains the measured voltage for monitor 3 (if enabled).

Address: 0x129

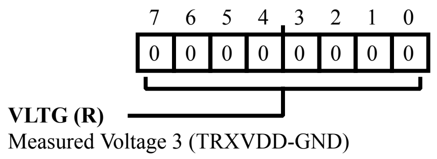


Figure 11-174: A2B_VMTR_VLTG3 Register Diagram

Table 11-175: A2B_VMTR_VLTG3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 3 (TRXVDD-GND). The A2B_VMTR_VLTG3 register is updated with the measured voltage in the next superframe when A2B_VMTR_VEN is enabled.

VMAX Register 3

The A2B_VMTR_VMAX3 register is used to configure the maximum allowed voltage for monitor 3.

Address: 0x12A

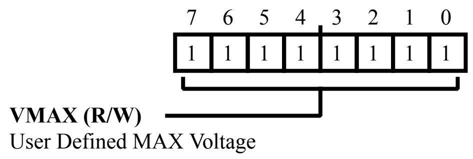


Figure 11-175: A2B_VMTR_VMAX3 Register Diagram

Table 11-176: A2B_VMTR_VMAX3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX3 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 3

The A2B_VMTR_VMIN3 register is used to configure the minimum allowed voltage for monitor 3.

Address: 0x12B

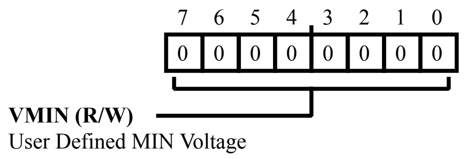


Figure 11-176: A2B_VMTR_VMIN3 Register Diagram

Table 11-177: A2B_VMTR_VMIN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN3 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 4

The A2B_VMTR_VLTG4 register contains the measured voltage for monitor 4 (if enabled).

Address: 0x12C

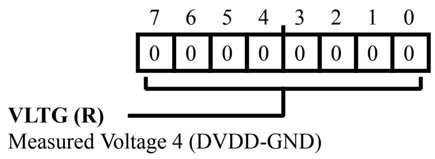


Figure 11-177: A2B_VMTR_VLTG4 Register Diagram

Table 11-178: A2B_VMTR_VLTG4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 4 (DVDD-GND). The A2B_VMTR_VLTG4 register is updated with the measured voltage in the next superframe when A2B_VMTR_VEN is enabled.

VMAX Register 4

The A2B_VMTR_VMAX4 register is used to configure the maximum allowed voltage for monitor 4.

Address: 0x12D

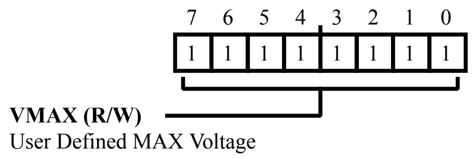


Figure 11-178: A2B_VMTR_VMAX4 Register Diagram

Table 11-179: A2B_VMTR_VMAX4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX4 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 4

The A2B_VMTR_VMIN4 register is used to configure the minimum allowed voltage for monitor 4.

Address: 0x12E

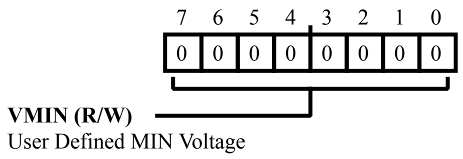


Figure 11-179: A2B_VMTR_VMIN4 Register Diagram

Table 11-180: A2B_VMTR_VMIN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN4 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 5

The A2B_VMTR_VLTG5 register contains the measured voltage for monitor 5 (if enabled).

Address: 0x12F

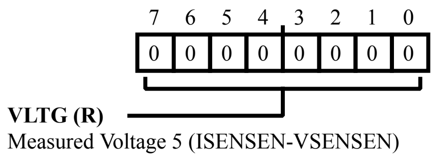


Figure 11-180: A2B_VMTR_VLTG5 Register Diagram

Table 11-181: A2B_VMTR_VLTG5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 5 (ISENSEN-VSENSEN). The low-side current is measured as voltage between ISENSEN and VSENSEN. The A2B_VMTR_VLTG5 register is updated with the measured voltage in the next super-frame when A2B_VMTR_VEN is enabled.

VMAX Register 5

The A2B_VMTR_VMAX5 register is used to configure the maximum allowed voltage for monitor 5.

Address: 0x130

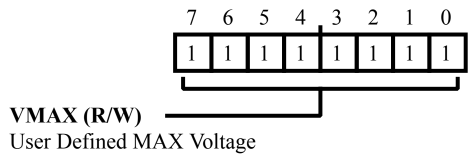


Figure 11-181: A2B_VMTR_VMAX5 Register Diagram

Table 11-182: A2B_VMTR_VMAX5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX5 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 5

The A2B_VMTR_VMIN5 register is used to configure the minimum allowed voltage for monitor 5.

