OS81210

Intelligent Network Interface Controller for 50 Mbit/s Automotive Networks

Features

- Complete 50 Mbit/s synchronous network interface
- Embedded network management functions
  - Network protection mode
  - Hardware & application watchdog timer
  - Intelligent muting
  - Diagnostics
  - Fallback Operation
- IEEE MAC addressing and Ethernet channel
- Universal Serial Bus (USB) Port supports USB 2.0 High-speed upstream data transfers using either:
  - USB 2.0 physical layer
  - High-Speed Inter-Chip (HSIC) physical layer
- Media Local Bus (MediaLB®) Port
  - Eases inter-chip communication and streaming
  - MediaLB 3-pin interface at speeds up to 1024xFs
- I^{2}C™ Control Port inter-chip message exchange
- Streaming Port supports synchronous, fixed latency data exchange for a variety of serial audio formats including time-division multiplex (TDM) and pulse density modulation (PDM)
- SPI Port supports asynchronous and control packets
- General Purpose I/O (GPIO) Port
- Remote control and configuration for operation without a local External Host Controller.
  - I^{2}C (master) message tunneling
  - GPIO port control
- Operating voltages 3.3 V/1.8 V (and 1.2 V for HSIC)
- Available in 64-pin QFN package with exposed pad
- -40 to +125 °C junction temperature

Conformity

- This document applies to hardware revision B2B

Applications

- Automotive infotainment network nodes including head unit, instrument cluster, amplifier, and rear seat entertainment.

General Description

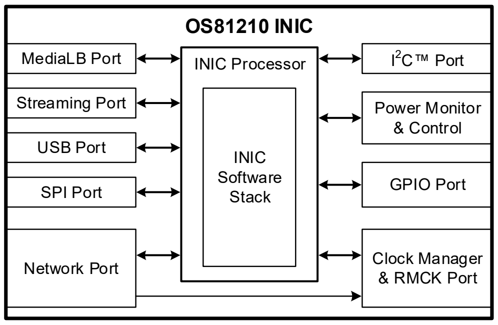
The OS81210 is a highly integrated *Intelligent Network Interface Controller* (INIC) for 50 Mbit/s INICnet-based automotive networks with a transformer-less balanced media physical layer (bPHY) optimized for unshielded twisted pair (UTP) copper wire.

The INIC provides encapsulation of all low-level functions necessary to develop a network-compliant device, significantly simplifying network implementation in a node. Integration of the *INIC Software Stack* into the INIC provides network-compliant real-time behavior. The *INIC Software Stack* significantly relieves the External Host Controller (EHC) from real-time processing tasks. Supervision of the application is also provided, including a protection mode that is entered when an application is not present (i.e. start-up) or the EHC malfunctions. This protection mode prevents application malfunctions from influencing the integrity of the network and the system.

When the EHC is engaged, a message-based interface, as opposed to a register-based interface, is available for communication with INIC. A unified and centralized network management software stack (UNICENS) is available for the EHC to build a complete, lean, system solution.

The INIC conforms to the ISO 21806 standard developed by the International Organization for Standardization (ISO®).

FIGURE: OS81210 BLOCK DIAGRAM



TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com** to receive the most current information on all of our products.

TABLE OF CONTENTS

Conventions

The following abbreviations and symbols are used to improve readability.

Example	Description
BIT	Name of a single bit within a field
FIELD.BIT	Name of a single bit (BIT) in FIELD
x…y	Range from x to y, inclusive
BITS[m:n]	Groups of bits from m to n, inclusive
PIN	Pin Name
SIGNAL	Signal Name
msb, lsb	Most significant bit, least significant bit
MSB, LSB	Most significant byte, least significant byte
zzzzb	Binary number (value zzzz)
0xzzz	Hexadecimal number (value zzz)
zzh	Hexadecimal number (value zz)
rsvd	Reserved memory location. Must write 0, read value indeterminate
`code`	Instruction code, or API function or parameter
Multi Word Name	Used for multiple words that are considered a single unit, such as: *Resource Allocate* message, or *Connection Label*, or *Decrement Stack Pointer* instruction.
Section Name	Emphasis, Reference, Section or Document name.
$\overline{\text{VAL}}$	Over-bar indicates active low pin or register bit
x	Don’t care
<Parameter>	<> indicate a Parameter is optional or is only used under some conditions
{,Parameter}	Braces indicate Parameter(s) that repeat one or more times.
[Parameter]	Brackets indicate a nested Parameter. This Parameter is not real and actually decodes into one or more real parameters.

1.0 OS81210 PINOUT

Input pins must not be left floating; therefore, they must be driven, have pull-ups or pull-downs, or connected directly to a power pin or ground.

Digital pins that can be configured as outputs (i.e., pin types D_{OUT}, D_{OUTD}, D_{OUTZ}, $D_{I/O}$, and $D_{I/OD}$) are high impedance during power-up/reset. The pin types shown in Table 1-1 are the values after power-up/reset.

Note: Hardware port pins other than the I^2C Port and JTAG Port pins are high impedance until opened/enabled. This includes the **RMCK** pin, which remains high impedance until the output clock is specifically enabled.

FIGURE 1-1: OS81210 PIN DIAGRAM

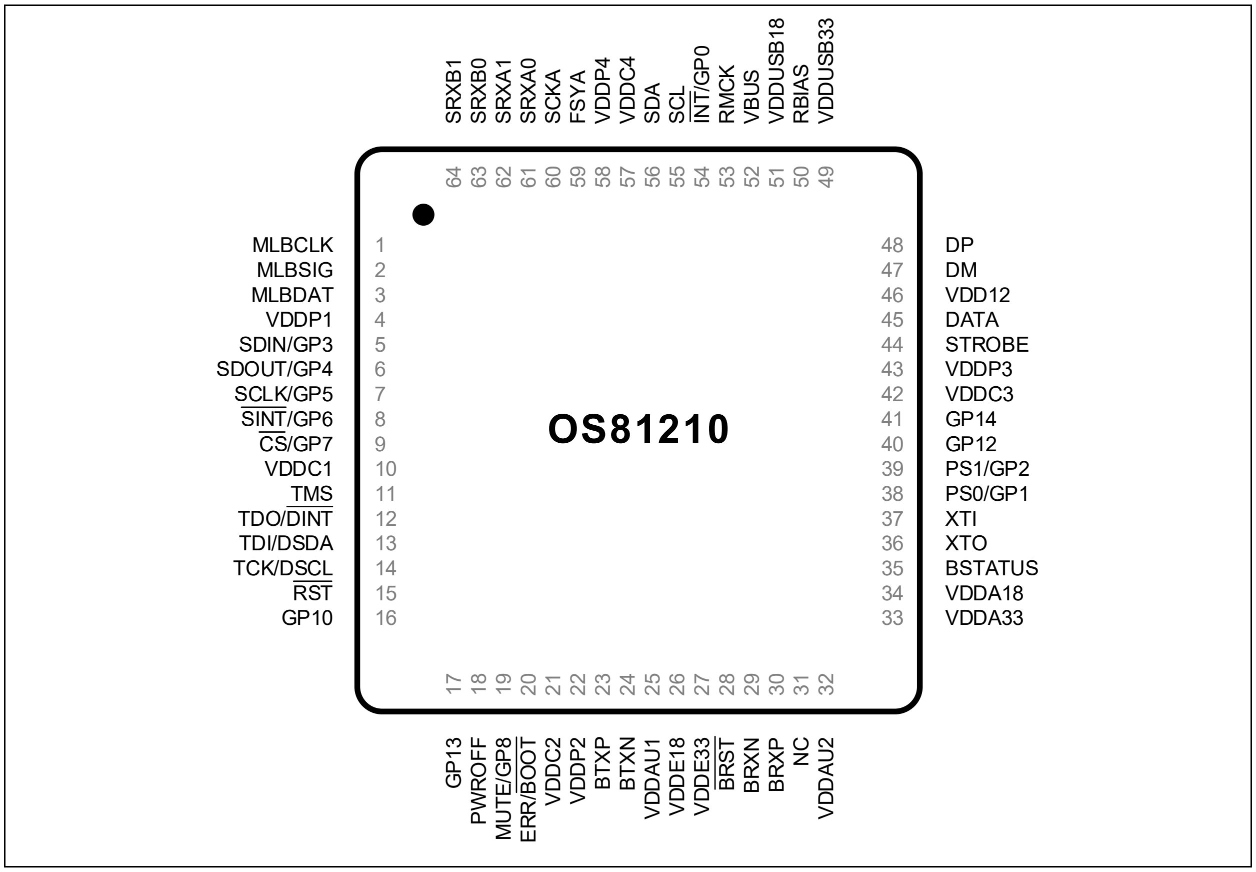


TABLE 1-1: OS81210 PIN ALLOCATION TABLE

Pin	Name	Type	HW Port	Description
1	MLBCLK [2]	D_{OUT}	MediaLB	Singled-ended Clock line for MediaLB 3-pin Interface
2	MLBSIG [2]	$D_{I/O}$	MediaLB	Singled-ended Signal line for MediaLB 3-pin Interface
3	MLBDAT [2]	$D_{I/O}$	MediaLB	Singled-ended Data line for MediaLB 3-pin Interface
4	VDDP1			3.3 V periphery power supply (digital)
5	SDIN	D_{IN}	SPI	Data In (MOSI - Master Out, Slave In)
	GP3	$D_{I/O}$	GPIO	General Purpose Input/Output 3
6	SDOUT	D_{OUT}	SPI	Data Out (MISO - Master In, Slave Out)
	GP4	$D_{I/O}$	GPIO	General Purpose Input/Output 4

Note 1: Pull-up resistor required.

2: Pull-down resistor required.

TABLE 1-1: OS81210 PIN ALLOCATION TABLE (CONTINUED)

Pin	Name	Type	HW Port	Description
7	SCLK	D_{IN}	SPI	Clock
	GP5	$D_{I/O}$	GPIO	General Purpose Input/Output 5
8	$\overline{SINT}$	D_{OUT}	SPI	Interrupt (active low)
	GP6	$D_{I/O}$	GPIO	General Purpose Input/Output 6
9	$\overline{CS}$	D_{IN}	SPI	Chip Select (active low)
	GP7	$D_{I/O}$	GPIO	General Purpose Input/Output 7
10	VDDC1			1.8 V core power supply (digital)
11	TMS[1]	D_{IN}	JTAG	Test Mode Select
12	TDO[1]	D_{OUTZ}	JTAG	Test Data Output
	$\overline{DINT}$[1]	D_{OUTD}		Debug Interrupt (active low)
13	TDI[1]	D_{IN}	JTAG	Test Data Input
	DSDA[1]	$D_{I/OD}$		Debug Data
14	TCK[1]	D_{IN}	JTAG	Test Clock Input
	DSCL[1]	$D_{I/OD}$		Debug Clock
15	$\overline{RST}$	D_{IN}		Hardware Reset Input (active low). (Pull-up resistor to **VDDPn** supply should be used when not driven high by an external device. A series resistor should be used in lieu of the pull-up when always driven by an external device.)
16	GP10	$D_{I/O}$	GPIO	General Purpose Input/Output 10
17	GP13	$D_{I/O}$	GPIO	General Purpose Input/Output 13
18	PWROFF[1]	D_{OUTD}		External Power Management Power-Down Indicator. This pin is driven low by INIC after initialization. When high, indicates that the INIC Processor is ready to be shut down. A pull-up resistor is required when used. If not used, this pin may be left unconnected.
19	MUTE[1]	D_{OUTD}		Mute Indicator Output. A pull-up resistor is required when used. If not used, this pin may be left unconnected.
	GP8	$D_{I/O}$	GPIO	General Purpose Input/Output 8
20	ERR	D_{OUT}	Network	Network Error Indicator Output. This pin is driven high when the network is unlocked. When low, this pin indicates the INIC is locked to the network.
	$\overline{BOOT}$[1]	D_{IN}		Configuration Pin. This pin is attached to the configuration/debug header and used by the Microchip INICkit Tool [3] to load initial configuration data into INIC. May also be connected to the EHC to allow in-system configuration of the INIC.
21	VDDC2			1.8 V core power supply (digital)
22	VDDP2			3.3 V periphery power supply (digital)
23	BTXP	$A_{I/O}$	Network	Positive (differential) bPHY network transmitter output
24	BTXN	$A_{I/O}$	Network	Negative (differential) bPHY network transmitter output
25	VDDAU1			3.3 V continuous power supply (analog)
26	VDDE18			1.8 V bPHY power supply (analog)
27	VDDE33			3.3 V bPHY power supply (analog)
28	$\overline{BRST}$[1]	$A_{I/O}$	Network	Hardware Reset Input (active low) for the Balanced Media Physical Layer. When asserted, the transmitter output is disabled. A pull-up resistor to **VDDPn** is required.
29	BRXN	$A_{I/O}$	Network	Negative (differential) bPHY network receiver input
30	BRXP	$A_{I/O}$	Network	Positive (differential) bPHY network receiver input
31	NC			No Connect. This pin must be left open and floating.

Note 1: Pull-up resistor required.

2: Pull-down resistor required.

TABLE 1-1: OS81210 PIN ALLOCATION TABLE (CONTINUED)

Pin	Name	Type	HW Port	Description
32	VDDAU2			3.3 V continuous power supply (analog)
33	VDDA33			3.3 V power supply (analog)
34	VDDA18			1.8 V power supply (analog)
35	BSTATUS	D_{OUT}	Network	bPHY Network Activity Status Output used during wake-up: - Driven low when a valid signal is detected - Driven high to **VDDAUn** when a qualified signal is not present
36	XTO	$A_{I/O}$		Crystal Oscillator Output
37	XTI	$A_{I/O}$		Crystal Oscillator Input or External CMOS Clock Input
38	PS0	D_{IN}		External Power Management Status Bit 0
	GP1	$D_{I/O}$	GPIO	General Purpose Input/Output 1
39	PS1	D_{IN}		External Power Management Status Bit 1
	GP2	$D_{I/O}$	GPIO	General Purpose Input/Output 2
40	GP12	$D_{I/O}$	GPIO	General Purpose Input/Output 12
41	GP14	$D_{I/O}$	GPIO	General Purpose Input/Output 14
42	VDDC3			1.8 V core power supply (digital)
43	VDDP3			3.3 V periphery power supply (digital)
44	STROBE	$D_{I/O}$	USB	Strobe line for HSIC physical interface. Connect to **GND** when HSIC is not used.
45	DATA	$D_{I/O}$	USB	Data line for HSIC physical interface. Connect to **GND** when HSIC is not used.
46	VDD12			1.2 V power supply for HSIC physical interface transceiver. Connect to **GND** through a 1 kΩ resistor when HSIC is not used.
47	DM	$A_{I/O}$	USB	Negative (differential) data line for USB physical interface. Connect to **GND** when the USB physical interface is not used.
48	DP	$A_{I/O}$	USB	Positive (differential) data line for USB physical interface. Connect to **GND** when the USB physical interface is not used.
49	VDDUSB33			3.3 V USB power supply (analog)
50	RBIAS [2]	$A_{I/O}$	USB	Connect to **GND** through a 12 kΩ resistor (0.5 %, 1/16 W, ≤ ± 100 ppm). This pin may be left unconnected when both the USB and HSIC physical interfaces are not used.
51	VDDUSB18			1.8 V USB power supply (analog)
52	VBUS	D_{IN}	USB	USB Bus Power State Indicator Input. The application should drive this pin high when an external USB Host Controller is present. Note that this pin is not 5 V tolerant and must not be connected directly to USB bus power. This signal is ignored when using the HSIC physical interface. Connect to **GND** when the USB physical interface is not used.
53	RMCK	D_{OUT}	RMCK	Recovered Master Clock Output
54	$\overline{\text{INT}}$ [1]	D_{OUTD}	I^2C	Interrupt (active low). Indicates a service request from the EHC when the Control Port is operating as an I^2C slave.
	GP0	$D_{I/O}$	GPIO	General Purpose Input/Output 0
55	SCL [1]	$D_{I/OD}$	I^2C	Clock
56	SDA [1]	$D_{I/OD}$	I^2C	Data
57	VDDC4			1.8 V core power supply (digital)
58	VDDP4			3.3 V periphery power supply (digital)
59	FSYA	$D_{I/O}$	Streaming	Frame Sync for Streaming Port A and B
60	SCKA	$D_{I/O}$	Streaming	Bit Clock for Streaming Port A and B
61	SRXA0	$D_{I/O}$	Streaming	Data I/O Signal 0 for Streaming Port A

Note 1: Pull-up resistor required.

