

# An Introduction to the Mobile Industry Processor Interface (MIPI) Alliance Standard

## Serial Low-power Inter-chip Media Bus (SLIMbus™)

**By**  
**Kenneth Boyce**  
**Technology Director**  
**National Semiconductor Audio Products Group**

### **NOTICE**

SLIMbus™ is a trademark of the MIPI Alliance, Inc.

This article is a general overview and introduction to SLIMbus™ and does not provide a thorough explanation of SLIMbus™. This article is not a substitute for any part of the SLIMbus™ Specification. For a complete description of operational capabilities for SLIMbus™, please use the SLIMbus™ Specification which is available to members of the MIPI Alliance, Inc.

MIPI Alliance, Inc.  
c/o IEEE-ISTO  
445 Hoes Lane  
Piscataway, NJ 08854  
<http://www.mipi.org>

### **About The MIPI Alliance**

The Mobile Industry Processor Interface (MIPI) Alliance is a collaboration of mobile industry leaders with the objective to define and promote open standards for interfaces to mobile application processors. Through these open standards, the MIPI Alliance intends to speed deployment of new services to mobile users by establishing specifications for standard hardware and software interfaces to mobile application processors and encouraging the adoption of those standards throughout the industry value chain. The MIPI Alliance is intended to complement existing standards bodies with a focus on microprocessors, peripherals and software interfaces.

## Introduction

The Serial Low-power Inter-chip Media Bus (SLIMbus™) is a standard interface between baseband or application processors and peripheral components in mobile terminals. The SLIMbus™ specification was developed within the MIPI Alliance by a dedicated group of MIPI member companies: Austria Microsystems, Freescale Semiconductor, Infineon Technologies AG, Intel Corporation, Marvell Semiconductor, Motorola, National Semiconductor, Nokia, NXP Semiconductor, RF Microdevices, Sonion, Sony-Ericsson, STMicroelectronics N.V., Synopsys, Texas Instruments, Toshiba, and Wolfson Microelectronics.

Excerpt from the MIPI ALLIANCE website ([www.mipi.org](http://www.mipi.org)):

*“The Mobile Industry Processor Interface (MIPI) Alliance is an open membership organization that includes leading companies in the mobile industry that share the objective of defining and promoting open specifications for interfaces in mobile terminals. MIPI Specifications establish standards for hardware and software interfaces between the processors and peripherals typically found in mobile terminal systems. By defining such standards and encouraging their adoption throughout the industry value chain, the MIPI Alliance intends to reduce fragmentation and improve interoperability among system components, benefiting the entire mobile industry.”*

SLIMbus™ development was driven by the increased demand for multimedia functions within mobile terminals and other portable entertainment devices, as well as the recognition that high quality digital audio is a major driver of unit growth and product differentiation.

## SLIMbus™ Features Highlights

A partial, but representative, listing of the features of SLIMbus™ is given below.

- Audio, data, bus and Device control on a single bus
- Reduced pin count for lower overall product cost
- Support for multiple, high quality audio channels
- Multiple, concurrent sample rates on a single bus
- Efficient, host-less, peer-to-peer generic data communication
- Standardized Message set for improved software reuse and increased interoperability
- Use of common digital audio clocks as well as established system clocks
- Dynamic clock frequency changes for optimizing bus power consumption

SLIMbus™ addresses limitations of existing digital audio interfaces such as I<sup>2</sup>S and PCM (which are primarily point-to-point connections between single components and support only one or two digital audio channels) by providing a scalable multi-drop architecture supporting many components and digital audio channels on a single bus structure.

For even greater flexibility and simplicity, SLIMbus™ eliminates the need for a control bus such as I<sup>2</sup>C, SPI, microWire™, UART or GPIO pins on the digital audio components. Additionally, it may also reduce (or eliminate) instances of these bus structures on other types of low bandwidth components within the mobile terminal.

SLIMbus™ is implemented by a synchronous 2-wire, flexible TDM frame structure and surrounding bus arbitration mechanisms and message structures, which taken together establish flexible and robust data connections between SLIMbus™ Devices.

Although optimized for the transport of constant-rate media streams, SLIMbus™ can transport various types of asynchronous data as well as control data.

## **SLIMbus™ Physical Description**

Physically, SLIMbus™ consists of two terminals, the data line (DATA) and the clock line (CLK) which are used to interconnect multiple SLIMbus™ Devices.

SLIMbus™ uses a multidrop bus topology where all bus signals are common to all components on the bus. As such, all devices on the bus must use the same protocol for communication. This architecture was chosen as it significantly reduces bus interconnect wiring in any product using it, while at the same time allowing a multiplicity of devices and device types to be connected to the bus.

A multidrop connection requires that only one transmitter device communicate at any given time on the bus to one or more receivers. SLIMbus™ devices contend for the bus access through an arbitration procedure.

SLIMbus™ uses a Time Division Multiplexed (TDM) architecture which allows multiple receiver and transmitter devices to reside on the bus and allows all devices to inter-communicate within allocated channels and time frames. SLIMbus™ supports both device to device communication as well as single device to multiple device broadcast communication.

SLIMbus™ is not designed for hot-swap capability since the intended use is completely within a client terminal such as a cellular phone. However, SLIMbus™ devices may dynamically “drop off” the bus and “reconnect” to the bus as dictated by system usage requirements using appropriate protocols outlined in the SLIMbus™ specification.

## **SLIMbus™ Devices and Device Classes**

A SLIMbus™ Device is a logical implementation of a system function.

Devices in a Device Class category share certain characteristics and functions. SLIMbus™ Devices fall into Device Class definitions which specifies the minimum requirements for Device control information, Device behavior, data Transport Protocol support, and data storage necessary to implement a Device of that Device Class.

Requirements for all Device Classes are:

- The Device Class code, identifying the type of Device
- The version number of the Device Class
- Transport support requirements, e.g. the number of Ports and required attributes, directionality, and Transport Protocols, etc., that are supported by these Ports
- Message support requirements, identifying which Messages in addition to Core Messages are supported by the Device.
- Information support, identifying the Core Information Elements and associated codes that are supported by the Device.

- Operation requirements, identifying all other behavior that is important to the operation of a Device that belongs to that Device Class.