Address: 0x131

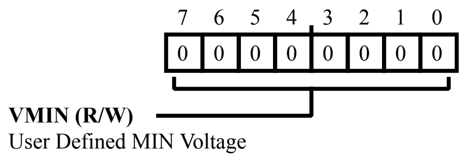


Figure 11-182: A2B_VMTR_VMIN5 Register Diagram

Table 11-183: A2B_VMTR_VMIN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN5 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

Measured Voltage 6

The A2B_VMTR_VLTG6 register contains the measured voltage for monitor 6 (if enabled).

Address: 0x132

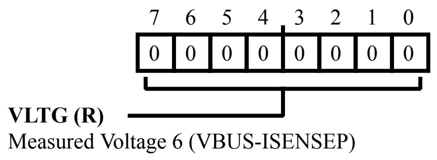


Figure 11-183: A2B_VMTR_VLTG6 Register Diagram

Table 11-184: A2B_VMTR_VLTG6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VLTG	Measured Voltage 6 (VBUS-ISENSEP). The high-side current is measured as voltage between VBUS and ISENSEP. The A2B_VMTR_VLTG6 register is updated with the measured voltage in the next super-frame when A2B_VMTR_VEN is enabled.

VMAX Register 6

The A2B_VMTR_VMAX6 register is used to configure the maximum allowed voltage for monitor 6.

Address: 0x133

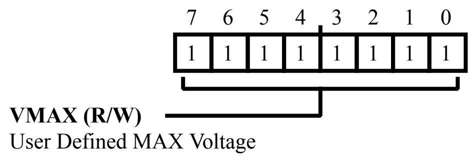


Figure 11-184: A2B_VMTR_VMAX6 Register Diagram

Table 11-185: A2B_VMTR_VMAX6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMAX	User Defined MAX Voltage. The A2B_VMTR_VMAX6 bit field indicates the threshold to generate the A2B_VMTR_MXSTAT interrupt.

VMIN Register 6

The A2B_VMTR_VMIN6 register is used to configure the minimum allowed voltage for monitor 6.

Address: 0x134

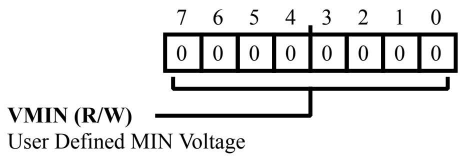


Figure 11-185: A2B_VMTR_VMIN6 Register Diagram

Table 11-186: A2B_VMTR_VMIN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VMIN	User Defined MIN Voltage. The A2B_VMTR_VMIN6 bit field indicates the threshold to generate the A2B_VMTR_MNSTAT interrupt.

PWM Configuration Register

The A2B_PWMCFG register is used to configure PWM functionality.

Address: 0x140

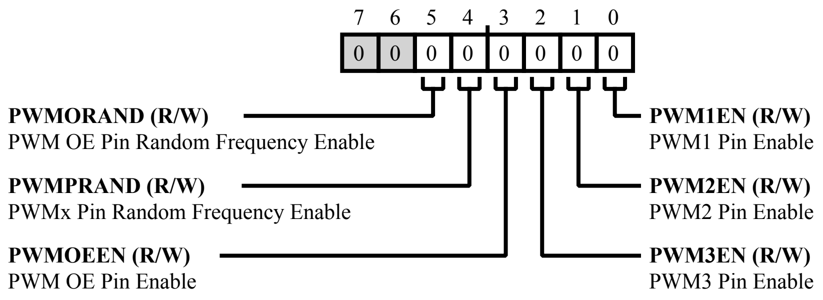


Figure 11-186: A2B_PWMCFG Register Diagram

Table 11-187: A2B_PWMCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	PWMORAND	PWM OE Pin Random Frequency Enable.
4 (R/W)	PWMPRAND	PWMx Pin Random Frequency Enable.
3 (R/W)	PWMOEEN	PWM OE Pin Enable.
2 (R/W)	PWM3EN	PWM3 Pin Enable.
1 (R/W)	PWM2EN	PWM2 Pin Enable.
0 (R/W)	PWM1EN	PWM1 Pin Enable.

PWM Frequency Register

The A2B_PWMFREQ register is used to set PWM frequencies for the PWM1/PWM2/PWM3 pins and for the PWMOE pin. These values are not used if random frequencies are enabled in A2B_PWMCFG.