2: Pull-down resistor required.

TABLE 1-1: OS81210 PIN ALLOCATION TABLE (CONTINUED)

Pin	Name	Type	HW Port	Description
62	SRXA1	$D_{I/O}$	Streaming	Data I/O Signal 1 for Streaming Port A
63	SRXB0	$D_{I/O}$	Streaming	Data I/O Signal 0 for Streaming Port B
64	SRXB1	$D_{I/O}$	Streaming	Data I/O Signal 1 for Streaming Port B
ePAD	GND			The exposed paddle on the bottom side of the QFN package is the primary ground for the OS81210 and must be connected to ground on the PCB for proper operation.

Note 1: Pull-up resistor required.

2: Pull-down resistor required.

2.0 EQUIVALENT SCHEMATICS FOR PINS

FIGURE 2-1: PIN-EQUIVALENT FOR ANALOG I/O PIN - $A_{I/O}$

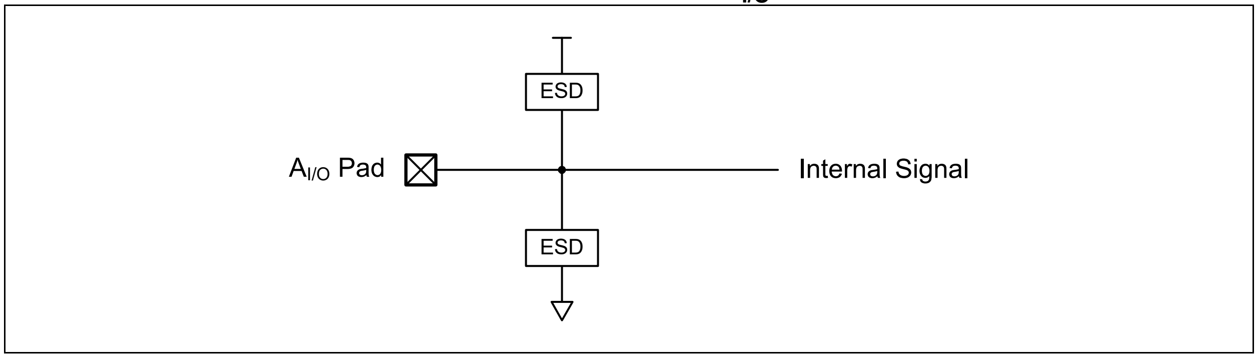


FIGURE 2-2: PIN-EQUIVALENT FOR DIGITAL INPUT PIN - D_{IN}

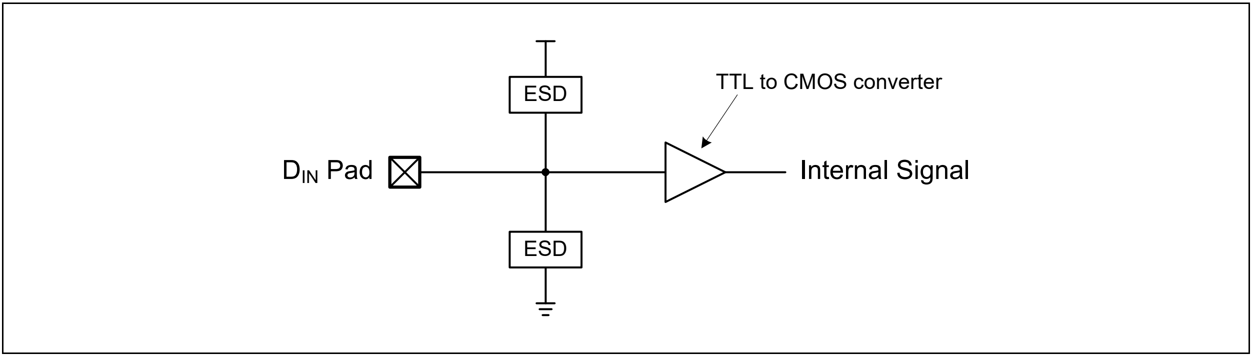


FIGURE 2-3: PIN-EQUIVALENT FOR DIGITAL OUTPUT PIN - D_{OUT}

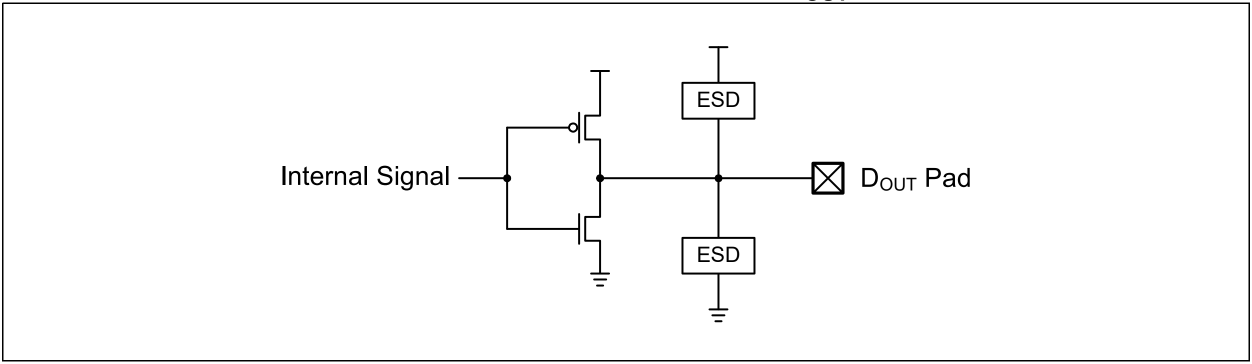


FIGURE 2-4: PIN-EQUIVALENT FOR OPEN-DRAIN DIGITAL OUTPUT PIN - D_{OUTD}

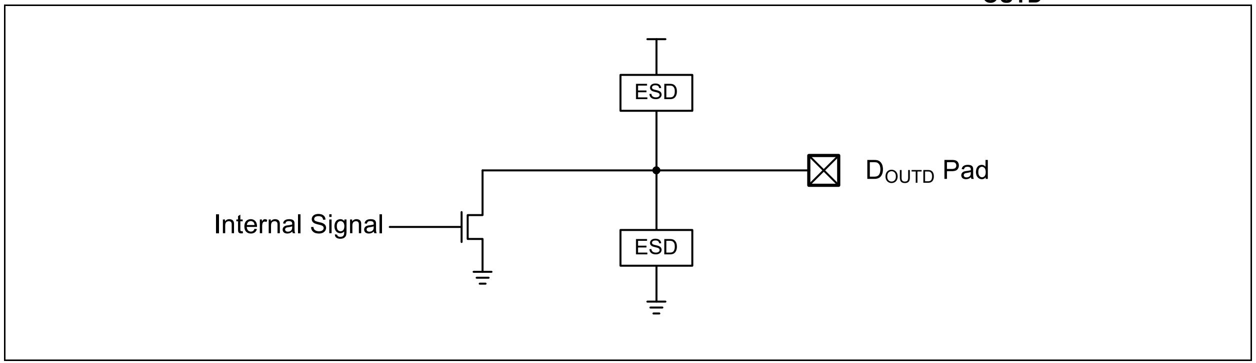


FIGURE 2-5: PIN-EQUIVALENT FOR DIGITAL OUTPUT PIN WITH HIGH-Z CONTROL - D_{OUTZ}

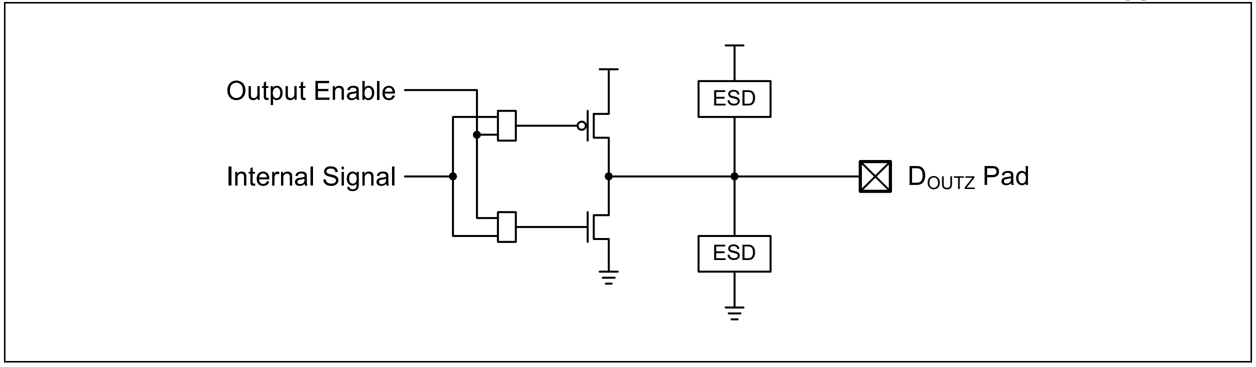


FIGURE 2-6: PIN-EQUIVALENT FOR DIGITAL I/O PIN - $D_{I/O}$

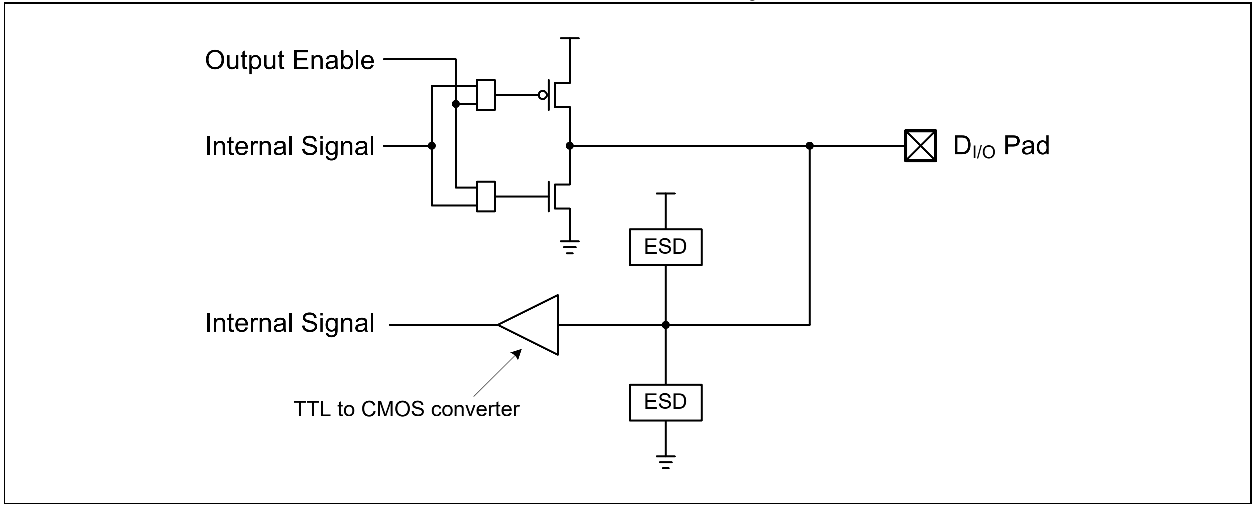
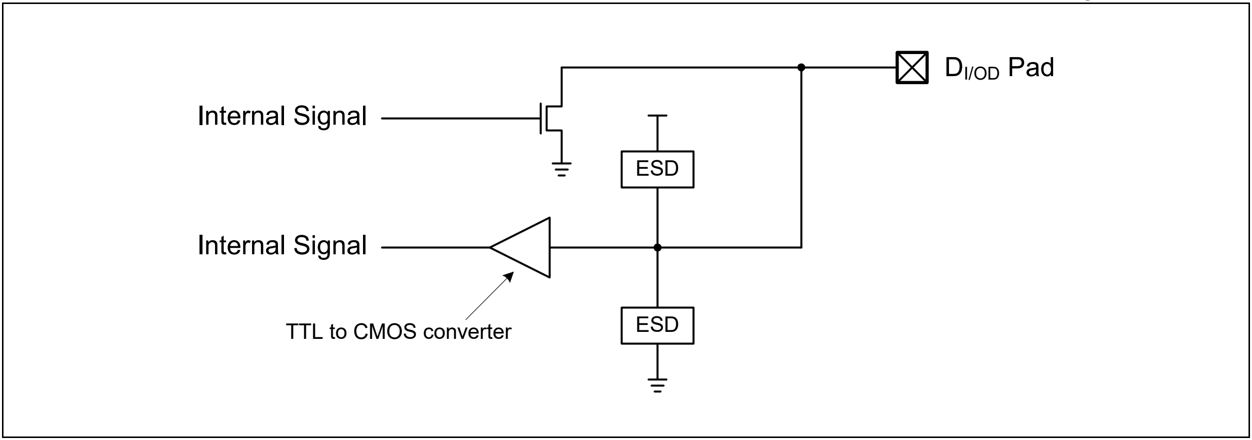


FIGURE 2-7: PIN-EQUIVALENT FOR DIGITAL INPUT/OPEN-DRAIN OUTPUT PIN - $D_{I/OD}$



3.0 OVERVIEW

Note: This Data Sheet is designed to be used in conjunction with the INIC Interface Specification [4] to provide a complete reference for the operation and use of the OS81210.