There are four SLIMbus™ Device Classes defined at the release of Version 1.0 of the SLIMbus™ specification: Manager, Framer, Interface, and Generic. These Device Classes permit complete SLIMbus™ systems to be designed and implemented without any additional Device Classes. However, the set of Device Classes is expandable if desired. When additional Device Classes are defined, those Device Class codes will be allocated by the MIPI Alliance.

## **Manager Device**

A Manager Device is responsible for booting SLIMbus™, and performs bus administration (enumeration of Components and Devices, bus configuration, and dynamic channel allocation).

In a typical SLIMbus™ system the Manager would be located in a baseband or application processor rather than in a peripheral component.

If a system contains two Managers, only one is allowed to be active at any given time.

## **Framer Device**

The Framer delivers a clock signal on the CLK line to all SLIMbus™ Components, and also contains logic to transmit the Guide and Framing Channels (Framing Information) on the DATA line in order to establish the highest level TDM Frame Structure of the bus and communicate that information to other SLIMbus™ Devices for establishing synchronization.

The clock may be a high quality clock useful for audio decoding and DACs which may eliminate the need for additional clock generation within the system.

## **Interface Device**

In each Component the Interface Device provides bus management services, controls the Frame Layer, monitors Message Protocols implemented by the Component, reports information about the status of the Component, and manages the Component reset so that a Component can properly sequence its Devices.

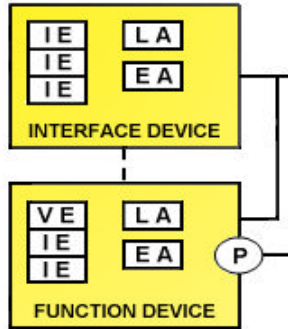
## **Generic Device (Function)**

A Generic Device is a Device other than a Manager, Framer, or Interface. A Generic Device is generally considered to be the device to provide certain application functionality such as, for example, conversion from digital audio to analog (DAC) and vice versa (ADC). For this reason, Generic Devices are labeled "Function Device" in block diagrams in this article.

The Generic Device Class is used if no other specific device class for the application functionality exists. As an example, if a Microphone Device Class existed you would not ordinarily use a Generic Device class definition for microphone functionality.

To implement a Functional SLIMbus™ Device also requires the use of a SLIMbus™ Interface Device, and the associated Enumeration and Logical Addresses (EA and LA), Information and Value Elements (IE and VE), and Ports (P) of each device, which are used to establish bus connections, control and status information flow, and digital audio (or other data) flow.

Figure 1 below is an example showing a partial section of an Interface Device and a partial section of a Function Device and associated elements. Further detail is shown in Figure 2 below.



**Figure 1: Partial Sections of SLIMbus™ Interface Device and Function Device**

## Device Information and Value Elements

Information Elements (IE) and Value Elements (VE) are data storage elements used to hold status, configuration or other important information needed by a Device. The data stored may be Boolean in nature, or may have many values, depending upon the type of Device.

These IE and VE elements effectively replace registers typically found behind conventional control interfaces such as I<sup>2</sup>C or SPI.

An Information Element is a specific piece of data that resides in a Device, and is available to other Devices via Messages. Categories of Information Elements are:

- Core - Information Elements which are the same for all Devices of all Device Classes
- Device Class-specific - Information Elements which are the same for all Devices of a particular Device Class, but which may be different for all Devices of a different Device Class
- User - Information Elements which are specific to a particular product or product family

A Value Element provides a standardized method to read and update Device parameters. Unlike an Information Element, a Value Element can be set to a specific value using a specific Message for the purpose.

## Device Addressing

SLIMbus™ uses a 48-bit Enumeration Addresses (EA) to uniquely identify Devices which can announce their presence on the bus. Each Device has an EA, which incorporates Manufacturer ID, Product Code, Device Index, and Instance Value for a Device. The Manufacturer ID code is supplied by the MIPI Alliance and uniquely identifies the manufacturer of the Device, similar to the manufacturer IDs used with PCI bus components. The Device Index code uniquely identifies multiple Devices within a single Component. The Instance Value code is for the case where multiple Devices of the same type or Class are attached to the bus.

After enumeration, the Active Manager assigns an 8-bit Logical Address to Device (LA) which speeds up communication with devices, and is typically used until the bus is powered down. On the next bus power up, or if the Device drops off the bus and re-connects, it is entirely possible that a new Logical Address is assigned.

## Ports

A Port provides the connection path to data flow between Devices. A particular Device may have up to sixty-four Ports.

Port capabilities vary depending upon the Device and are to be specified in the Component data sheet. Typical Port attributes include data directionality, i.e. input-only (sink), output-only (source), or both input and output; supported Transport Protocols; and data width. For example, the Port attributes for a MEMS microphone could be output-only, Isochronous Transport Protocol and 16-bit data width.

A Port status moves through states prior to being used for data transfer.

The power on, or reset, condition of a Port is in the *Disconnected* state, and the Port does not produce or consume any data.

When the Port is connected to a Data Channel, it moves to the *Unconfigured* state and also does not produce or consume any data.

Once in the *Unconfigured* state, the Port receives channel configuration Messages and acts on them accordingly.

After receiving all necessary configuration parameters, the Port state changes to the *Configured* state, and the Port is now ready for data transport.

## SLIMbus™ Component

A SLIMbus™ Component contains two or more SLIMbus™ Devices. A SLIMbus™ Component must have one (and only one) SLIMbus™ Interface Device, and may have one or more other types of SLIMbus™ devices.

A simple SLIMbus™ Component example is shown in Figure 2 below, and a complex SLIMbus™ Component example is shown in Figure 3 below.