Address: 0x141

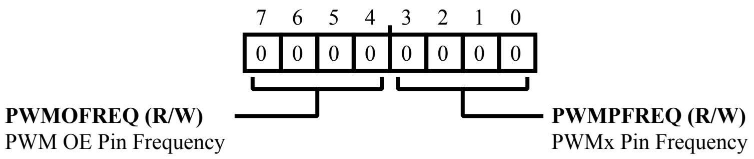


Figure 11-187: A2B_PWMFREQ Register Diagram

Table 11-188: A2B_PWMFREQ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/W)	PWMOFREQ	PWM OE Pin Frequency.	
		0	192 kHz
		1	96 kHz
		2	48 kHz
		3	24 kHz
		4	12 kHz
		5	6 kHz
		6	3 kHz
		7	1500 Hz
		8	750 Hz
		9	375 Hz
		10	187.5 Hz
3:0 (R/W)	PWMPFREQ	PWMx Pin Frequency.	
		0	192 kHz
		1	96 kHz
		2	48 kHz
		3	24 kHz
		4	12 kHz
		5	6 kHz
		6	3 kHz
		7	1500 Hz

Table 11-188: A2B_PWMFREQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8	750 Hz
		9	375 Hz
		10	187.5 Hz

PWM Blink Register 1

The A2B_PWMBLINK1 register is used to control blink behavior for PWM1 and PWM2.

Address: 0x142

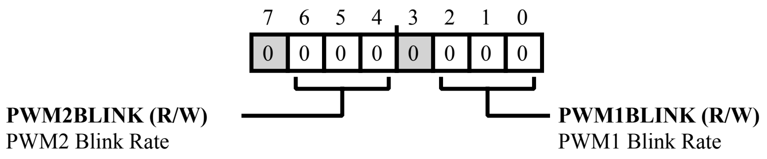


Figure 11-188: A2B_PWMBLINK1 Register Diagram

Table 11-189: A2B_PWMBLINK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PWM2BLINK	PWM2 Blink Rate.	
		0	No Blink
		1	1/4 Second Blink
		2	1/2 Second Blink
		3	3/4 Second Blink
		4	1 Second Blink
2:0 (R/W)	PWM1BLINK	PWM1 Blink Rate.	
		0	No Blink
		1	1/4 Second Blink
		2	1/2 Second Blink
		3	3/4 Second Blink
		4	1 Second Blink

PWM Blink Register 2

The A2B_PWMBLINK2 register is used to control blink behavior for PWM3 and PWMOE.

Address: 0x143

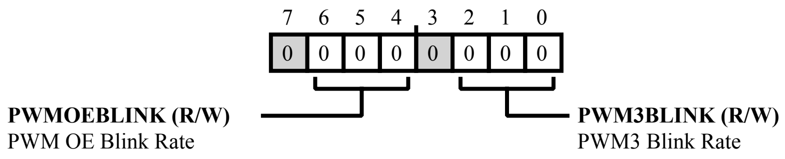


Figure 11-189: A2B_PWMBLINK2 Register Diagram

Table 11-190: A2B_PWMBLINK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PWMOEBLINK	PWM OE Blink Rate.	
		0	No Blink
		1	1/4 Second Blink
		2	1/2 Second Blink
		3	3/4 Second Blink
		4	1 Second Blink
2:0 (R/W)	PWM3BLINK	PWM3 Blink Rate.	
		0	No Blink
		1	1/4 Second Blink
		2	1/2 Second Blink
		3	3/4 Second Blink
		4	1 Second Blink

PWM1 Value Low Bits Register

The A2B_PWM1VALL register contains the LSBs for the PWM value for PWM1.

Address: 0x148

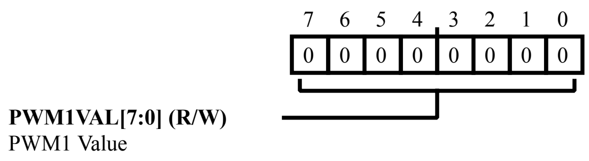


Figure 11-190: A2B_PWM1VALL Register Diagram

Table 11-191: A2B_PWM1VALL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM1VAL	PWM1 Value.

PWM1 Value High Bits Register

The A2B_PWM1VALH register contains the MSBs for the PWM value for PWM1.

Address: 0x149

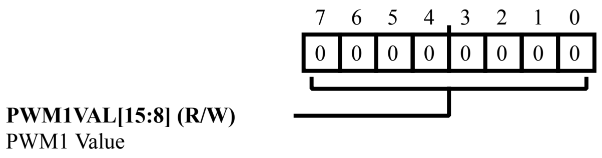


Figure 11-191: A2B_PWM1VALH Register Diagram

Table 11-192: A2B_PWM1VALH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM1VAL	PWM1 Value.