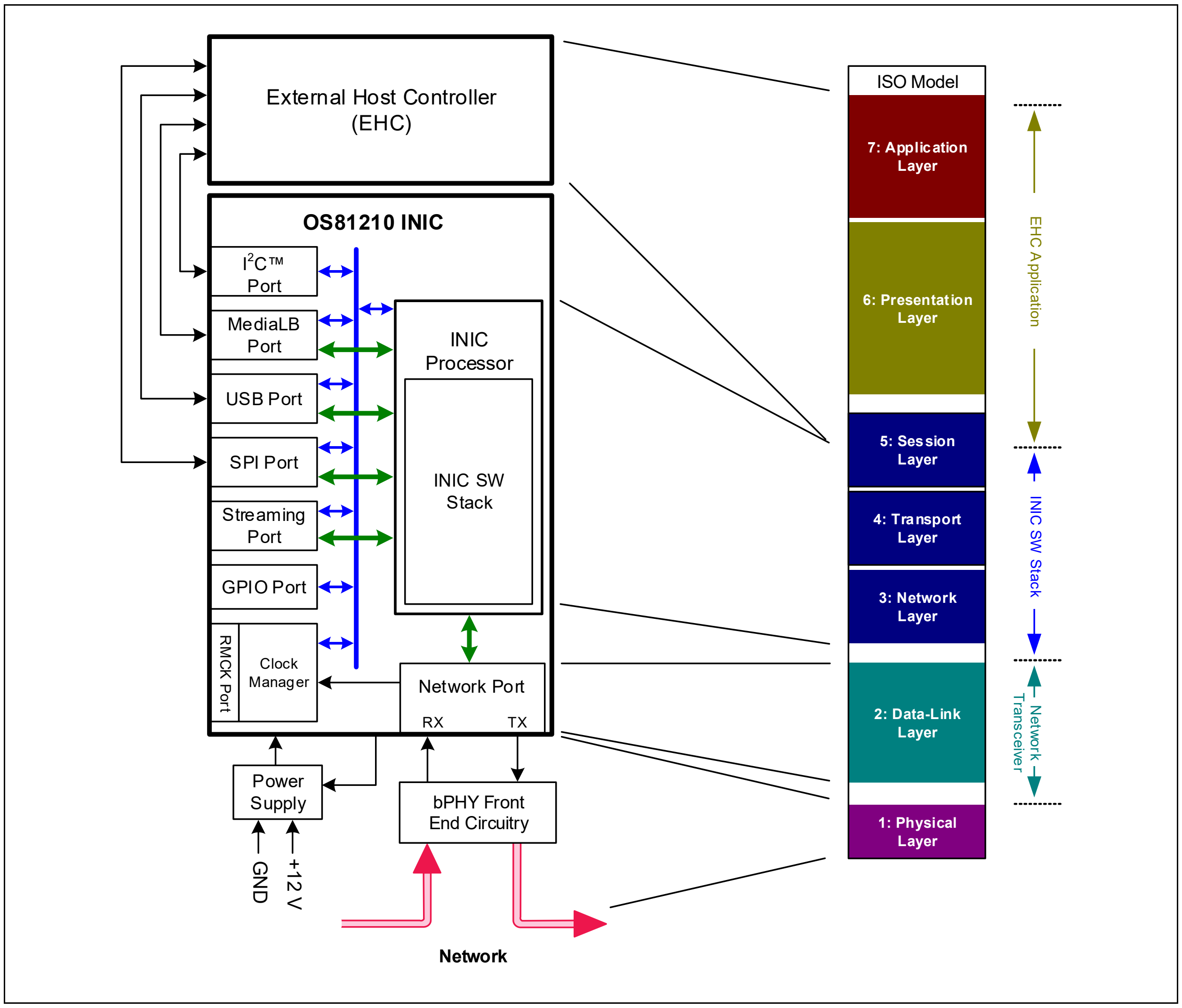
The OS81210 is a network transceiver device belonging to the Microchip *Intelligent Network Interface Controller* (INIC) family. The OS81210 INIC creates a time division multiplexed synchronous network by transmitting 128 bytes frames at a 48 kHz frame rate (Fs) yielding a 49.2 Mbit/s data rate. A network of Electronic Control Units (ECUs) based on INICs creates an INICnet network. INICnet can be configured and managed at the application level by the provided UNICENS software stack. Alternatively, the MOST NetServices software stack can be used to create a MOST compliant network.

There are two major differences between a MOST compliant network and a UNICENS based network, besides meeting the requirements of the MOST Specification. First, the configuration of the network and routing of streaming data on a UNICENS network is done by one central node. In a MOST network, the configuration is distributed among the nodes, with each node being responsible for connecting to a network resource under the direction of the local EHC. Since UNICENS does this configuration under a single node (Root Node), it enables a new class of nodes, called Slim Nodes (Remote Nodes) that do not require a local processor. Second, the MOST Specification describes a dedicated control channel and a specific MOST message format for messages between applications. Whereas in a UNICENS network, it is assumed that application messages between nodes will be transmitted over the Ethernet channel in the form of IP messages using an OEM defined protocol such as SOME/IP, MQTT, or any other OEM defined protocol.

All relevant network management functions are handled on-chip, providing a complete system interface to the physical layer components. Minimal additional components are required due to the high-level of integration. An on-chip PLL with ultra-low jitter guarantees accurate audio and video transmission and clock recovery over a wide frequency range.

A typical network node consists of the physical layer devices connected to the network, the OS81210 to handle the low-level protocols, and an *External Host Controller* (EHC) for the mid- to high-level functions. This architecture eliminates the need for the user to implement the lower protocol levels required by the ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications [1], the ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification [2], or the UNICENS stack, thereby drastically shrinking development time. Network management functions are off-loaded from the programmer, allowing full concentration on the application being developed. Depending on the embedded firmware, the functions and API of the low-level protocol can vary. Refer to the INIC Interface Specification [4] for more information. Figure 3-1 illustrates the OS81210 with the protocol stack implementation.

FIGURE 3-1: INIC HARDWARE/SYSTEM OVERVIEW



3.1 Network

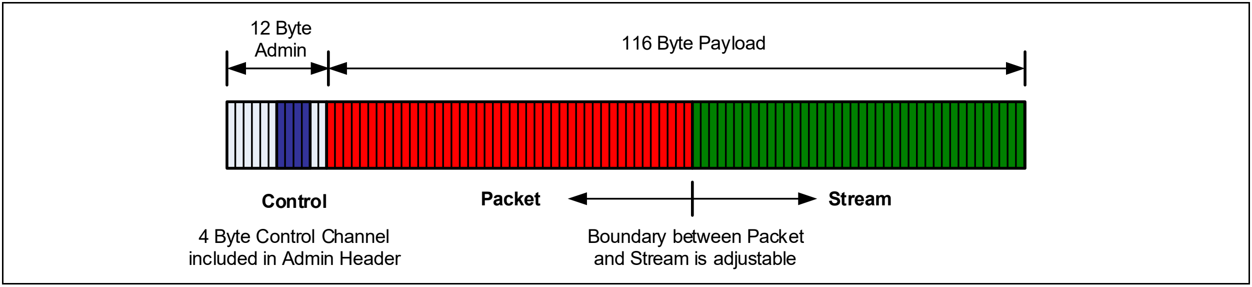
To minimize costs, the network supports a peer-to-peer methodology, eliminating excessive hardware overhead such as a hub (although hub-based architectures can also be implemented). In addition to handling network interface and communication management functions, the OS81210 INIC also handles all of the important low-level network management functions such as node position sensing (Plug-and-Play), start up, and shut down. Other features include error reporting, fail-safe operation, and channel allocation. Placing these features in hardware off-loads the EHC, allowing it to focus on higher-level network functions.

The network implements multiple simultaneous data transport methods operating across a single medium. Once bandwidth is allocated, each data transport method operates independently (not affecting the others), providing a robust, dependable, and deterministic system architecture. The OS81210 INIC supports all of the network data transport methods, including:

- Control - consists of a single *control channel*
- Asynchronous (Packet) - consists of a single *packet channel*
- Synchronous Streaming - consists of one or more *synchronous streaming channels*
- Isochronous Streaming - consists of one or more *isochronous channels*

The network frame's channel boundaries are shown in Figure 3-2.

FIGURE 3-2: NETWORK FRAME CHANNELS


12 Byte Admin
116 Byte Payload
Control
Packet
Stream
4 Byte Control Channel included in Admin Header
Boundary between Packet and Stream is adjustable

3.1.1 CONTROL

The INIC has a unique 4 byte (1.5 Mbit/s data rate) control channel shared by all network nodes. Applications use this channel to transmit control messages between nodes. These control messages from the EHC application are referred to as *Application Control Frames. Application Control Frames* are transported as ICMs, RCMs, or MCMs between an EHC driver and a INIC peripheral port (i.e. I^{2}C, MediaLB, USB, or SPI). *Application Control Frames* are then transported as *Control Frames* on the network's control channel. Any network node that needs to transmit a *Control Frame* on the network must arbitrate for the control channel. The arbitration is handled by INIC. UNICENS and NetServices libraries provide API commands to send and receive *Application Control Frames* which abstracts the internal INIC software stack from the application.

The INIC automatically monitors the control channel on the Network Port and uses information in the header to determine whether the node is the intended receiver of the *Control Frame*. When a *Control Frame* is to be received (e.g. node address match), it is buffered within the INIC and made available to the application via one of the OS81210 peripheral hardware ports that supports RCM or MCM traffic (see Table 3-1).

3.1.2 ASYNCHRONOUS (PACKET)

The INIC has a single asynchronous packet channel with an adjustable bandwidth that is shared by all network nodes. Applications use this channel to transmit large data packets (e.g. navigation maps) in bursts to other network nodes. These data packets from the EHC application are referred to as *MHP Frames* or *Ethernet Frames*. *MHP Frames* or *Ethernet Frames* are transported as MEPs (Meta Ethernet Packets) or MDP*s (Meta Data Packets)* from the EHC driver to the INIC peripheral port (MediaLB, USB, or SPI) where the message is then converted into a *Packet Frame* or *Ethernet Data Frame* to be sent across the network to the intended node. Any network node that needs to transmit a *Packet Frame* or *Ethernet Data Frame* can arbitrate for the packet channel. The arbitration is handled by INIC.

The INIC automatically monitors the packet channel on the Network Port and uses information in the *MHP Frame* or *Ethernet Frame* header to determine whether the node is the intended receiver (e.g. node address match). When a frame on the packet channel is to be received, the appropriate *MHP Frame* or *Ethernet Frame* is buffered within the INIC and made available as an MDP or MEP on one of the OS81210 peripheral ports that support MDP or MEP traffic (see Table 3-1).

3.1.3 SYNCHRONOUS STREAMING

The *Synchronous Streaming transport method* can consist of multiple synchronous streaming channels, each transporting raw, real-time synchronous data (e.g. audio, video). The data on a synchronous streaming channel is sourced by a single transmitting node that has been granted channel bandwidth. Synchronous streaming channels are broadcast, making the data available for reception by one or more sink nodes.

For high-speed synchronous data, INIC acts like a cross-point switch to connect synchronous streaming channels to sources/sinks attached to OS81210 hardware ports that support synchronous data (see Table 3-1).

3.1.4 ISOCHRONOUS STREAMING

The *Isochronous Streaming transport method* can consist of multiple isochronous channels, each transporting data that is sourced by a single transmitting node (no arbitration) that has been granted channel bandwidth. Isochronous channels are broadcast, making the data available for reception by one or more sink nodes.

Streaming data is transported across isochronous channels, albeit in a *structured* format. Various streaming data structures can be transmitted across an isochronous channel, including:

- DiscreteFrame Isochronous (DFI) Streaming Phase
- A/V Packetized (AVP) Isochronous Streaming Packets
- Quality of Service (QoS) IP Streaming Packets

3.1.4.1 DiscreteFrame Isochronous Phase

A *DiscreteFrame Isochronous* (DFI) phase channel transports the *clock phase component* (also referred to as DFI phase), which includes the phase information necessary to regenerate a specific clock frequency, asynchronous to the network, on the sink node. DFI phase is typically used in association with A/V Packetized (AVP) isochronous to transfer video clock data from the source encoder to the sink decoder. OS81210 hardware ports that support the exchange of DFI phase are shown in Table 3-1.

3.1.4.2 A/V Packetized (AVP) Isochronous Streaming Packets

A/V Packetized (AVP) isochronous streaming packets are used to transmit streaming data in an application-specific format (e.g. MPEG2-TS 188 bytes). Sink nodes access the data using a locally generated clock that is independent of the network frame rate, as well as the frame rate used by the node sourcing the AVP packets. OS81210 hardware ports that support the exchange of AVP packets are shown in Table 3-1.

3.1.4.3 Quality of Service (QoS) IP Streaming Packets

An isochronous channel can be used to transport *Ethernet Data Frames* over *dedicated* network bandwidth. In such a system, the *Ethernet Data Frames* are sourced by a single transmitting node that has been granted channel bandwidth. Although the same *Ethernet Data Frames* could alternatively be transmitted across the packet channel, arbitration prior to each packet transmission would be required. *Ethernet Data Frames* exchanged across an isochronous channel are referred to as *QoS Frames*, and are intended for Internet Protocol (IP) message applications.

QoS Frames are transmitted in a streaming, broadcast fashion, which makes the *QoS Frame* stream available for reception by one or more sink nodes. However, unlike sink nodes receiving messages on the asynchronous packet channel, nodes that need to sink *QoS Frames* must first be properly configured by the application for reception on a specific channel.

Once the INIC is attached to a network channel that transmits *QoS Frames*, the destination address in the packet headers is ignored and the *QoS Frame* is always received.

Due to the streaming nature of this data transport, received packets are not buffered in their entirety within INIC; rather, an internal circular buffer is used to route the stream of *QoS Frames* to the OS81210 hardware port that supports and is configured for *QoS Frame* reception (see Table 3-1).

3.2 INIC Processor

The INIC Processor manages the transfer of data between the network and other hardware ports. The Network Port connects the INIC Processor to the network using the OS81210 network transmit and receive pins (**BTXP**/**BTXN** and **BRXP**/**BRXN**) to exchange data over the bPHY interface. The bPHY interface of the OS81210 requires external *Analog Front-End (AFE)* circuitry to connect the INIC Processor to the balanced media network and increase system immunity to EMI.

Working in conjunction with the Clock Manager, the Network Port recovers the network clock, decodes received data, and passes the data to the INIC Processor. The INIC Processor then routes data to the appropriate destinations on and off the chip.

The network frame to be transmitted by the OS81210 is constructed by the INIC Processor by combining incoming network data from the Network Port with outgoing data from other on-chip resources. The Network Port then encodes the data for network transmission.

The *INIC Software Stack* runs on the INIC Processor and configures the Network Port at power-up, which allows the network to become operational without any external protocol stack. This has been accomplished by placing all the lower layers of the ISO protocol stack inside of the device. Table 3-1 illustrates the position of the *INIC Software Stack* within the ISO software model.

Each network node contains a general set of functions to manage the network. These functions are split with the INIC managing the low-level functions and the EHC managing the higher-level functions.

The INIC also contains functions used to configure the chip. This includes the setup of hardware ports and the usage of network bandwidth for transporting streaming data.

The upper levels of the network protocol must be implemented by an EHC that is capable of processing the user application with the minimal overhead imposed by the INIC API. During real-time operation, the EHC and the *INIC Software Stack* within the INIC Processor communicate using control messages via the I^{2}C, SPI, MediaLB, or USB Ports.

3.3 Hardware Ports

Each OS81210 hardware port transports different network data types. Table 3-1 shows the data types that can be transported across each hardware interface.

TABLE 3-1: HARDWARE PORT DATA TRANSPORT

	ICM	RCM, MCM	MEP, MDP	Synchronous Data	DFI Phase	AVP Packets	QoS (MEP)
Peripheral Hardware Ports:							
MediaLB	X	X	X	X	X	X	X
Streaming				X			
RMCK				X			
USB	X	X	X	X		X	
SPI	X	X	X				
I^{2}C Port	X	X					
Network Port:							
Network Channel	**Control**	**Control**	**Packet**	**Synchronous**	**Isochronous**	**Isochronous**	**Isochronous**
		X	X	X	X	X	X

Using the OS81210 API, a *socket* is created for each data stream transported across a hardware port. The specific data type to be exchanged is specified when creating the socket. Data routing between hardware ports is setup when sockets of like data types are linked together. For more information on sockets, refer to the INIC Interface Specification [4].