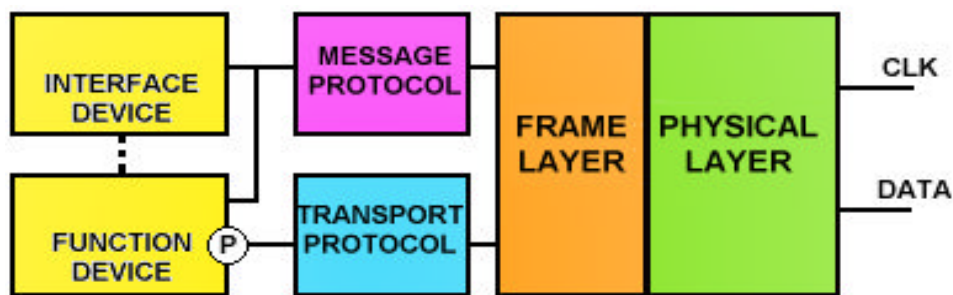
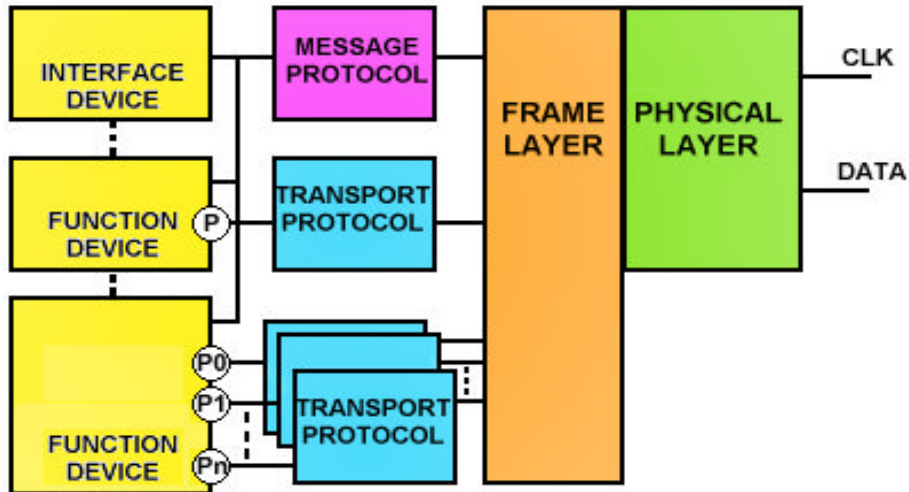


Figure 2: Simple SLIMbus™ Component



**Figure 3: Complex SLIMbus™ Component**

As shown in the Figure 3, Data and Control information sent from a Device are first encoded using the Message Protocol for control information and the Transport Protocol for data. The data and control streams are interleaved by the Frame Layer and converted by the Physical Layer into electrical signals on the DATA and CLK lines.

In the opposite direction, the signals on the CLK and DATA lines are translated by the Physical Layer into a bit stream, which is then split into Data and Control streams by the Frame Layer, which, in turn, are decoded by the corresponding protocols and sent to the appropriate Devices within the Component.

### **SLIMbus™ DATA and CLK**

The DATA and CLK lines are typically attached to two or more SLIMbus™ Devices contained in one or more SLIMbus™ Components to form a SLIMbus™ system.

All SLIMbus™ Devices use DATA and CLK to synchronize with the bus configuration in use, to receive or transmit messages and data, and to implement bus arbitration, collision detection, and contention resolution between devices.

For all SLIMbus™ Components (except one containing a Framer Device), the CLK terminal is input only.

If a SLIMbus™ Component is the Framer Device, or contains a Framer Device, the CLK signal is bi-directional. This is because the Framer Device is the only SLIMbus™ device allowed to control the CLK line. The CLK line has only two states; driven high or driven low. The Component that contains the active Framer is called the Clock Source Component.

For all SLIMbus™ Components, the DATA line is bidirectional, and carries all information sent or received on the bus using Non-Return-to-Zero Inverted, or NRZI encoding. The DATA line is driven on the positive edge and read on the negative edge of the CLK line. The DATA line can be driven high, driven low, or held at the high or low level, depending upon the particular operational mode of a SLIMbus™ Device.

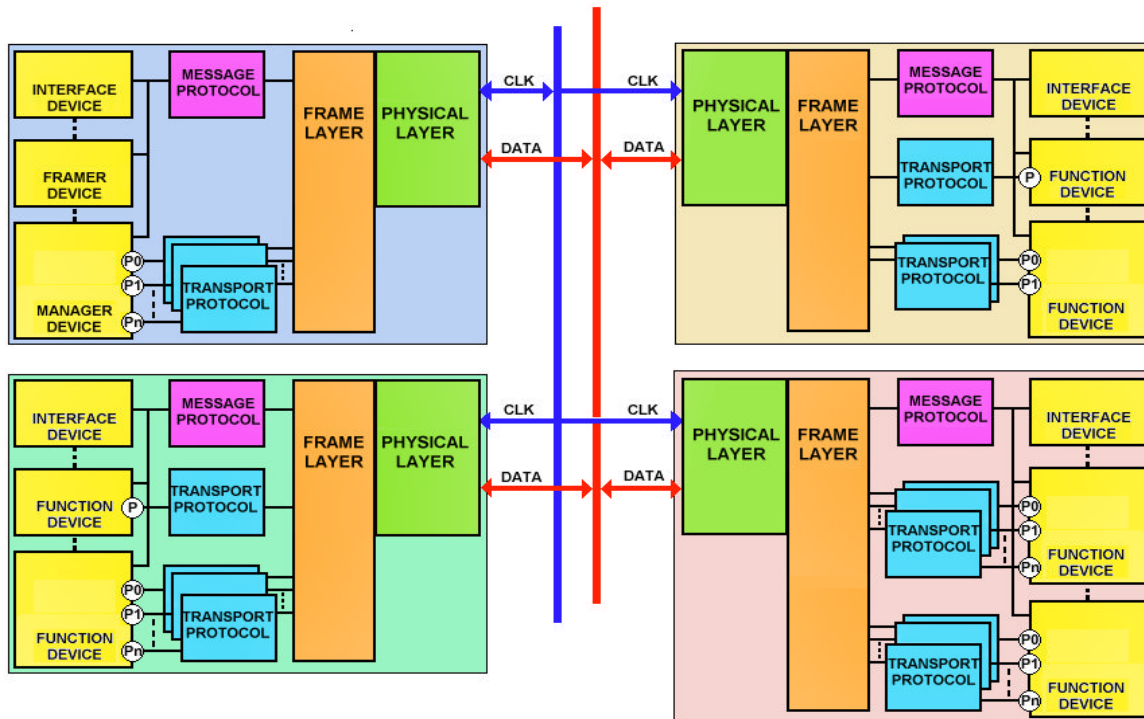
The SLIMbus™ interface DATA and CLK lines use CMOS-like single-ended, ground referenced, rail-to-rail, voltage mode signals, and signaling voltages are specified with respect to the interface supply voltage.

In the SLIMbus™ specification, interface supply voltages +1.8V<sub>dd</sub> and +1.2V<sub>dd</sub> are recommended even though other parts of a SLIMbus™ device may use different supply voltages.