PWM2 Value Low Bits Register

The A2B_PWM2VALL register contains the LSBs for the PWM value for PWM2.

Address: 0x14A

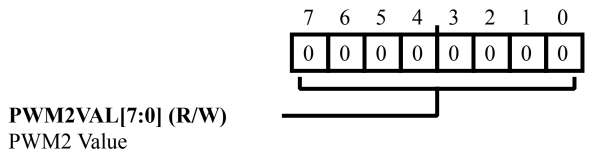


Figure 11-192: A2B_PWM2VALL Register Diagram

Table 11-193: A2B_PWM2VALL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM2VAL	PWM2 Value.

PWM2 Value High Bits Register

The A2B_PWM2VALH register contains the MSBs for the PWM value for PWM2.

Address: 0x14B

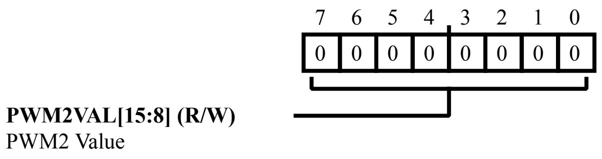


Figure 11-193: A2B_PWM2VALH Register Diagram

Table 11-194: A2B_PWM2VALH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM2VAL	PWM2 Value.

PWM3 Value Low Bits Register

The A2B_PWM3VALL register contains the LSBs for the PWM value for PWM3.

Address: 0x14C

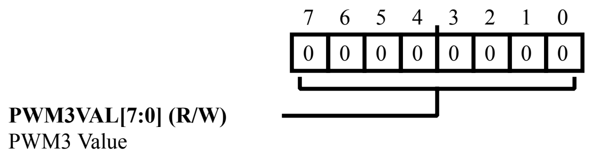


Figure 11-194: A2B_PWM3VALL Register Diagram

Table 11-195: A2B_PWM3VALL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM3VAL	PWM3 Value.

PWM3 Value High Bits Register

The A2B_PWM3VALH register contains the MSBs for the PWM value for PWM3.

Address: 0x14D

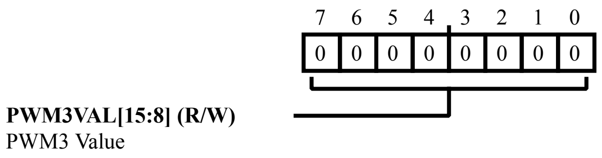


Figure 11-195: A2B_PWM3VALH Register Diagram

Table 11-196: A2B_PWM3VALH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWM3VAL	PWM3 Value.

PWM OE Value Low Bits Register

The A2B_PWMOEVALL register contains the LSBs for the PWM value for PWMOE.

Address: 0x14E

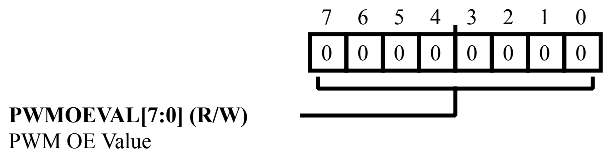


Figure 11-196: A2B_PWMOEVALL Register Diagram

Table 11-197: A2B_PWMOEVALL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWMOEVAL	PWM OE Value.

PWM OE Value High Bits Register

The A2B_PWMOEVALH register contains the MSBs for the PWM value for PWMOE.

Address: 0x14F

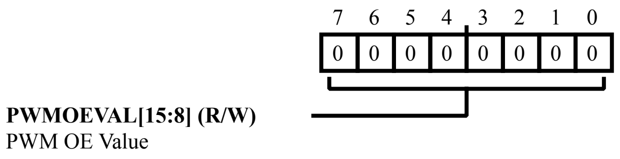


Figure 11-197: A2B_PWMOEVALH Register Diagram

Table 11-198: A2B_PWMOEVALH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PWMOEVAL	PWM OE Value.

MMR Page Register

The A2B_MMRPAGE1 register provides a page address for MMR register accesses. This register is written to 0x01 to access the VMTR and PWM registers. All other registers are in MMR page 0. This register should be 0 while accessing the page 0 registers. Regardless of the value of this register, a write to address 0xE0 always updates its value. MMR page values other than 0 and 1 are not supported and must not be written.

Address: 0x1E0

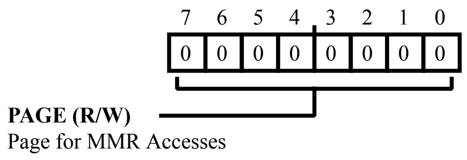


Figure 11-198: A2B_MMRPAGE1 Register Diagram

Table 11-199: A2B_MMRPAGE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	PAGE	Page for MMR Accesses.