3.3.1 NETWORK PORT

The Network Port receiver is over-sampled at a high-frequency and data is recovered by a digital state machine. When the port is configured as a timing-slave, the OS81210 recovers the network clock, which the Clock Manager then uses to generate other internal clocks. When the port is configured as the timing-master, the internal clocks and the network clock are generated by the Clock Manager based on an external crystal (**XTI/XTO**). The Network Port transmitter works in conjunction with the INIC Processor to encode data for network transmission.

After reset, the OS81210 automatically interacts with the network, performing all necessary low-level network management functions. This enables the EHC to configure its interface to the OS81210 when the application is ready.

3.3.2 MediaLB PORT

The MediaLB Port supports communication between the EHC and the OS81210 INIC by means of the *Media Local Bus* (MediaLB) protocol. MediaLB provides a low cost, easy to implement hardware interface that standardizes and simplifies network application development. MediaLB has one bus master, referred to as the *MediaLB Controller* (or simply *Controller*), which is always the OS81210. All other connected components (including the EHC) are known as *MediaLB Devices* (or simply *Devices*). MediaLB Device functionality is a subset of Controller functionality; the Controller (OS81210) functions as a Device when it is receiving data from other connected components.

The OS81210 supports the MediaLB protocol with a single-ended MediaLB 3-pin interface. This interface supports the exchange of network data types, including: control messages, asynchronous packets, synchronous data, DFI phase, QoS packets, and AVP packets. The transmitting Device sends data on a dedicated data line, while information about the data is simultaneously transported on an independent signal information line.

When using the MediaLB 3-pin interface, the OS81210 MediaLB Port output clock is configurable as 256×Fs (12.288 MHz at 48 kHz), 512×Fs (24.576 MHz at 48 kHz), or 1024×Fs (49.152 MHz at 48 kHz). When configured for 256×Fs, the OS81210 can transmit and receive up to 28 bytes (7 quadlets) per frame of data. When configured for 1024×Fs, the OS81210 can transmit and receive up to 124 bytes (31 quadlets) per frame of data.

MediaLB is a token-passing bus, where the Controller manages the token and generates the data clock. The Controller passes the token by sending out a *ChannelAddress* on the signal information line. The *ChannelAddresses* are pre-assigned by the board integrator and are associated with a specific transmitting Device and receiving Device. Once the transmitting Device associated with the *ChannelAddress* is granted bus access, it can send one quadlet of data on the MediaLB data line after a defined time delay. The Device provides information about the data being transmitted by simultaneously sending out a command on the signal information line. After receiving the *ChannelAddress*, the receiving Device must decide whether to receive the associated message or reject it. A status byte from the receiving Device is placed on the signal information line, one byte after the command from the sending Device is received. Once per network frame, the Controller generates a unique pattern (*FRAMESYNC*) on the signal information line. This defines the MediaLB frame edge, as well as the byte boundary of signal and data lines for all Devices.

Since MediaLB is a high-speed bus, connections to external devices should be implemented in hardware and not *bit-banged* on generic ports. A MediaLB Device interface, in the form of VHDL code, is available from Microchip. For more information about the Media Local Bus, refer to the MediaLB Specification [5].

3.3.3 STREAMING PORT

The Streaming Port provides a gateway for synchronous data exchange between the OS81210 and external legacy devices. The OS81210 Streaming Port supports four serial data pins sharing a common synchronization signal and bit clock. The data formats supported by the Streaming Port are compatible with industry-standard serial interfaces found on many ADCs, DACs and other devices.

3.3.4 RMCK PORT

The RMCK Port is a hardware interface controlled by the internal Clock Manager. This port provides a single clock output (**RMCK**) that is synchronous to the network frame rate (Fs). The RMCK Port is typically used in conjunction with the Streaming Port to exchange synchronous data over the network.The **RMCK** output is used to synchronize external application devices (e.g. ADC or DAC) to the common network timebase.

3.3.5 USB PORT

The USB Port allows connection to the OS81210 (USB Device) by an EHC (USB Host Controller) and supports on-PCB upstream USB 2.0 high-speed bulk transfers using either a standard USB 2.0 physical interface or a *High-Speed Inter-Chip* (HSIC) physical interface at a rate of 480 Mbit/s. The USB Port provides access to the network via an interface commonly found in consumer applications. This is especially useful for microcontrollers that do not support the typical interfaces found in an automotive application (e.g. MediaLB).

3.3.6 SPI PORT

The *Serial Peripheral Interface* (SPI) Port supports the transmission of asynchronous and control packets over an interface that is compatible with many microprocessors, data converters and other devices. When the SPI Port is enabled, it operates as an SPI bus slave.

3.3.7 I^{2}C PORT

Using the default *INIC Configuration String*, the I^{2}C Port operates as a bus slave and supports control message (such as ICMs, RCMs, and MCMs) exchange. An interrupt pin ($\overline{\textbf{INT}}$) is used to notify the EHC when the I^{2}C Port requires service (e.g. control message received). While the EHC can communicate with the OS81210 over the I^{2}C Port alone, the I^{2}C Port data rate limits the throughput on the Application Layer. Application performance can be maximized by using either the MediaLB Port, SPI Port, or USB Port which also supports the exchange of control messages.

The OS81210 I^{2}C Port may be configured as an I^{2}C master by changing the settings in the *INIC Configuration String*. In master mode, the OS81210 performs read and write operations with on-board I^{2}C slave devices based on remote commands received from the network.

3.3.8 GPIO PORT

Depending on the *INIC Configuration String* settings and firmware loaded in OS81210, the *General Purpose Input/Output* (GPIO) Port may be available to support various external operations. Some of the OS81210 hardware ports may not be available when the GPIO Port is used.

3.3.9 PORT EXPANSION

Devices belonging to the Microchip *I/O Companion* (IOC) family can be used as companions to INIC to provide additional features and hardware ports to the application. When an IOC is used as a companion to the OS81210 INICs, the MediaLB bus is typically the medium used to exchange data between the two devices. Data exchanged over MediaLB is routed to IOC hardware interfaces on which external devices may reside (e.g. ADC, DSP, video displays, etc.).

The OS85654 IOC, as depicted in Figure 3-3, is a high-performance routing engine that supports I/O expansion and provides a *Digital Transmission Content Protection* (DTCP) coprocessor. The primary function of the OS85654 is the routing of data streams between various hardware ports, while the integrated DTCP coprocessor provides optional encryption/decryption of data streams routed through the IOC device. Various services are available to support full *Authentication and Key Exchange* (AKE) implementation. For applications that do not require DTCP, the OS85656 IOC provides the same functionality as the OS85654, albeit without the DTCP coprocessor.

Applications supporting video benefit from the OS85621 IOC with its integrated low-latency H.264 CODEC, industry standard video I/O part, and DTCP coprocessor. See Figure 3-4. Applications not requiring content protection may instead utilize the OS85623, which provides the same functionality as the OS85621 but does not include the DTCP coprocessor.

Contact Microchip for more information about the I/O Companion family of devices.

FIGURE 3-3: INIC-IOC DTCP APPLICATION EXAMPLE

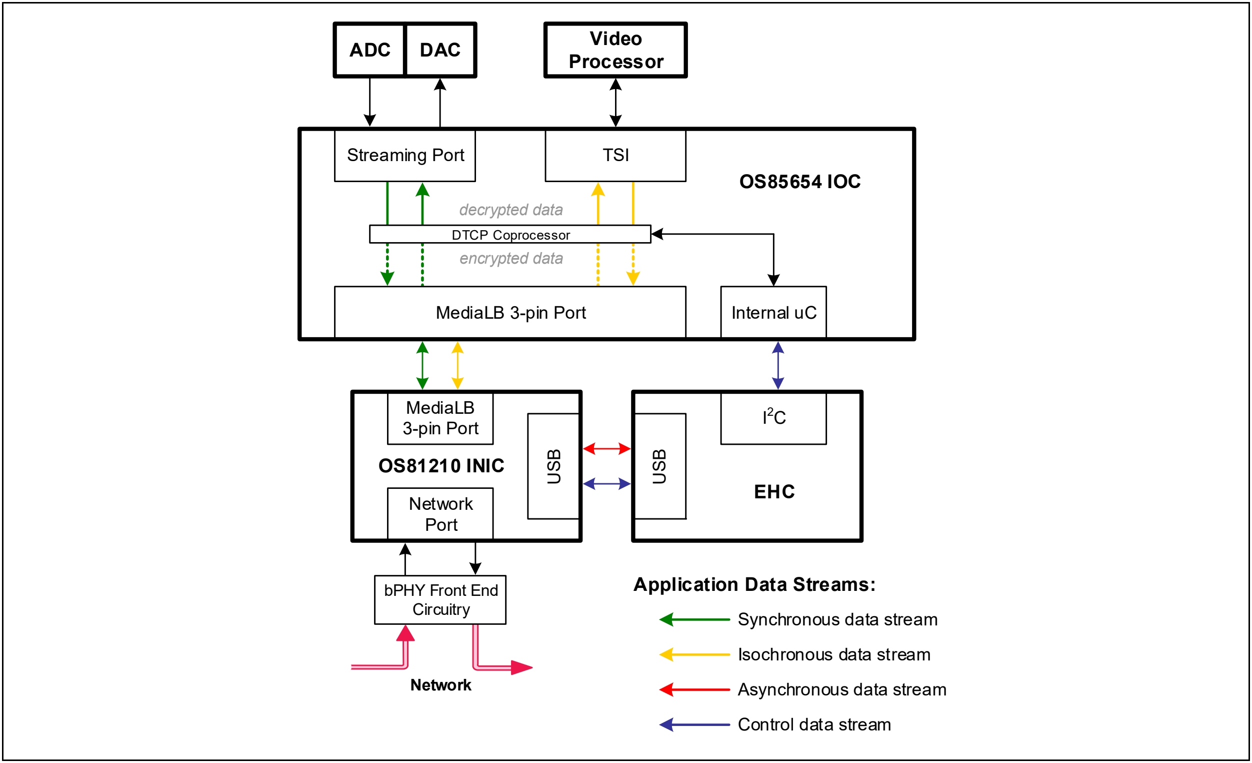
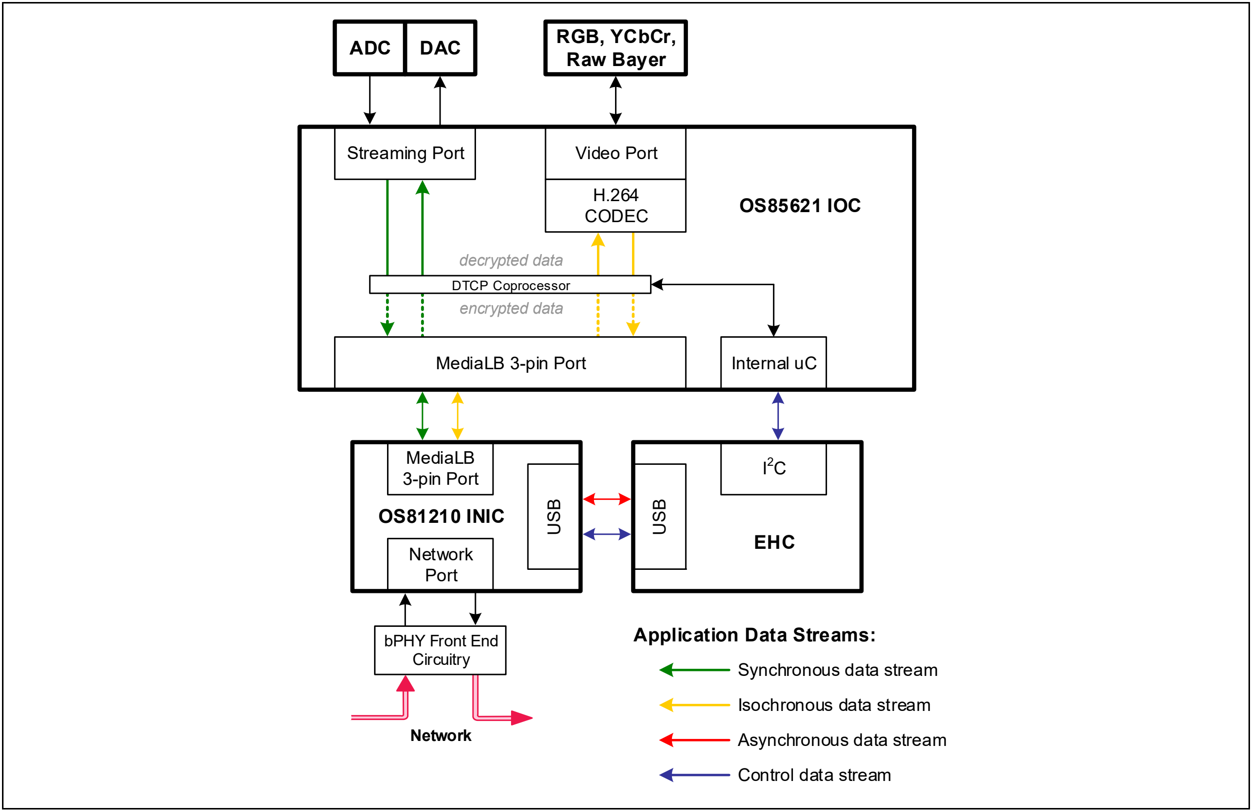


FIGURE 3-4: INIC-IOC VIDEO APPLICATION EXAMPLE



3.4 Centralized Network Diagnostics and Fallback Operation

The OS81210 network interface includes an embedded feature allowing the data path to reverse the direction of communication (receive on the **BTXP/BTXN** pins and transmit on the **BRXP/BRXN** pin). This reversal of direction enables two special modes of operation: isolation of cable faults if the normal ring is broken, and a special simplex mode of operation called Fallback Operation that can be used for fault-tolerant network operation.

3.4.1 CENTRALIZED NETWORK DIAGNOSTICS

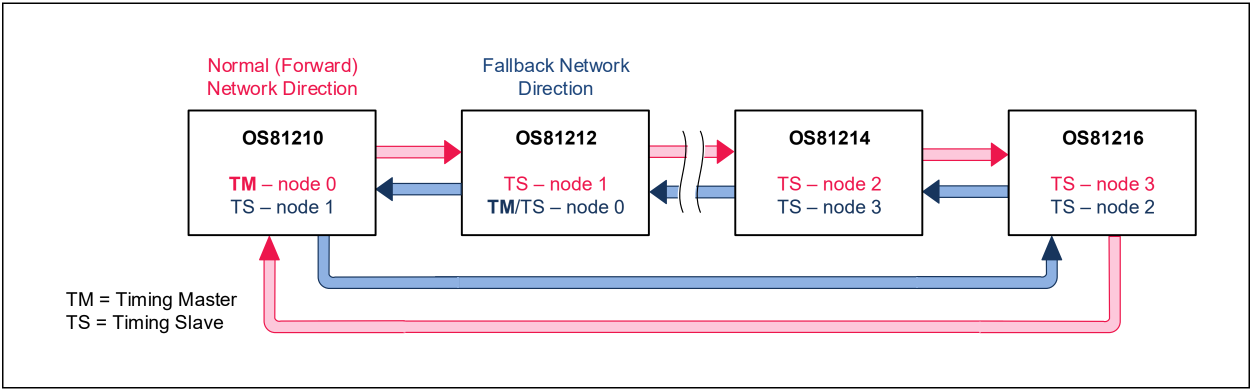
Centralized Network Diagnostics are used to collect diagnostic information from nodes in the network such as cable link connections in case normal operation is not possible due to a break in the ring architecture. These diagnostics are executed by the Timing Master (*Root Node*) over a control channel therefore requiring no additional cable wiring such as an electrical control line. All other nodes are *Remote Nodes* which do not require any additional hardware (EHC) or logic. For more information on triggering Central Network Diagnostics, please refer to the INIC Interface Specification [4].