## SLIMbus™ System

A possible SLIMbus™ system for illustrative purposes only is shown in Figure 4 below. All of the Components are different from each other. Note that the upper left SLIMbus™ Component in this example contains a Framer Device, and therefore the CLK signal for this component is bidirectional.

The upper left SLIMbus™ Component also contains a Manager Device. However, there is no requirement that the Manager and the Framer Devices be in the same SLIMbus™ Component.



**Figure 4: An Illustrative SLIMbus™ System**

All of the elements shown in the upper left SLIMbus™ Component can also be incorporated into baseband and/or applications processors (Figure 5) typically used to build mobile terminals.

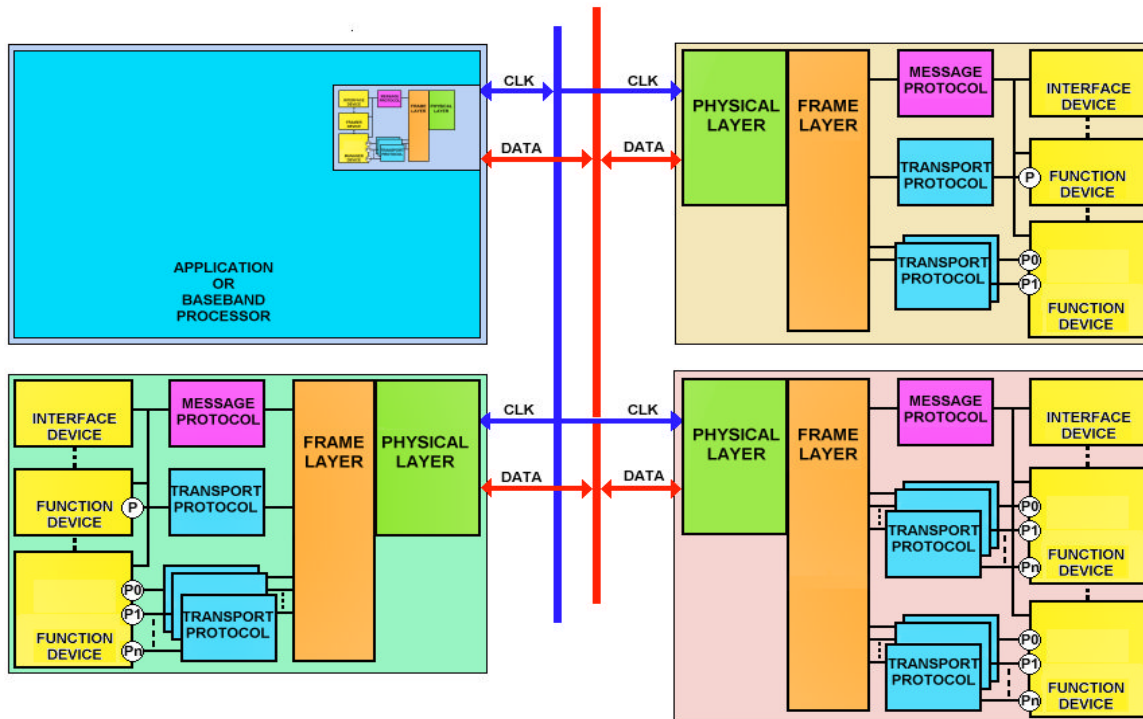


Figure 5: SLIMbus™ System with Manager and Framer in Processor

## SLIMbus™ Model and Operational Description

A SLIMbus™ system model consists of a set of SLIMbus™ Devices in Components communicating with each other using a shared data (DATA) line, and common clock (CLK) signal.

Information on the SLIMbus™ DATA line is allocated to Control Space and Data Space channels.

The Control Space carries bus configuration and synchronization information as well as inter-Device Message communication. The Control Space may be dynamically programmed to take as much of the SLIMbus™ bandwidth as required, even up to 100% at times.

Data Space, when present, carries application-specific information such as isochronous, near-isochronous and asynchronous data streams.

SLIMbus™ Component Devices convey control and data information between each other using Control and Data Channels with Transport Protocols to achieve the required system operation. Messages are used for Control functions. Transport Protocols handle both Control data and Application data flow types.

### Channels

Channels can be established between a pair of Devices (inter-Device communication), or between one Device and many Devices (broadcast communication).

### Control Channels

Control Space or Control Channels are actually three different types of channels: Framing, Guide, and Message, with each one having a specific purpose.

The Framing Channel carries Frame Sync symbol and Framing Information in two Slots of each Frame which conveys bus configuration parameters so that all Components can properly synchronize to the bus configuration being used at the time. Flow control is not available. The channel width is fixed.

The Guide Channel is carried in one Slot for the first and second Frame of a Superframe, and provides the necessary information for a Component to acquire and verify Message synchronization in the Message Channel. Flow control is not available. The channel width is fixed.

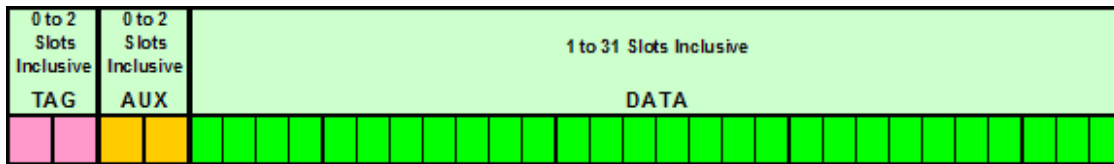
The Message Channel carries various types of information including bus configuration, Device control and Device status. Flow control is implemented by an acknowledgment symbol. Channel width is programmable.

## Data Channels

Any SLIMbus™ bandwidth not allocated to Control Space is available for Data Space (Data Channels). Data Space is composed of one or more Data Channels that are dynamically created by the active Manager depending on the application. The number of Data Channels depends on the size of Data Space and type of data streams carried by the channels. Data Space can contain up to 256 Data Channels.

A Data Channel is a stream of one or more contiguous Data Slots that repeats at a fixed interval. The group of contiguous Slots within a Data Channel is known as a Segment. Because Segments repeat at known intervals relative to Superframe boundaries, Data Channels can be viewed as a virtual bus with their own bandwidth guarantee and latency.

A Segment is composed of three fields called the TAG, AUX, and DATA fields. Note the TAG and AUX fields are optional and may be omitted. These fields are organized within the Segment as shown in Figure 6.



**Figure 6 Segment Organization**

The TAG bits carry flow control information if any is needed for the data channel.

The auxiliary (AUX) bits carry side information linked to the content of the DATA field.

The exact composition of the Segment depends on the Transport Protocol being used for the Data Channel.

The active Manager initializes a Data Channel and communicates the related content parameters to all Devices using that Data Channel.

A Data Channel is additionally defined by the parameters; Channel Definition and Content Definition, where each Definition contains a number of other parameters such as, but not limited to, Data rate, type, and field length, active/non-active status, and Transport Protocol to use. Thus, information needed to use the Channel is completely described and is made available to all Devices involved in using a particular Data Channel.

Figure 7 below shows a conceptual view of a SLIMbus™ system.

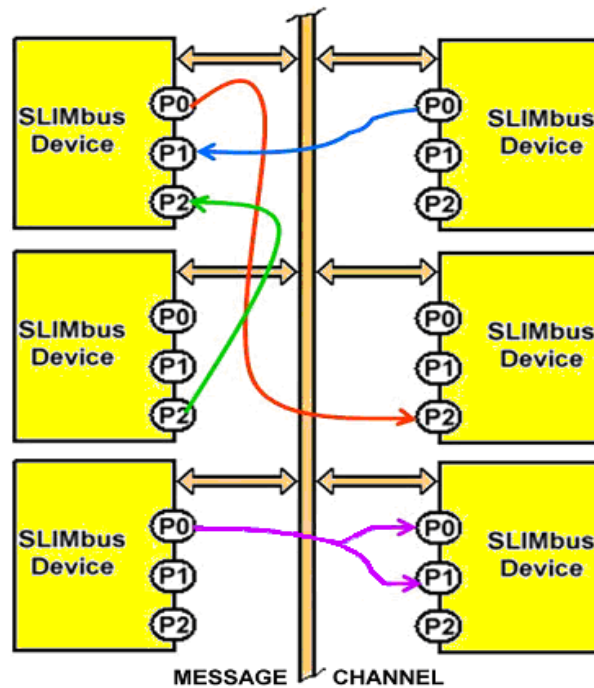


Figure 7: SLIMbus™ Model Representation

### Data Channel Transport Protocols and Flow Control

Information carried by a Data Channel is application specific and many different data formats can co-exist.

SLIMbus™ does not directly support various data formats. Instead, to transport any data format, a small group of frequently used Transport Protocols (including a User Defined Transport Protocol) are specified which define data flow type, flow control mechanism, and a side-channel (if any) for any additional application-specific information.

Data flow between Ports is by one of several Transport Protocols.

SLIMbus™ Device Ports are associated with channels using appropriate channel connection and disconnection Messages.

Transport Protocols are either Uni-cast or Multi-cast types. The types of Transport Protocols defined in SLIMbus™ are summarized in Table 1 below.

Isynchronous (Multi-cast)	Pushed (Multi-cast)	Pulled
Asynchronous - Simplex	Asynchronous - Half duplex	Extended Asynchronous - Simplex
Extended Asynchronous - Half duplex	Locked (Multi-cast)	User Defined 1 & 2

## TABLE 1: SLIMbus™ Supported Transport Protocols

A Data Channel has exactly one data source at a time and may have one or more data sinks depending upon the Transport Protocol used in the channel.

Flow Control in the Channel, if needed, depends on the Devices and the type of Data involved. TAG bits are used to carry the flow control information.

No flow control is needed if the frequency of the CLK line is an integer multiple of the data flow rate. Therefore, the Isochronous Transport Protocol can be used.

If flow control is needed, one of the two supported styles of flow control needs to be selected: Single-ended, or Double-ended flow control.

Single-ended data flows are regulated by a shared algorithm (for the Locked Protocol), or by the use of a 'Presence' bit (for the Pulled and Pushed Protocol respectively). These protocols are designed to optimally carry constant rate media streams (such as LPCM audio) where the actual control method used for a flow depends on the bus Root Frequency, as well as the characteristics of the flow.

When the Pushed protocol is used to carry data whose rate is equal to, or lower than, the channel rate, the source Device drives the data flow and the TAG bits indicate availability of data in the DATA field. A Pushed Protocol Data Channel can be connected to multiple sinks (multicast) since there is no feedback from the sinks.

When the Pulled Transport Protocol is used, the sink Device requests, or pulls, data from the source Device when needed and the TAG bits indicate availability of data in the DATA field.

With double-ended handshaking, either Device involved in the data transfer can stop or start the transfer using two or more flow control bits in every Data Segment's TAG field. The four Asynchronous Transport Protocols all use this type of flow control. These Transport Protocols have been designed to optimally support asynchronous data flow.

The Extended Asynchronous Protocols have 2 TAG fields plus a DATA field size that can range from 2 to 28 Slots inclusive by increments of 2 Slots, and is used when the channel rate is low and the data rate is high.

User 1 & 2 protocols are used to extend SLIMbus™'s data transmission mechanisms and it is assumed that a Device connected to a User Protocol Data Channel knows the definition of the TAG and AUX bits and how they are used. For example, a User Protocol could be created to support SPDIF (IEC 60958) digital audio data flows on SLIMbus™.

## SLIMbus™ Frame Structure

SLIMbus™ uses a synchronous, two-wire bus to carry information between Devices. The SLIMbus™ bit stream is organized in a Time Domain Multiplexed (TDM) configuration. The organization of information on the bus is called the Frame Structure.

As stated before, SLIMbus™ Control Space and Data Space information is transported in channels, with each channel representing a particular information flow. The bandwidth used by the Control Space and Data Space is configurable so that the bus can be adapted to virtually any application.

## **Cells, Slots, Subframes, Frames, and Superframes**

The Frame Structure has five building blocks: Cells, Slots, Frames, Subframes, and Superframes.

### **Cell**

The smallest subdivision of a SLIMbus™ data flow is the Cell. A Cell is defined as a region of the DATA signal that is bounded by two consecutive positive edges of the CLK line. Each Cell can hold a single bit of information.

### **Slot**

A Slot is defined as four contiguous Cells (4-bits) and is the unit of bandwidth allocation on SLIMbus™. Cells within a Slot are labeled C0 through C3, and are transmitted in MSB to LSB order. A variety of data bit sizes, from 4-bits to 32-bits or more, can be easily accommodated by groups of 4-bit Slots.