3.4.2 FALLBACK OPERATION

The Fallback Operation mode allows a segment of the network to operate in the reverse direction even if the ring is damaged. This feature can be used to implement an emergency call (eCall) feature (e.g. react to a network physical interruption) or to implement a simplex daisy chain network topology for simple microphone applications. This mode is intended for applications required to sustain synchronous streaming data and limited control communication during a fault state. In this mode, the direction of communication is reversed and a new node will negotiate to become the Timing Master. Nodes that are required to operate in this fault-tolerant mode must enable Fallback Operation in the *INIC Configuration String.* If enabled, further parameters must be set allowing nodes permission to become the new timing master or a timing slave, or react only as a timing slave during Fallback Operation. Additionally, these nodes can be configured to automatically stream synchronous data on pre-defined channels when Fallback Operation begins. This enables the transfer of streaming data with no application firmware required.

A normal forward network direction example is shown in Figure 3-5. If the network is broken between nodes 1 and 2, Fallback Operation can be initiated and ultimately the furthermost node upstream of the break allowed to become a Timing Master will take over (Forward network node 1 would become Fallback node 0). For more information on Fallback Operation, please see the INIC Interface Specification [4].

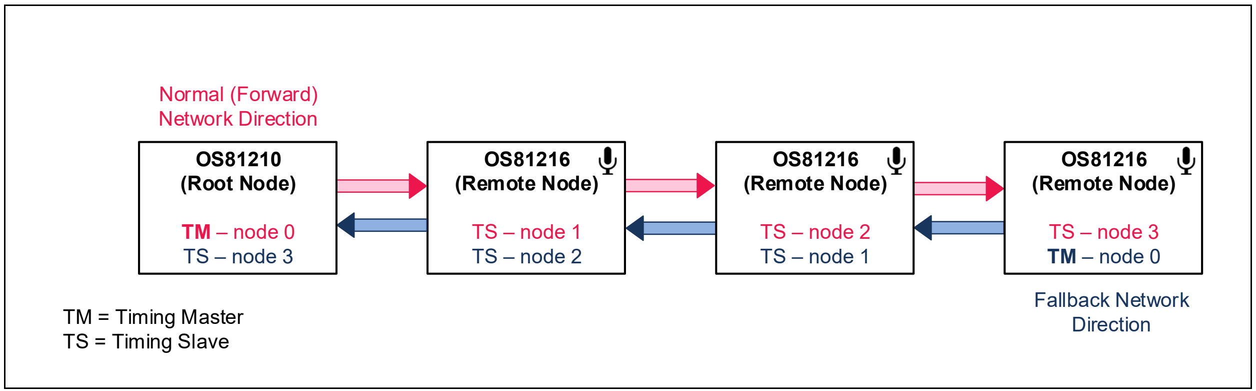
FIGURE 3-5: eCall EXAMPLE



Fallback Operation can also be used to implement a limited simplex daisy chain network which can automatically stream synchronous data in one direction. This feature is useful to create a simple microphone network. Only streaming data can be transmitted during the Fallback Operation mode. Each node must be statically configured via the *INIC Configuration String* to operate in Fallback Operation and use a predetermined channel to transmit streaming data. In Figure 3-6, the *Root Node* (forward network direction timing master node 0) will start the network and begin the negotiation phase. The network direction is reversed, and the previous forward direction timing slave node 3 becomes the timing master node 0 in Fallback Operation. Once the negotiation timer as expired, streaming data will begin transmitting over each microphone's channel in the Fallback Network direction to the *Root Node*. This limited simplex daisy chain mode enables the creation of a completely statically defined network that operates without further configuration following a single startup command.

For more information on Fallback Operation, please see the INIC Interface Specification [4].

FIGURE 3-6: SIMPLEX DAISY CHAIN MICROPHONE NETWORK EXAMPLE



3.5 MOST NetServices

In addition to a generic automotive network, the OS81210 can be used to implement a MOST network. To accelerate development of MOST network applications that use the OS81210, Microchip offers the MOST NetServices API, which provides a seamless software interface between INIC and the EHC. The division of services between the OS81210 INIC and the EHC in a MOST network is illustrated in Figure 3-1. A software library is available to provide all services that are relevant for:

- exchanging application data on a MOST network,
- managing hardware port data connections (e.g. sockets),
- managing high-level system tasks, and
- using control messages to access INIC for specific operations.

The MOST NetServices code incorporated into the EHC is divided into two distinct components: *Basic Services* and *Applications Socket*.

The *Basic Services* component includes message management, configuration management, and state control and supervision. It provides the facilities that enable direct communication with INIC through an interface to a low-level driver that is independent of the hardware port used. Formatted messages are sent and received by the low-level driver across the interface between the EHC and INIC.

The *Applications Socket* component resides on top of *Basic Services* and provides the functions required to implement the full FBlock `NetBlock` on the node. This component also contains a command interpreter, which provides a simple API for developing application FBlocks (e.g. FBlock `AudioAmp`).

MOST NetServices code is modular, allowing it to be customized for a particular application. Implemented in ANSI C, the MOST NetServices API can be adapted for individual requirements through configuration files.

With respect to the ISO communications model (shown in Figure 3-1):

- The OS81210 is connected to the bPHY *balanced media physical layer* with external passive front-end components.
- The OS81210 supports the *Data Link Layer* up to a portion of the *Session Layer*.
- The EHC must provide the remaining portion of the *Session Layer*, the *Presentation Layer*, and the *Application Layer*. When the MOST NetServices code is integrated:
  - the *Basic Services* component provides the remaining portion of the *Session Layer*, and
  - the *Applications Socket* component provides the *Presentation Layer* and a portion of the *Application Layer*.

The INIC Interface Specification [4] defines the FBlocks (and all associated functions) that are supported by the OS81210.

3.6 Unified Centralized Network Stack (UNICENS)

The *Unified Centralized Network Stack* (UNICENS) is an alternative to NetServices simplifying the handling of a complex, heterogeneous infotainment network. While NetServices is required to be implemented and running on every node of the system, UNICENS only needs to be implemented on a single *Root Node*. UNICENS provides centralized control and management for the entire network.

The network may be either statically or dynamically configured by UNICENS. In a static configuration, UNICENS scans the network on startup to discover the nodes, remotely configuring and controlling each node via *Control Frames*. A Remote Host Controller (RHC) may be implemented at a *Remote Node*, but is not required thereby significantly reducing hardware cost. This allows the network software to be centralized into the *Root Node* EHC, reducing system complexity and development cycles. UNICENS controls hardware at the *Remote Node*s through remote I^{2}C messages and GPIO control.

The OS81210 is ideal for implementation in the *Root Node* with its integrated USB Port. The OS81212, OS81214, and OS81216 are optimized for use in *Remote Nodes* with or without a Remote Host Controller (RHC).

More information regarding UNICENS may be obtained upon request from Microchip.

3.7 Operating System Drivers

Drivers are available which aid integration of the INIC with operating systems being used in many complex systems. For example, Microchip has made an INIC device driver available within the Linux®[1] mainline kernel.

The Linux driver for INIC implements each of the network data types presenting them to the application in standard programming interfaces. Synchronous streaming channels are implemented as Advanced Linux Sound Architecture (ALSA) hardware and appear as standard audio sources and sinks. A/V Packetized data channels are accessed using the Video4Linux2 (V4L2) application framework. The network Ethernet channel appears as a standard Ethernet interface. Control channels are accessed as a standard character device. The standard presentation of network data channels greatly simplifies application programming effort.

Contact Microchip for more information regarding the INIC driver, including the Linux kernel driver for INIC.

1.Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

4.0 INIC PROCESSOR

The OS81210 is a valid network node even without an attached EHC, due to the on-chip INIC Processor. The *INIC Software Stack*, running on the INIC Processor, provides a message-based interface that is easily implemented in a high-level language. This message-based interface not only simplifies the interface between the application and the network, but allows the EHC software to be completely driven by message events. Using the INIC API, as described in the INIC Interface Specification [4], the EHC has access to the same functionality that would be available on a register-based interface, but the programmer is spared the effort of making software align with the chip architecture. Much of the INIC application interface is common across INIC chips, minimizing the effort of upgrading the software for different application platforms. The power-up default settings of the INIC Processor, such as timing-master/timing-slave functionality and other configuration options, can be changed through the *Customer Configuration Interface* (see Section 16.6 "Configuration and Debug").

Partitioning the node architecture between INIC and the EHC results in an efficient implementation, allowing INIC to respond to many network events without intervention by the EHC. Network initialization is optimized, not requiring the network transceiver at each node to wait for the Application Layer to respond. Application software runs efficiently since the number of interrupts to the EHC is minimized. Interaction between the EHC and INIC can be further reduced if the EHC enables notification for INIC properties.

4.1 Data Exchange

General information on data exchange with the OS81210 is given below. Refer to the INIC Interface Specification [4] for more detailed information. The INIC Message Interface consists of two virtual paths - the Configuration Interface which handles message to control INIC itself and the Application Interface which is used to pass application specific messages between nodes.

4.1.1 CONFIGURATION DATA

The OS81210 Configuration Interface is used for *message-based* data exchange between INIC and the EHC.

These messages transferred between the local INIC and the EHC via the MediaLB Port, SPI Port, USB Port, or I^2C Port are called *Application Control Frames*. *Application Control Frames* can be routed to the local INIC for control and configuration, or through the INIC and across the network for controlling and configuring remote INIC nodes.

Application Control Frames exchanged between the local EHC and the internal *INIC Software Stack* are called *INIC Control Messages* (ICMs). *Remote Control Messages* (RCMs) are similar to ICMs with the exception that they are routed through the local INIC to remote INIC nodes for configuration and network management purposes.

All *Application Control Frames* transferred up or down the *INIC Software Stack* are exchanged using the second generation Port Message Protocol (PMPv2). The message format (defined as the *Port Message*) is the same, regardless of the physical hardware port used. For more information on PMPv2, refer to the Port Message Protocol Version 2 User's Guide [6].

All control messages require a status message response from the receiving device. This status message indicates the OS81210 received the message successfully. Similarly, when the OS81210 sends a control message to the EHC (from the network or as a response to a previous request), the EHC is required to respond with a status message indicating message reception was successful. INIC provides several mechanisms (e.g. idle, buffer timer, watchdog, etc.) to monitor the *Port Message* transfer during a predefined time frame. Refer to the INIC Interface Specification [4] for more information on configuring these mechanisms.

4.1.2 APPLICATION DATA

The INIC supports multiple data formats and channels to send messages between applications on different nodes (e.g. a volume command between a head unit and amp module). *Ethernet Data Frames* can be used over the packet channel to transmit high bandwidth data between nodes using standard Ethernet formats such as TCP/IP or UDP. This allows the use of standardized protocols such as SOME/IP or MQTT to be used at the application level. The Ethernet frames are transferred into INIC via the MediaLB, USB, or SPI Port.

If the high bandwidth or standardized Ethernet protocols provided by the packet channel is not needed, it is possible to use the Application Interface to send user defined application specific control messages between nodes via the I^2C, MediaLB, USB, or SPI Port. These messages transported over the INIC control channel are called *Meta Control Messages* (MCMs).

Quality of Service (QoS) packets are handled like real-time data on a synchronous streaming network channel. QoS packet headers are ignored by the INIC and all channel data is directly routed in a continuous fashion to the appropriate INIC hardware port, bypassing the INIC Message Interface.

4.1.3 INTER-PROCESSOR COMMUNICATION (IPC)

An Inter-Processor Communication (IPC) connection provides a point-to-point packet connection between MediaLB and USB. Packets can be transferred directly between MediaLB and USB without going through the Network Port. For more information on IPC packets please refer to the INIC Interface Specification [4].

5.0 I²C PORT

The I²C Port supports the I²C protocol and is processed in bytes by the internal firmware of the OS81210. The **SCL** pin clocks data in and out; **SDA** is a bi-directional data pin in which data is transferred MSB-first. When the OS81210 I²C Port is configured as an I²C slave device, it must be managed externally. The I²C Port is configured as an I²C slave in the default *INIC Configuration String*, however, the power-up defaults can be changed through the *Customer Configuration Interface* (see Section 16.6 "Configuration and Debug"). If the I²C Port is not configured as an I²C master by default at power-up (or reset), it can be also be configured as an I²C master through the INIC API.

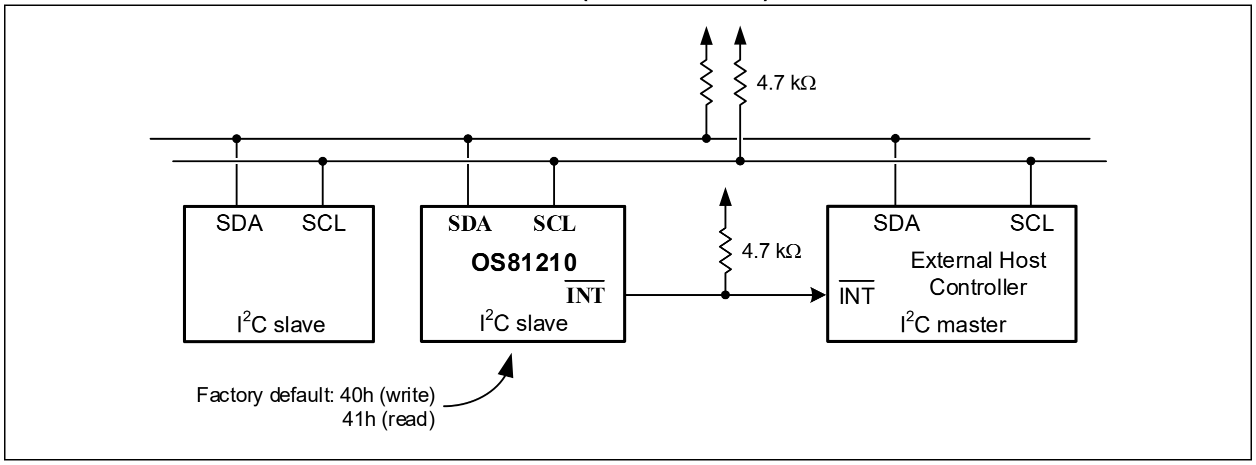
Note: When used in I²C master mode, the I²C Port is unavailable for use as an I²C slave device and cannot be used for Port Message exchange with an EHC.