### **Frame**

A Frame is defined as 192 contiguous Slots. Slots within a Frame are labeled S0 through S191, and are transmitted as S0, followed by S1, S2 ... S191 in that order.

The first Slot (Slot 0) of each Frame is a Control Space slot which contains the 4-bit Frame Sync symbol. Slot S96 of each Frame is also a Control Space slot which contains 4-bits of Framing Information.

A Component uses the Frame Sync data and 32 bits of Framing Information to synchronize to the bus. Therefore, to receive all 32 bits of the Framing Information, data must be read from eight successive Frames, called a Superframe. See description below.

Included in the Framing Information is an encoded Whitening Signal which is used to reduce the autocorrelation properties and electromagnetic spectrum of the data.

Additionally, the Framing Information also contains a 25 Hz timing reference signal, which is the highest common factor of many common audio sample rates.

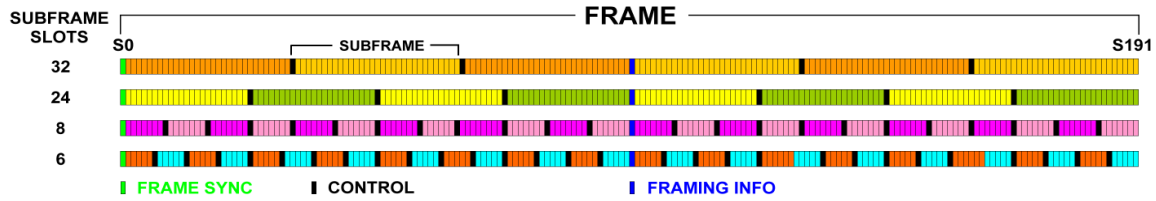
Further discussion of the creation and use of Whitening Signal or the 25 Hz timing reference signal is beyond the scope of this article.

The active Framer writes all Framing Information to the Data line at the appropriate time.

### **Subframe**

A Subframe is defined as the division of the Frame Structure at which Control Space and Data Space are interleaved. The first Slot of a Subframe is always allocated to Control Space.

Subframes do not have a single, fixed length. As shown in Figure 8 below, the Subframe length is programmable to 6, 8, 24 or 32 contiguous Slots (24, 32, 96 or 128 Cells). The number of possible Subframes per Frame is therefore, 32, 24, 8, or 6 respectively. The Subframe configuration used can be dynamically changed, and depends upon the data flow requirements of the applications being supported at the time.

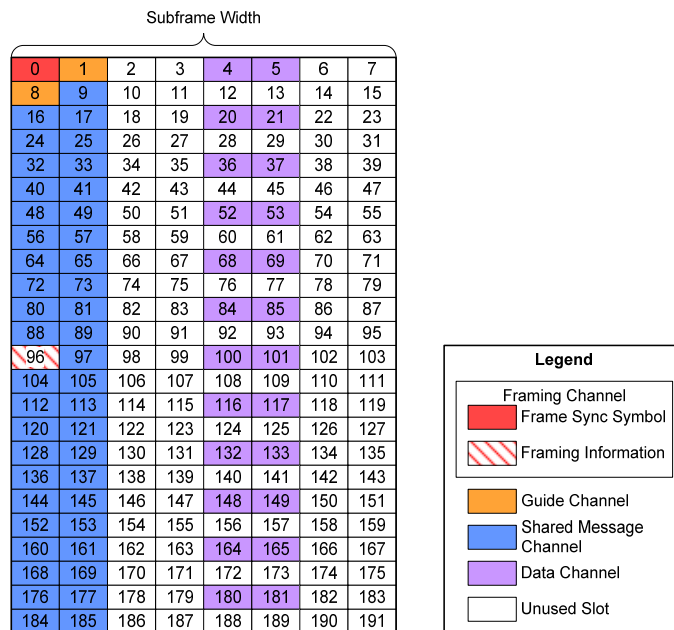


**Figure 8: Cell, Slot, Subframe, Frame Structure**

A discussion of the reasons behind the particular SLIMbus™ Subframe organization is beyond the scope of this article.

Recall that Control Space is composed of three different types of channels: Framing (Frame Sync and Framing Information), Guide, and Message. However, Frame Sync and Framing Information only occupy two slots per Frame, or the first slot in two different Subframes in a Frame. This means that the first slots in the remaining Subframes of a Frame are used to transmit the other Control Space information, e.g. Guide Channel and Message Channel. See Figure 8 above.

Any Slots not allocated to Control Space are considered Data Space. One possible example within a single Frame is shown in Figure 9 below.



**Figure 9: One Possible Bus Configuration Arrangement of Control Space and Data Space**

(Example from SLIMbus™ Specification, Version 1.0. Used with permission)

## Superframe

A Superframe is defined as eight contiguous Frames (1536 Slots). Frames within a Superframe are labeled Frame 0 through Frame 7.

Each Frame of a Superframe contains the Frame Sync symbol in Slot 0. The first Frame (Frame 0) of a Superframe contains the first 4-bits of a total of 32-bits of Framing Information in Slot 96. Frames 1 through Frame 7 successively also contain four (4) bits of Framing Information in Slot 96 with Frame 7 carrying the last 4-bits of Framing Information.

The beginning of a Superframe is defined by a Superframe Sync pattern transmitted 1-bit at a time in five successive Frames.

A Component uses a complete set of Framing Information (32-bits over 8 Frames, 4-bits per Frame) and Superframe Sync to achieve Superframe synchronization.

The Guide Channel (used for Message Synchronization) consists of two slots, one carried in the first Frame of a Superframe, and one carried in the second Frame of a Superframe.

The duration of a Superframe is fixed in terms of Slots (and, therefore, Cells), but not in terms of time. The Superframe rate can be dynamically changed on SLIMbus™ to suit the particular application by changing either SLIMbus™ Root Frequency or Clock Gear, or both.

## **SLIMbus™ Clock Frequencies and Gears**

The SLIMbus™ specification does not specify a particular SLIMbus™ CLK frequency. Instead, three frequency definitions are provided: Root, Natural, and Cardinal.

The use of Natural Frequencies or Cardinal Frequencies is often convenient, but is not mandated on SLIMbus™.

Also included is the Clock Gear construct which provides 10 Gears for altering the clock frequency by powers of two.

There is also a defined procedure to pause or stop the CLK signal and re-start it.

### **Root Frequency**

The Root Frequency is defined as  $2^{(10-G)}$  times the frequency of the CLK line, where G is the current Clock Gear. In Gear 10, the CLK frequency is the same as the Root Frequency.

The Root Frequency may be a Natural or a Cardinal Frequency, but is not mandated to be so. The Root Frequency may be any frequency up to 28 MHz.

The Root Frequency may be changed while the bus is active while leaving the Frame Structure unchanged, thus allowing scaling the bus power consumption according to the application.

### **Natural Frequencies**

A Natural Frequency is defined as a CLK frequency that allows a family of isochronous data flows to be supported without flow control, thereby simplifying channel allocation on SLIMbus™.

For example, Natural Frequencies for 11.025 kHz and 44.1 kHz digital audio sample rate flows include 5.6448 MHz, 11.2896 MHz, and 22.5792 MHz. Similarly, Natural Frequencies for 8 kHz and 48 kHz digital audio sample rate flows include 6.144 MHz, 12.288 MHz, and 24.576 MHz.

The different sample rate flows can be optimally supported by changing a particular Natural Frequency by 0.5X or 2X using the Clock Gear.

## Cardinal Frequencies

In audio applications, one important family of sample rates is comprised entirely of multiples of 4 kHz, i.e. 8, 12, 16, 24, 32, 48, 96 kHz, etc. Another sample rate family is comprised entirely of multiples of 11.025 kHz, i.e. 11.025, 22.05, 44.1, 88.2 kHz, etc.

Concurrent support for digital audio data flows, or families of flows, with non-integer frequency relationships is sometimes required, e.g. 8 kHz and 44.1 kHz sample rates. In these cases, the CLK line cannot be set to a frequency that is a Natural Frequency for all flows on the bus.

The CLK line frequencies 24.576 MHz, 12.288 MHz, 6.144 MHz, etc. have special significance because they can carry the 4 kHz family of flows isochronously, and the 11.025 kHz family of flows with relatively high efficiency (using push or pull flow control techniques). For this reason, these clock frequencies are referred to as Cardinal Frequencies.

## Clock Gears

There are ten Clock Gears (1 to 10), providing a frequency span of 512 from lowest to highest Gear. Clock Gears provide a range of power-of-2 frequency steps for operating SLIMbus™. For example, increasing to the next Clock Gear without changing the Root Frequency doubles the SLIMbus™ CLK frequency, while decreasing the next Clock Gear without changing the Root Frequency will halve the SLIMbus™ CLK frequency.

## SLIMbus™ Messaging

SLIMbus™ provides a robust set of Messages for bus management, Device control, and data transfer. Basic types of Messages are:

- Core Messages, which include
  - Device Management Messages
  - Data Channel Management Messages
  - Information Management Messages
  - Reconfiguration Messages
  - Value Management Messages
- Destination-referred Device Class-specific Message
- Destination-referred User Message
- Source-referred Device Class-specific Message
- Source-referred User Message
- Escape

## Message Channel

The Message Channel is used by a Device for sending Messages to other Devices on the bus. In order to transmit or receive Messages, a Component shall first acquire Message Synchronization. Prior to sending any Messages, each SLIMbus™ Device uses a priority-based Arbitration mechanism to gain access to the Message Channel.

## Message Channel Size

Messages are carried in Message Channels in the Control Space.

Recall from Figure 8 above that there are four possible subframe modes in SLIMbus™. Each Subframe has at least one Slot as Control Space. The Framing Channel occupies two Slots per Frame, or 16 Slots per Superframe. The Guide Channel (used for Message Synchronization) occupies two Slots per Superframe for all bus configurations. Therefore, a total of 18 Slots of available Control Space per Superframe are allocated to specific purposes.

Control Space Slots not used for the Framing Channel or the Guide Channel are available for the Message Channel, or a mix of Message Channel and Data Channels. Therefore, the size of the Message Channel varies according to the bus configuration.

A minimum number of Slots for Message Channel use occurs in the 6 Subframes/Frame mode. So, there is a minimum of six Control Space Slots per Frame, and therefore, 48 Control Space Slots per Superframe. Since 18 Slots are allocated for bus purposes, then that leaves 30 ( $48 - 18 = 30$ ) Control Space Slots available for Message Channel use.

The maximum number of Slots for Message Channel use would just be the total number of Slots in a Superframe less the pre-allocated 18 Slots for bus purposes; e.g.  $1536 - 18 = 1518$ .

## Message Syntax

Messages which are exchanged between Devices consist of four fields; Arbitration, Header, Payload, and Integrity/Response. All messages must contain the Arbitration, Header, and Integrity/Response fields.

The Arbitration field is present in all Messages and varies in size (2- or 7-bytes) because the Source Address of the Device requesting arbitration could be either an 8-bit Logical Address (LA) or a 48-bit Enumeration Address (EA).

The Header field is present in all Messages and varies in size (3-, 4- or 9-bytes) because it, too, may use either no address, the Destination Device Logical Address or Enumeration Address. If all Devices receive a Broadcast Message, then the LA or the EA are not used.

Depending upon the type of message, the Message Payload field may be of zero length. The maximum length of the Message Payload field varies between 22- or 28-bytes depending upon message-specific parameters and data.

The Integrity/Response field is a fixed length of 4-bytes. The Primary and Message Integrity Fields provide full CRC coverage of the Message contents.

All Messages are associated with Message Codes which are specific to a Message Type.

The maximum Message length is 39-bytes.

A discussion of the contents of all the Message Fields is beyond the scope of this article.

## Core Messages

Core Messages are messages which are interpreted in the same way by all Devices. They are intended for managing and operating the SLIMbus™ and SLIMbus™ Devices, and consist of Device Management, Data Channel Management, Information and Value Element Management, and Bus Reconfiguration messages.

Depending upon the type of Device, the Device may or may not support all Core Messages.

However, there is a subset of the Core Messages that exist in each of the four Device Classes defined in the SLIMbus™ Specification which all Devices must support. Accordingly, depending upon the type of Device, there may also be a subset of support required in Devices for Information and/or Value Element data read and write.