5.1 Slave Operation

As a slave, the I²C Port supports communication of control messages between the OS81210 and the EHC. The control messages may be used to configure and control INIC, target local INIC FBlocks (ICMs), or be routed through the INIC between the EHC and the network control channel (RCMs and MCMs). The control messages are encapsulated within *Port Message*s that are transferred over the I²C bus. The first two bytes of the *Port Message* contains the *Port Message Length* (PML) which is the number of bytes remaining in the message. Refer to the INIC Interface Specification [4] for more detailed information.

Figure 5-1 illustrates the OS81210 as a slave in an I²C environment.

FIGURE 5-1: I²C PORT PIN CONNECTIONS (SLAVE MODE)



When the I²C Port is enabled as a slave, the $\overline{\text{INT}}$ pin is used by the INIC to signal the communication status to the EHC I²C bus master, eliminating the need for a polling by the EHC. The $\overline{\text{INT}}$ pin is an open-drain, active low output that can be wire-OR'ed with the interrupt outputs of other devices into the EHC's interrupt input. INIC drives $\overline{\text{INT}}$ low to inform the EHC when service through the port is required (e.g. message available for reading). Following reset, the EHC should wait for the first falling edge of $\overline{\text{INT}}$, which occurs when initialization is complete. The first *Port Message* available for reading through the I²C Port indicates INIC is ready to receive messages and the EHC should initiate synchronization of the OS81210 as described in the INIC Interface Specification [4].

The first byte of an I²C transaction is the bus address plus a read/write bit (R/$\overline{\text{W}}$) that determines whether the EHC is reading or writing *Port Messages*. The OS81210 factory default I²C address is 41h for read, and 40h for write; however, using the *Customer Configuration Interface* (Section 16.6 "Configuration and Debug"), the default I²C address can be changed in the *INIC Configuration String*.

The INIC processing of I²C transactions varies based on the overall loading of internal tasks. Clock stretching is a handshaking mechanism implemented in the I²C Port as it provides an appropriate mechanism to adapt the data transfer rate dynamically, thereby maximizing the transfer rate at any given time. If the OS81210 cannot keep up with a message, it stretches the clock following the eighth data bit or the acknowledge bit; therefore, the I²C bus master must monitor **SCL** when communicating with INIC. If the OS81210 is busy supporting higher priority tasks, it will NACK the I²C address, indicating that the master should retry the message at a later time. I²C NACK responses occur at the address byte boundary (e.g. following the R/$\overline{\text{W}}$ bit).

The I^2C Port does not support repeated START conditions in slave mode. Therefore, the master must terminate each transaction with a STOP condition before the next transaction can be started.

For more information on the I^2C protocol, refer to the I2C-Bus Specification [7].

5.1.1 WRITING TO THE I^2C PORT

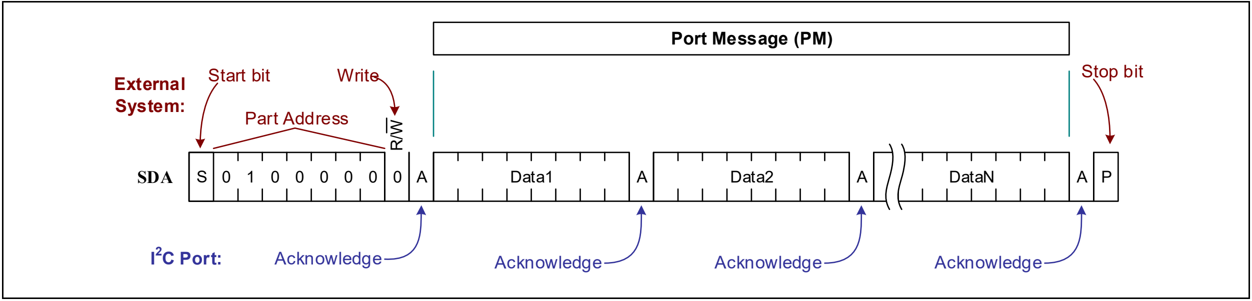
The transmission of a *Port Message* to the I^2C Port is accomplished by performing an I^2C write transaction. The beginning of a write transaction is marked by a START condition. The first byte transmitted specifies the chip address, as well as whether the transaction is a read or a write. The OS81210 address (factory default of 0100000b) occupies the upper seven bits of the first byte; the $R/\overline{W}$ bit is the LSB.

When the I^2C Port receives an address byte of 40h (factory default OS81210 address and $R/\overline{W}$ bit clear), it acknowledges reception of the byte through an acknowledge bit and the *Port Message* is then written into the I^2C Port. If the I^2C Port does not acknowledge the address byte, the master should halt the write transaction and retry the *Port Message* later. The *Port Message* format is described in the INIC Interface Specification [4].

Sending messages to the INIC does not involve the $\overline{\text{INT}}$ pin.

Figure 5-2 illustrates the I^2C write transaction. The characters “S” and “P” represent the start and stop conditions for messages.

FIGURE 5-2: I^2C PORT WRITE TRANSACTION (SLAVE MODE)



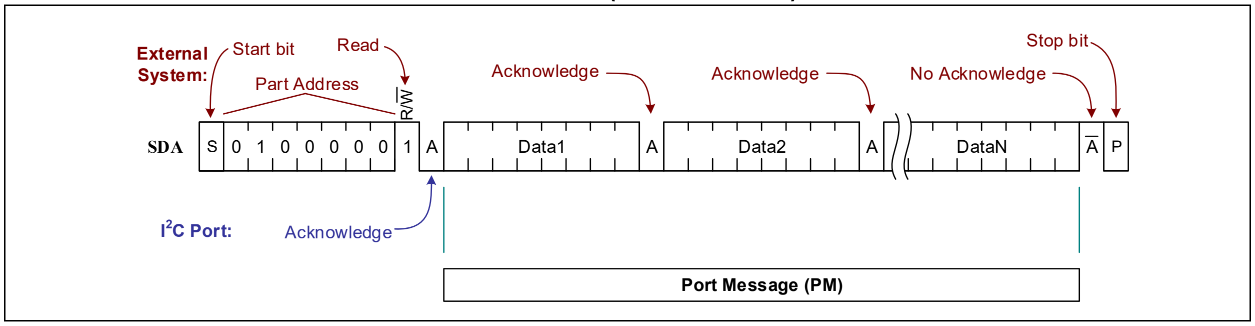
5.1.2 READING FROM THE I^2C PORT

Reception of a *Port Message* from the I^2C Port is accomplished by performing an I^2C read transaction. Each read transaction is marked by a START condition. The first byte specifies the chip address, as well as whether the transaction is a read or a write. The OS81210 address (factory default of 0100000b) occupies the upper seven bits of the first byte; the $R/\overline{W}$ bit is the LSB.

When the I^2C Port receives the address byte of 41h (factory default OS81210 address and $R/\overline{W}$ bit set), it acknowledges reception of the byte through an acknowledge bit and the *Port Message* is read from the I^2C Port. If the I^2C Port does not acknowledge the address byte, the master should halt the read transaction and retry the transaction later. The *Port Message* format is described in the INIC Interface Specification [4]

Figure 5-3 illustrates the I^2C transmission read transaction. The characters “S” and “P” represent the start and stop conditions for messages.

FIGURE 5-3: I^2C PORT READ TRANSACTION (SLAVE MODE)



The $\overline{\text{INT}}$ pin being asserted low indicates that at least one *Port Message* is available for reading. Once a message starts being read, the $\overline{\text{INT}}$ pin is de-asserted. If additional messages are pending (waiting to be read), the $\overline{\text{INT}}$ pin is asserted low again. This pending interrupt can occur as soon as the last byte of the current message is read.

Message availability can also be polled by the EHC without using the $\overline{\text{INT}}$ pin interrupt mechanism. The *Port Message Length* (PML) value inside the first two bytes of the *Port Message* indicates if a message is pending. The EHC initiates a read transaction and reads the PML value. If the read PML value is zero, no message is available to be read from the INIC. However, such a polling mechanism will keep the bus busy, hence preventing other transactions from being processed.

In each read transaction, only one message can be fetched from the I^{2}C Port. Only after the entire message is read will the INIC remove the message from its queue. If a message is not fully read (due to a STOP or repeated START condition received before the end of the message), $\overline{\text{INT}}$ is reasserted low and the next read command starts reading from the beginning of the disrupted message, not the position where the read was previously terminated. In addition, if more bytes are read than the message length specified by the *Port Message Length* field or if no messages are queues, the INIC will return zero (00h) values in the read result.

Ideally, the EHC reads the entire length of the *Port Message* in a single I^{2}C Port read transaction. To accomplish this, the EHC I^{2}C driver must be capable of pausing the transaction after reading the PML value. During this time, the EHC can allocate the necessary memory and proceed with reading the remaining bytes of the message. For EHC I^{2}C drivers that are not able to decipher the message length during a single read transaction, other methods of reading the *Port Message* are feasible, including the following:

- Example 1: The EHC can issue two consecutive I^{2}C Port read transactions. The first transaction reads the length field of the *Port Message*. Based on the *Port Message Length* obtained in the first read transaction, a second transaction reads the entire *Port Message*.
- Example 2: The EHC can always read the maximum length of a *Port Message* in a single read transaction, regardless of the actual message length. Once the *Port Message* is read, application software can later determine the relevant portion of the read data. The INIC returns 00h data when a read transaction extends beyond the actual length of the *Port Message*. However, this will result in very high bus traffic when reading the remaining 00h data following the end of the *Port Message*. This method is inefficient and therefore should only be chosen if the other methods cannot be used.

5.2 Master Operation

The OS81210 I^{2}C Port may be configured as a bus master to support applications requiring remote control of local I^{2}C slaves through the network. As an I^{2}C master, the OS81210 performs read and write operations with local I^{2}C slave devices based on INIC API commands.

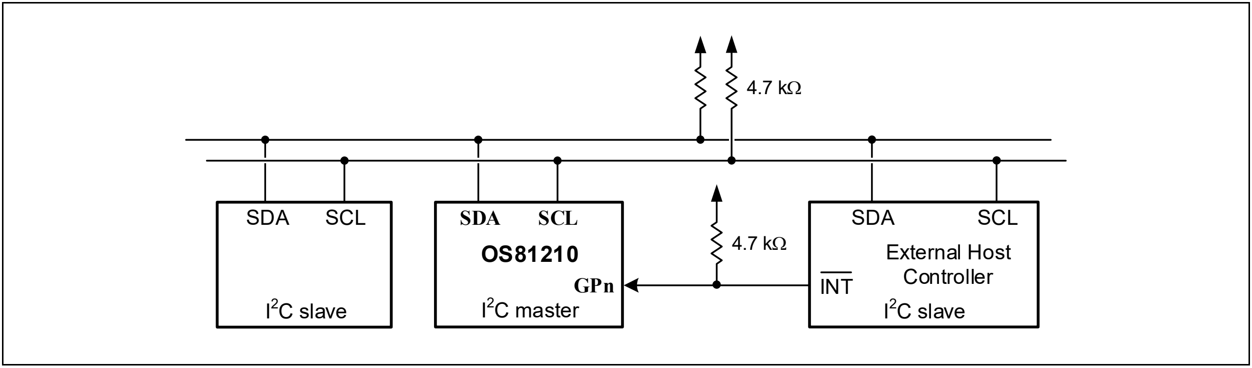
When operating as an I^{2}C bus master, two settings for I^{2}C transaction speed are available. The SCL clock frequency can be set to either 100 kHz (Standard-mode) or 400 kHz (Fast-mode). Faster speed modes as defined in the I2C-Bus Specification [7] (e.g. Fast-mode plus, High-speed mode) are not supported by the OS81210. The speed setting is selected when the port is created.

The OS81210 supports both the standard 7-bit device addressing and enhanced 10-bit I^{2}C addressing modes. Standard 7-bit device addressing is most commonly used by slaves. The 10-bit addressing mode expands the number of possible device addresses while maintaining backwards-compatibility with devices that only support 7-bit addressing. Per the I2C-Bus Specification [7], a '11110' following a START condition indicates the use of enhanced 10-bit addressing. For 10-bit addressing, the first byte contains the two most significant bits of the 10-bit device address as well as the R/$\overline{\text{W}}$ flag. The remaining eight bits of the address are sent in the second byte. The OS81210 supports full interoperability for systems with mixed 7-bit and 10-bit slave device addresses.

As described in the I2C-Bus Specification [7], multi-master configurations implement an arbitration mechanism to allow more than one master to control the bus at the same time without corrupting data exchange. The OS81210 supports multi-master I^{2}C configurations and implements the necessary bus-busy-state detection and bus arbitration logic to prevent data collision during read/write transfers inside a multi-master environment.

Figure 5-4 illustrates the OS81210 as a master in an I^{2}C environment.

FIGURE 5-4: I²C PORT PIN CONNECTIONS (MASTER MODE)



Clock stretching is a handshaking mechanism used by slave devices to adapt the data transfer rate dynamically. A slave device is permitted to hold SCL low after receiving (or sending) a bit to indicate it is not yet ready to process additional data. To ensure compatibility with all slave devices, the OS81210 master supports clock stretching at the bit and byte boundary.

Some I²C slave devices implement an interrupt pin to indicate to the I²C master that service is needed. The INIC GPIO Port can be configured and remotely monitored to support devices that implement interrupt functionality. Refer to Chapter 13.0 "General Purpose Input/Output (GPIO) Port" for more information.

Once master mode is enabled, remote control messages (*Control Frames*) are used by a remote node on the network to initiate I²C read and write transactions on the local bus. The read/write methods specify the transfer mode, the slave target address, and the number of bytes to be read/written. The maximum block size for a read/write transaction is 32 bytes of data. A successful read transaction returns the device address, the number of bytes read, and the list of data bytes. When a write transaction successfully finishes, the device address and number of bytes written is returned.

Refer to the INIC Interface Specification [4] for the function reference and more information on using the OS81210 I²C Port in master mode.

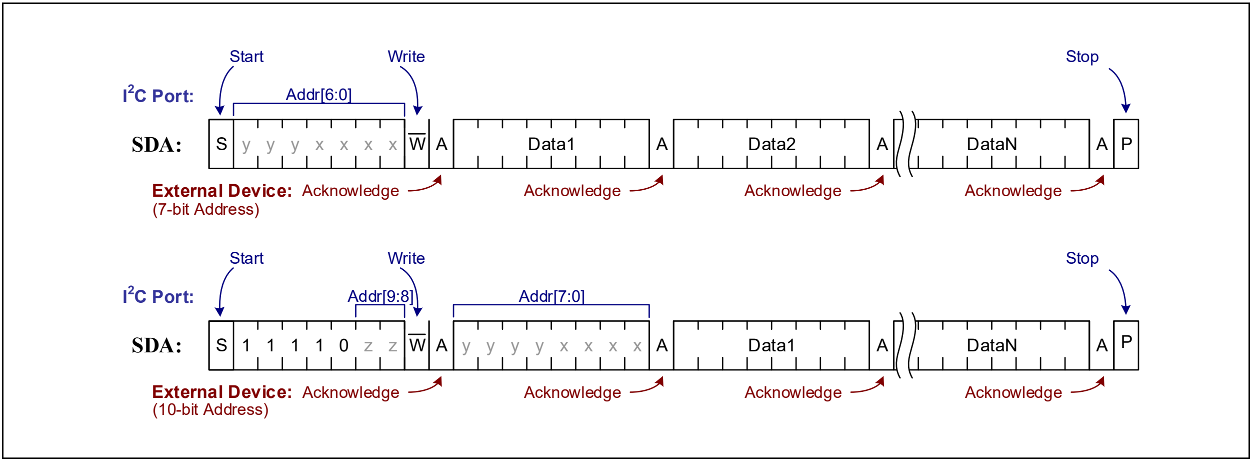
5.2.1 WRITING TO AN I²C SLAVE DEVICE

Per the I2C-Bus Specification [7] a '11110' following a START condition indicates use of the 10-bit addressing scheme. For a write operation, the first byte contains the two most significant bits of the 10-bit device address plus the R/$\overline{\text{W}}$ bit clear. Following acknowledgment from the external device, the remaining eight address bits are sent in the second byte. Message payload (*Data1* through *DataN*) begins in the third byte of the message for 10-bit addressing.