## Other SLIMbus™ Messages

In addition to the Core Messages, additional message types are:

- Destination-referred Device Class specific Message
- Destination-referred User Message
- Source-referred Device Class specific Message
- Source-referred User Message

## Bus Boot and Bus Processes

The bus boot process is defined relative to Components, so the terms Clock Source Component and Clock Receiver Component are used to distinguish the Component that contains the active Framer from other Components.

The Clock Source Component has its own boot process, while the Clock Receiver Component type follows a different, but defined, boot process. The boot process occurs as each Component progresses in stages from an *Undefined* state to an *Operational* state.

The only time there is a single state that represents all Components on the bus is when all Components are in their respective *Operational* state at the same time.

A Component joins the bus by following its appropriate boot process.

In order to have more control over system power consumption, the SLIMbus™ protocol allows a Component to stop participating on the bus while the SLIMbus™ is active, and rejoin at a later time.

Rules related to each state allow Components that lose sync for whatever reason to drop to the next lower state or the *Reset* state and attempt the boot process again.

## Clock Source Boot Sequence

The Clock Source Component boots SLIMbus™ by a specific sequence of operations for starting CLK, and for writing the information in the Framing Channel and Guide Channel on the DATA line.

During the boot process, the Clock Source Component moves through five different states: *Undefined*, *CheckingDataLine*, *StartingClock*, *StartingData*, and *Operational*.

A power on reset signal or other specific external event is used to initially move from the *Undefined* state to the *CheckingDataLine* state.

Upon reaching the *Operational* state, the Clock Source Component has successfully written all the necessary Framing Information on the DATA line, and has initially configured the bus in terms of CLK Frequency and Subframe mode. Lastly, it will also report its presence (REPORT\_PRESENT Message) on the Bus via the Message Channel.

## **Clock Receiver Boot Sequence**

Any other Device within a Component on the bus that is not a Framer is, by definition, a Clock Receiver Component. Clock Receiver Components have their own bus boot process which occurs concurrently with the Clock Source Component, but completes after the Clock Source Component has completed its boot process.

During the boot sequence, a Clock Receiver Component extracts any necessary information about the current state of the bus by monitoring the DATA and CLK lines. During the boot process, Clock Receiver Components move through various states; *Undefined*, *Reset*, *SeekingFrameSync*, *SeekingSuperframeSync*, *SeekingMessageSync*, and finally, *Operational*.

A power on reset signal or other specific external event is used to initially move from the *Un-defined* state to the *Reset* state.

Once the *Operational* state is achieved, a Clock Receiver Component has all of the Framing Information about the Bus Configuration, has access to the Message Channel, and is now able to perform its function when required. It, too, announces its presence on the bus by a REPORT\_PRESENT Message.

## **Device Discovery**

As Devices become Operational after Bus Boot, they arbitrate for the Message Channel, and send REPORT\_PRESENT Messages to the Active Manager. These messages identify the Device Class code and version and the Enumeration Address of the reporting Device.

Upon acknowledgement of the REPORT\_PRESENT message by the Manager, the Device waits for the Manager to assign a Logical Address, which is needed in order to receive and send data Source and Sink types of messages. Once the Logical Address is assigned, the Device is considered to be in the *Enumerated* state, and ready for functional operation.

The active Manager assigns unique Logical Addresses at the bus level so that no two Devices try to use the same Logical Address at the same time.

If for any reason the Device loses Message Sync after receiving its Logical Address, there is no need to assign a new Logical Address after the Device recovers Message Sync and re-joins the bus.

A Component *Reset* will cause the loss of an assigned Logical Address, and the Device Discovery process will need to be run again on that Component before it can join the bus.

## **Bus Management**

Many SLIMbus™ processes require that all active Components change state at exactly the same time, e.g. at a Superframe boundary. This is accomplished by a Message string which begins with a

BEGIN\_RECONFIGURATION Message, followed by a number of Reconfiguration Messages, and ending with a RECONFIGURE\_NOW Message.

The Reconfiguration Messages are also used to adapt SLIMbus™ to the CLK frequency, data rate, and numbers of data channels to match the particular application demands running at the time. Thus, bus power consumption is also managed.

## Conclusion

SLIMbus™ is a highly configurable multi-drop bus structure able to support many components simultaneously. In addition, there are powerful messaging constructs to set up and manage data flow between components on the bus. SLIMbus™ also provides the ability to re-configure the bus operational characteristics on-the-fly to adapt to particular system application needs at runtime.

Unlike traditional digital audio bus structures, SLIMbus™ has the ability to simultaneously and efficiently carry multiple digital audio data streams at widely differing sample rates and bit widths.

When using existing digital audio interfaces (PCM, I<sup>2</sup>S, SSI, AC-97), adding functions and digital audio channels to mobile terminals beyond those for voice communication and simple stereo music applications is very difficult without increasing the number of bus structures in the mobile terminal. This is because these legacy interfaces are primarily point to point (peer to peer) connections with limited channel capacity, and any new device in the system requires its own interface connection.

Though legacy interface systems are scalable just by duplicating interface structures, this approach limits design flexibility, and is costly in terms of pin count, package size, PCB layout area, and power consumption.

SLIMbus™ provides the mobile terminal industry and other small form factor product makers a standard, robust, scalable, low-power, high-speed, cost-effective, two wire multi-drop interface that supports a wide range of digital audio and control solutions. As such, it effectively replaces legacy digital audio interfaces such as PCM, I<sup>2</sup>S, and SSI in such products.

SLIMbus™ can also effectively replace some instances of many digital control buses such as I<sup>2</sup>C, SPI, or UART and GPIO, in a mobile terminal or portable product by providing flexible and dynamic assignment of bus bandwidth between digital audio and non-audio control and data functions.

SLIMbus™ is not backward compatible with any current digital audio bus or digital control bus.

Implementing the SLIMbus™ standard greatly increases the flexibility for designers to realize multiple products within a product line quickly, each with widely varying digital audio and user interface features. This can all be done without the duplication of multiple bus structures within the product.

SLIMbus™ reduces the time-to-market and design cost of mobile terminals and other portable devices by simplifying the interconnection of various functional products from different manufacturers.

Most company participants in the creation of the SLIMbus™ Specification have SLIMbus™ compatible product development underway with the goal that products suitable for a basic SLIMbus™ system would be available to the industry in 2008. For more information on the SLIMbus™ Specification, visit [www.mipi.org](http://www.mipi.org).