When using 10-bit addressing, the second address byte is actually sent as the first data byte. Therefore, if the message payload contains `n` data bytes, the total length of the write transaction must be set to `n+1`.

Figure 5-5 illustrates a write transaction for both 7-bit and 10-bit addressing modes.

FIGURE 5-5: I²C PORT WRITE TRANSACTION (MASTER MODE)

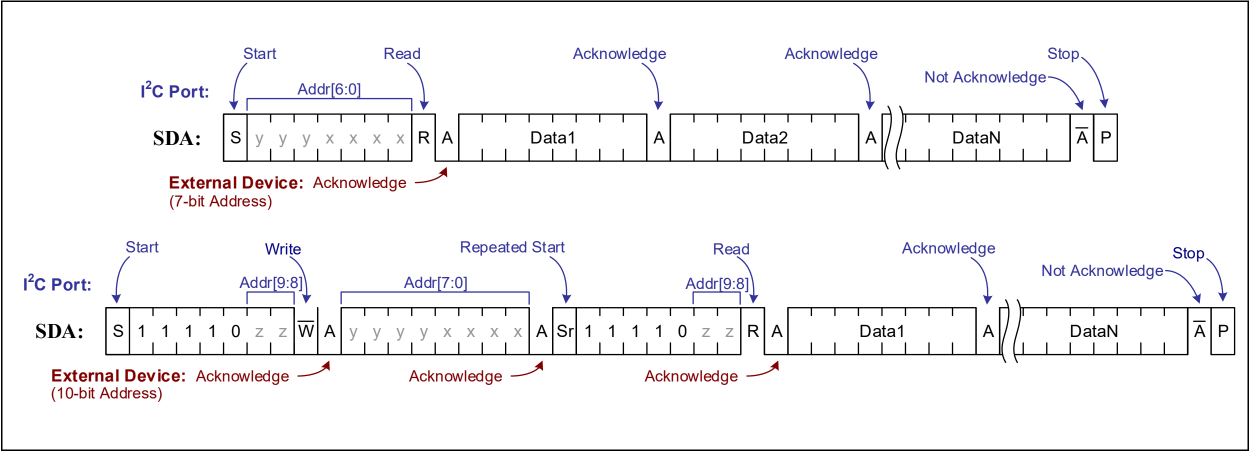


5.2.2 READING FROM AN I²C SLAVE DEVICE

Read transactions on slave devices with 10-bit addresses require a Repeated START condition. Refer to the I2C-Bus Specification [7] for more information. The master first fully addresses the 10-bit slave device with a write transaction (R/W̅ bit clear) and then issues a Repeated START condition. After the Repeated START, the direction is changed and the master completes the read transaction with an abbreviated version (only the two most significant bits) of the address and the R/W̅ bit set.

Figure 5-6 illustrates a read transaction for both 7-bit and 10-bit addressing modes.

FIGURE 5-6: I²C PORT READ TRANSACTION (MASTER MODE)

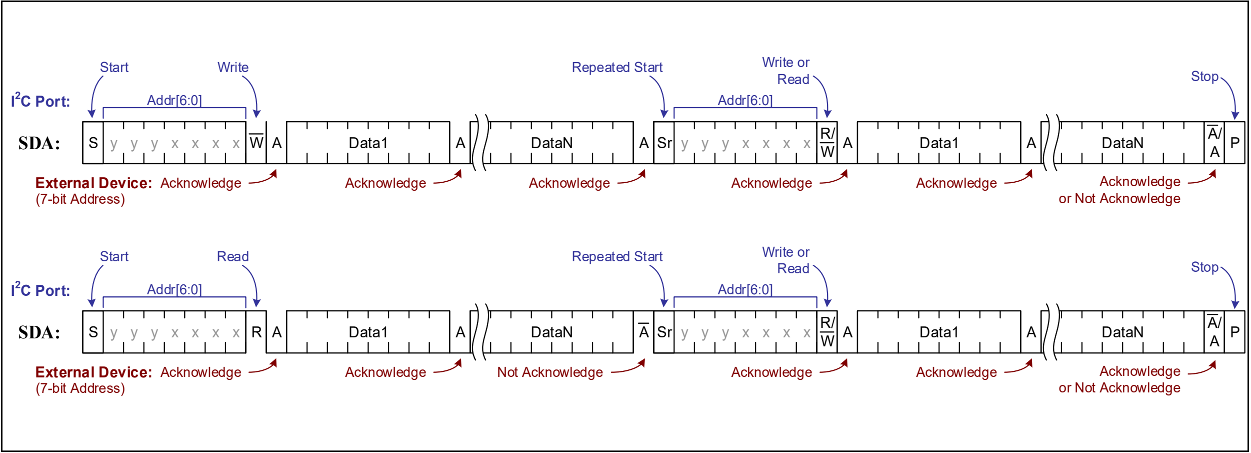


5.2.3 REPEATED STARTS ON AN I²C SLAVE DEVICE

A repeated START mode is available for both Read and Write transactions to a single I²C slave in both 7-bit and 10-bit addressing. The I²C master can issue any combination of subsequent writes or reads to an I²C slave using a repeated START condition in place of the previous transaction's STOP condition.

Figure 5-6 illustrates a repeated START following a write or a read transaction in 7-bit addressing mode.

FIGURE 5-7: I²C PORT REPEATED WRITES AND READS (MASTER MODE)


Start
Write
Repeated Start
Write or Read
Stop
I²C Port:
Addr[6:0]
Addr[6:0]
SDA:
S y y y x x x x W̅ A Data1 A DataN A Sr y y y x x x x R/W̅ A Data1 A DataN A̅/A P
External Device: Acknowledge
(7-bit Address)
Acknowledge
Acknowledge
Acknowledge
Acknowledge
Acknowledge or Not Acknowledge
Start
Read
Repeated Start
Write or Read
Stop
I²C Port:
Addr[6:0]
Addr[6:0]
SDA:
S y y y x x x x R A Data1 A DataN A̅ Sr y y y x x x x R/W̅ A Data1 A DataN A̅/A P
External Device: Acknowledge
(7-bit Address)
Acknowledge
Not Acknowledge
Acknowledge
Acknowledge
Acknowledge or Not Acknowledge

6.0 MediaLB PORT

A *Media Local Bus* (MediaLB) Port is available on the OS81210, supporting a MediaLB 3-pin interface. When enabled, the MediaLB Port provides the EHC access to network data types including: control messages, asynchronous packets, synchronous data, DFI phase, Quality of Service (QoS) IP streaming packets, and A/V Packetized (AVP) isochronous streaming packets.

The MediaLB Port supports communication of control messages between the OS81210 and the EHC. The control messages may be used to configure and control INIC (ICMs), target the local INIC FBlocks (ICMs), or be routed by INIC between the EHC and the network control channel (RCMs and MCMs). Refer to the INIC Interface Specification [4] for more detailed information.

The MediaLB Port also allows access to the internal functions of the OS81210, eliminating the need for I^{2}C Port access. With MediaLB, only one interface is needed for data and control. When combined with the MediaLB software API library, the system designer is insulated from much of the complexity of managing the interface between the application software and the network. When MediaLB is configured as the configuration or application interface, the first PMPv2 status message sent to the EHC after power-up/reset indicates that the OS81210 is initialized and ready to communicate with the EHC. In common applications, this message is used to instruct the EHC to synchronize and initialize communication. Additionally, the EHC may initiate synchronization of the OS81210, as defined in the INIC Interface Specification [4]. This ensures that OS81210 and the EHC can synchronize communication under various circumstances.

The MediaLB Port is disabled in the default *INIC Configuration String*, however, the power-up defaults can be changed through the *Customer Configuration Interface* (see Section 16.6 “Configuration and Debug”). If the MediaLB Port is not enabled by default at power-up (or reset), it can be enabled and configured through the local INIC FBlocks.

Data is transmitted onto the network most significant byte (MSB) first. To comply with the ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications [1] and the ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification [2], when transmitting multi-byte words, the most significant byte should be transmitted first. In addition, when transmitting stereo data, left should be sent before right.

6.1 MediaLB 3-pin Physical Layer

The MediaLB 3-pin physical layer is a single-ended bus consisting of:

- **MLBCLK** - MediaLB Controller output signal that clocks data in and out of the MediaLB Port
- **MLBSIG** - bi-directional signal that transports signal information to and from the MediaLB Port
- **MLBDAT** - bi-directional signal that transports data to and from the MediaLB Port

The MediaLB Controller (OS81210) uses **MLBSIG** to grant bus access to a MediaLB Device (e.g. EHC). That Device can then send data to other Devices (including the Controller). The OS81210 **MLBDAT** pin receives data for configuration or transmission onto the network, and transmits data to other Devices from the network receiver. Data and signal information is shifted into Devices by the MLBC bus clock, which is driven by the OS81210 **MLBCLK** pin. A complete description of the Media Local Bus can be found in the MediaLB Specification [5].

The MediaLB 3-pin connection diagrams are shown in Figure 6-1 and Figure 6-2. Each MediaLB line has a weak pull-down resistor to keep the bus in a known state when no device is driving the line. MediaLB devices also have a series resistor near the device for termination and rise/fall time control. The clock line may optionally have AC-parallel termination at the farthest point from the Controller (clock source) to further control rise and fall times. In a three node system, the routing to the device in the middle of the bus should be done in a pass-through manner so that stubs on the bus are eliminated. The MediaLB 3-pin bus should be routed as 50 Ω single-ended traces on a single layer over a continuous reference plane. As applications will vary, the MediaLB 3-pin bus should be simulated to fine-tune performance, minimize reflections, and optimize the series termination resistor values. Please see Section 16.9.2 “MediaLB Port” and the MediaLB Specification [5] for layout guidance.

FIGURE 6-1: MediaLB 3-PIN INTERFACE CONNECTION DIAGRAM 2-NODE SYSTEM

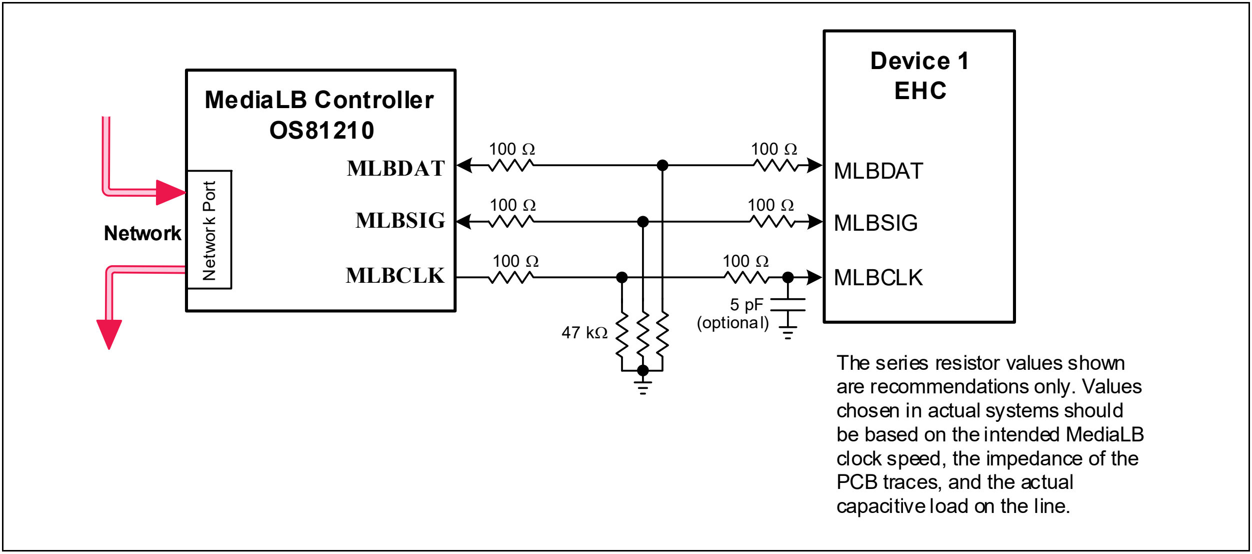
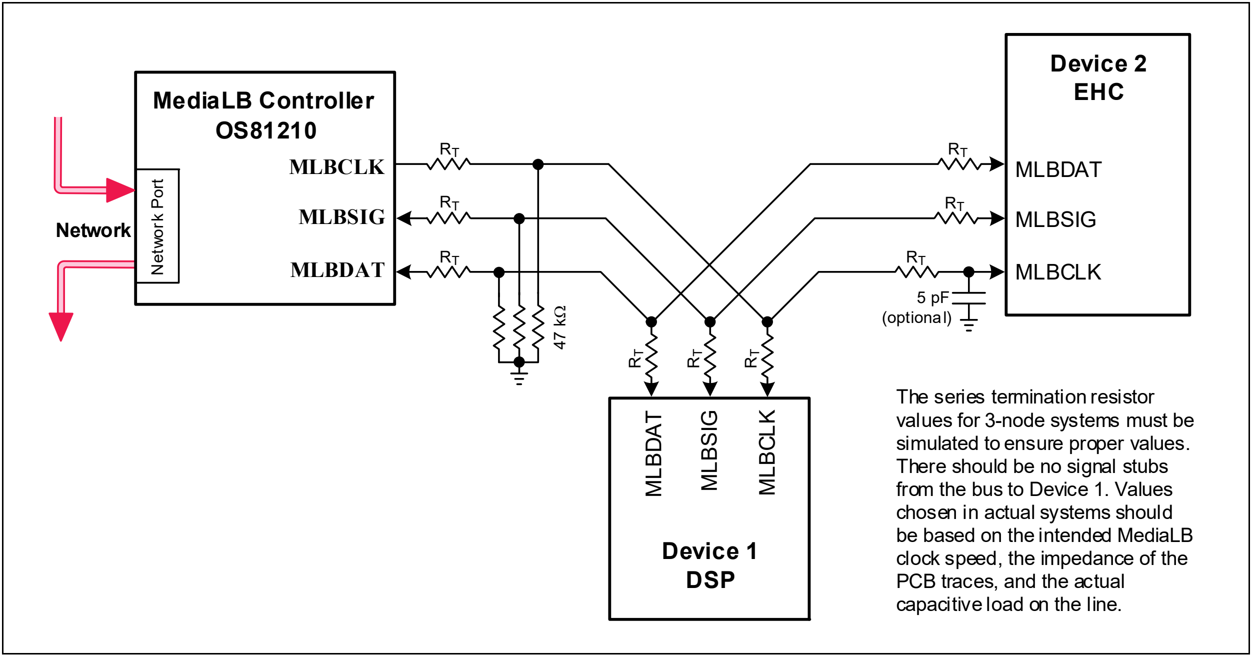

Device 1
EHC
MediaLB Controller
OS81210
Network Port
Network
MLBDAT
MLBSIG
MLBCLK
100 Ω
47 kΩ
5 pF
(optional)
The series resistor values shown are recommendations only. Values chosen in actual systems should be based on the intended MediaLB clock speed, the impedance of the PCB traces, and the actual capacitive load on the line.

FIGURE 6-2: MediaLB 3-PIN INTERFACE CONNECTION DIAGRAM 3-NODE SYSTEM


Device 2
EHC
MediaLB Controller
OS81210
Network Port
Network
MLBCLK
MLBSIG
MLBDAT
R_T
47 kΩ
5 pF
(optional)
Device 1
DSP
The series termination resistor values for 3-node systems must be simulated to ensure proper values. There should be no signal stubs from the bus to Device 1. Values chosen in actual systems should be based on the intended MediaLB clock speed, the impedance of the PCB traces, and the actual capacitive load on the line.

6.2 MediaLB 3-pin Data Link Layer

The MediaLB protocol supports a set of physical channels for sending data over the Media Local Bus. The physical channels (PCn) are four bytes in length (one quadlet) and can be grouped into logical channels. Each logical channel represents a unidirectional data path (one or more quadlets in length) and is defined by a unique *ChannelAddress*.

On each logical channel, the source (transmitter) is a unique Device, with a destination (receiver) of one or more Devices. In a system, the *ChannelAddresses* are generally static and defined at the development stage; however, the MediaLB Controller can dynamically define additional *ChannelAddresses* during operation to communicate with other Devices.

The MediaLB Controller initiates communication over the Media Local Bus by sending out a *ChannelAddress* on **MLBSIG**. This indicates to the addressed Device that it can begin transmitting on the logical channel associated with that *ChannelAddress*. The structure of the *ChannelAddress* is described in the MediaLB Specification [5], *Link Layer Section*.

The *ChannelAddress* range for the MediaLB 3-pin interface is 0x0002-0x003E; however, the number of *ChannelAddresses* supported is based on the MediaLB Port speed and the size of the logical channels associated with the *ChannelAddresses*. The total number of physical channels available at a given MediaLB Port 3-pin speed is indicated in Table 6-1. The mapping of specific physical channels into a logical channel is arbitrary and managed by the MediaLB Controller.

TABLE 6-1: AVAILABLE MediaLB BANDWIDTH

MediaLB 3-pin Speed	Quadlets per MediaLB Frame	Physical Channels	Available Physical Channels[1]	Bytes Available per Frame
256×Fs	8	PC0 - PC7	7	28
512×Fs	16	PC0 - PC15	15	60
1024×Fs[2]	32	PC0 - PC31	31	124

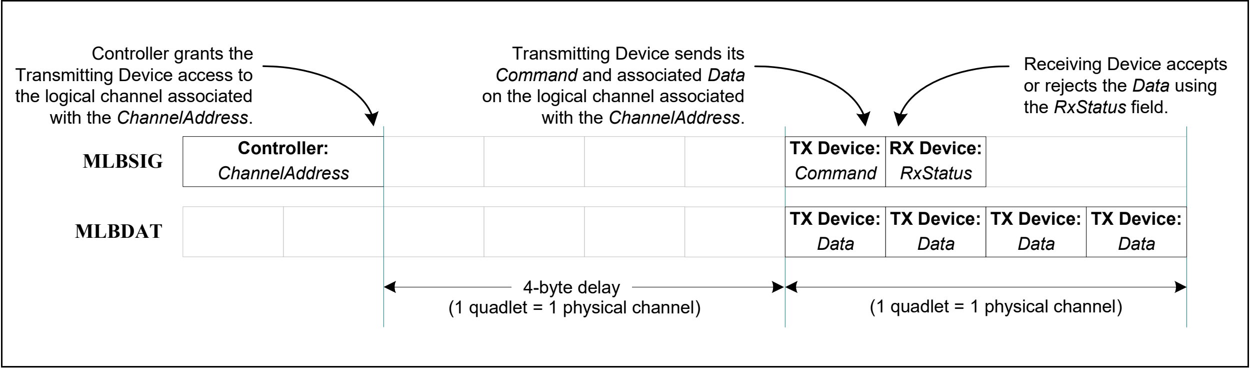
Note 1: PC0 (first physical channel of the MediaLB frame) is always used as the *SystemChannel*; therefore, the bytes of this physical channel are not available for application data.

2: The MediaLB 3-pin 1024×Fs speed should only be used if the bandwidth is required. Timing constraints at 1024×Fs limit the bus to a Controller and one closely spaced Device. The design should be thoroughly simulated to ensure feasibility and the layout must be optimized to ensure a successful implementation. Please consult the MediaLB Specification [5] and contact Microchip support if 1024×Fs is to be used.

Once per network frame, the MediaLB Controller generates a unique 16-bit *FRAMESYNC* pattern on the signal information line (**MLBSIG**). The end of the *FRAMESYNC* pattern defines byte and physical channel boundaries, and indicates that the MediaLB frame boundary occurs one quadlet later. Physical Channel 0 (PC0), defined as the first physical channel of the MediaLB frame, is always used as the *SystemChannel*. The *FRAMESYNC* pattern represents a special *ChannelAddress*, which grants the MediaLB Controller PC0 for system administration.

As illustrated in Figure 6-3, one quadlet after the *ChannelAddress* is sent by the Controller, the Device associated with that *ChannelAddress* sends out a MediaLB *Command* on **MLBSIG**, while simultaneously sending the corresponding data on **MLBDAT**. All MediaLB Devices, including the OS81210 Controller, compare the *ChannelAddress* to their internal Channel Address table to determine whether they are the intended receiver.

FIGURE 6-3: MediaLB 3-PIN DATA STRUCTURE


Controller grants the Transmitting Device access to the logical channel associated with the ChannelAddress.
Transmitting Device sends its Command and associated Data on the logical channel associated with the ChannelAddress.
Receiving Device accepts or rejects the Data using the RxStatus field.
MLBSIG
MLBDAT
Controller: ChannelAddress
TX Device: Command
RX Device: RxStatus
TX Device: Data
TX Device: Data
TX Device: Data
TX Device: Data
4-byte delay (1 quadlet = 1 physical channel)
(1 quadlet = 1 physical channel)

When the OS81210 Controller is a receiving Device, it places the *RxStatus* response on **MLBSIG** one byte after the MediaLB *Command* is sent by the transmitting Device. To accept the data, an *RxStatus* response of *ReceiverReady* is sent. To reject the data, an *RxStatus* response of *ReceiverBusy* is sent.

Note: In accordance with the MediaLB Specification [5], the OS81210 does not acknowledge synchronous data transmissions (*RxStatus* default response of *ReceiverReady*).

When the OS81210 is the receiving Device for a logical channel and recognizes a reception error, it responds with an *RxStatus* of *ReceiverProtocolError* in the next physical channel of that logical channel. The following conditions cause INIC to generate *ReceiverProtocolError*, when receiving control messages and asynchronous packets:

- Inappropriate MediaLB *Command* for a particular channel
  (e.g. a logical channel is setup for control messages when a synchronous command is received).

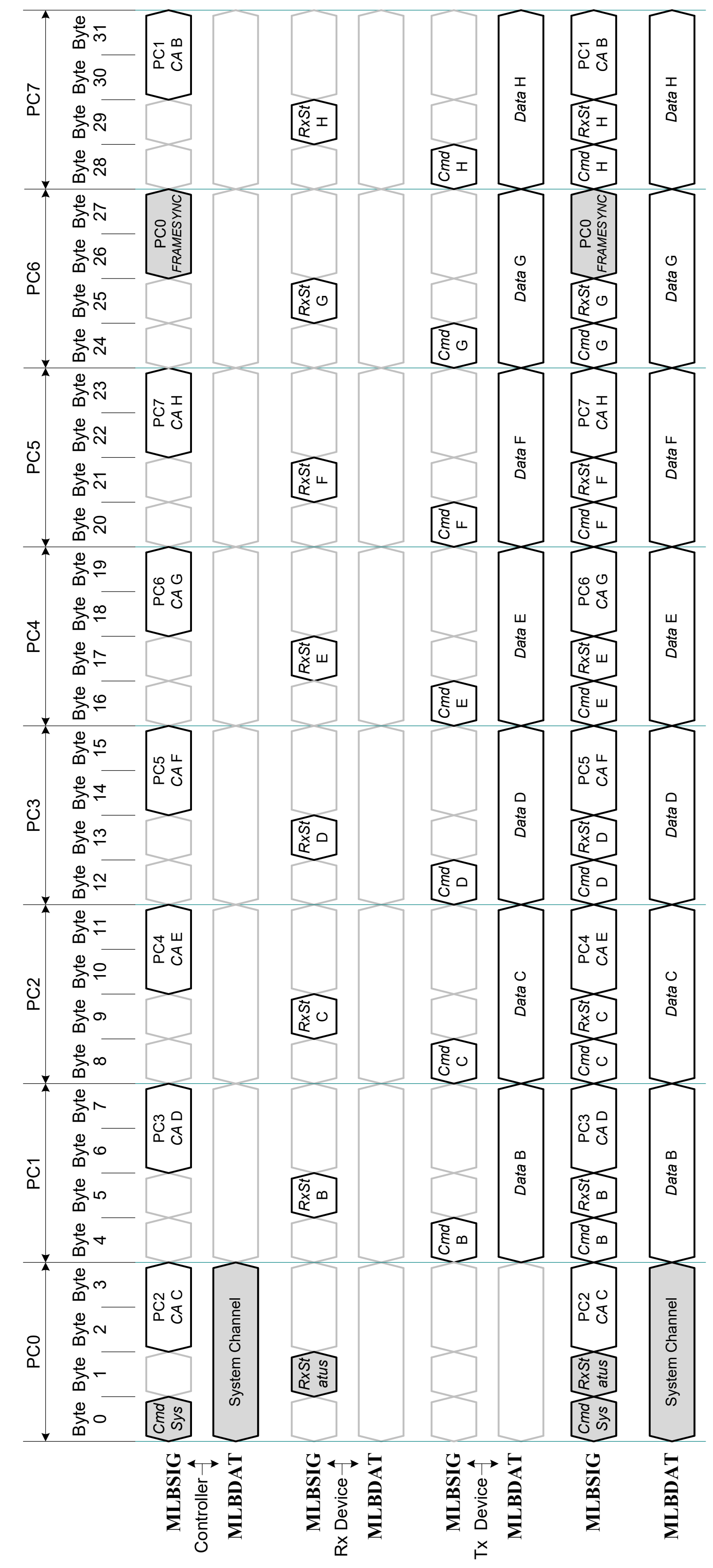
- *ControlStart*, *AsyncStart*, *ControlEnd*, or *AsyncEnd* is received while in the middle of a packet reception.
- *ControlEnd* or *AsyncEnd* is not received when the end of the buffer is expected, based on *Port Message Length* (PML).
- The PML indicates a size that is greater than allowed for a control message.

Figure 6-4 illustrates the MediaLB interface, as various Devices drive the bus (**MLBSIG**/**MLBDAT**) during a given network frame. This figure depicts the 256×Fs MediaLB speed; however, the logic also applies to the 512×Fs and 1024×Fs speeds (with more PCn channels).

The Controller drives the *ChannelAddresses* (*ChannelAddress* B through H in this example), granting the corresponding Devices bus access. The transmitting Device for the *ChannelAddress* drives the *Command* and *Data*, and the receiving Device responds with reception status (*RxStatus*). In the second to last physical channel of the frame (PC6 at 256×Fs), the Controller drives the *FRAMESYNC* pattern onto the signal information line (**MLBSIG**) in place of a *ChannelAddress*, for synchronization to the network frame. The *Command* and *Data* associated with the *FRAMESYNC* (also known as the *SystemChannel*) is sent by the Controller one quadlet following the *FRAMESYNC* pattern, in the first physical channel of the Network frame (PC0).

Depending on the number of physical channels grouped into logical channels, fewer unique *ChannelAddresses* may be seen in the frame. In Figure 6-4, each logical channel is one quadlet (one physical channel), mapping to seven *ChannelAddresses* (B through H). If one of these logical channel consisted of two quadlets and another consisted of three quadlets, then only four unique *ChannelAddresses* would be seen on the bus (B through E).

FIGURE 6-4: MediaLB 3-PIN INTERFACE EXAMPLE (AT 256×FS)



6.2.1 MediaLB COMMANDS AND RESPONSES

Network data can be transported across MediaLB using various MediaLB *Commands* and *RXStatus* responses. The commands/responses used for a logical channel are dependent on the data type transported. Table 6-2 shows the MediaLB *Commands* and *RxStatus* responses associated with each network data type.

TABLE 6-2: MediaLB COMMANDS & RESPONSES SUPPORTED

Network Data	Supported Commands	Supported Responses
Synchronous Data	*SyncData* *NoData*[*]	-
Asynchronous Packets	*AsyncStart* *AsyncContinue* *AsyncEnd* *AsyncBreak* *NoData*[*]	*ReceiverReady* *ReceiverBusy* *ReceiverBreak* *ReceiverProtocolError*
Control Messages	*ControlStart* *ControlContinue* *ControlEnd* *ControlBreak* *NoData*[*]	*ReceiverReady* *ReceiverBusy* *ReceiverBreak* *ReceiverProtocolError*
DiscreteFrame Isochronous (DFI) Phase	*IsoNoData* *Iso1Byte* *Iso2Bytes* *Iso3Bytes* *Iso4Bytes* *IsoSync1Byte* *IsoSync2Bytes* *IsoSync3Bytes* *IsoSync4Bytes*	-
A/V Packetized Isochronous (AVP) Packets	*IsoNoData* *Iso1Byte* *Iso2Bytes* *Iso3Bytes* *Iso4Bytes* *IsoSync1Byte* *IsoSync2Bytes* *IsoSync3Bytes* *IsoSync4Bytes*	-
QoS Packets[†]	*AsyncStart* *AsyncContinue* *AsyncEnd*	*ReceiverReady* *ReceiverBusy*

* For synchronous logical channels, the *NoData* command indicates that the Tx Device assigned to that *ChannelAddress* has not setup the channel yet. For asynchronous and control logical channels, *NoData* is used during packet data transfer where there is no data available to transmit.

† The *AsyncBreak Command* is not supported on logical channels exchanging QoS packets. As a MediaLB receiver, the OS81210 never responds with an *RxStatus* of *ReceiverProtocolError* or *ReceiverBreak*. The only supported responses for QoS data packets are *ReceiverReady* and *ReceiverBusy*.

The OS81210 does not support the following MediaLB *Commands* and *RxStatus* responses:

- System Commands:
  - E0h - *MOSTLock*
  - E2h - *MOSTUnlock*
  - E4h - *MlbScan*
  - E6h - *MlbSubCmd*
  - FEh - *MlbReset*
- System Responses:
  - 80h - *DevicePresent*
  - 82h - *DeviceServiceRequest*

6.3 MediaLB Debug Header

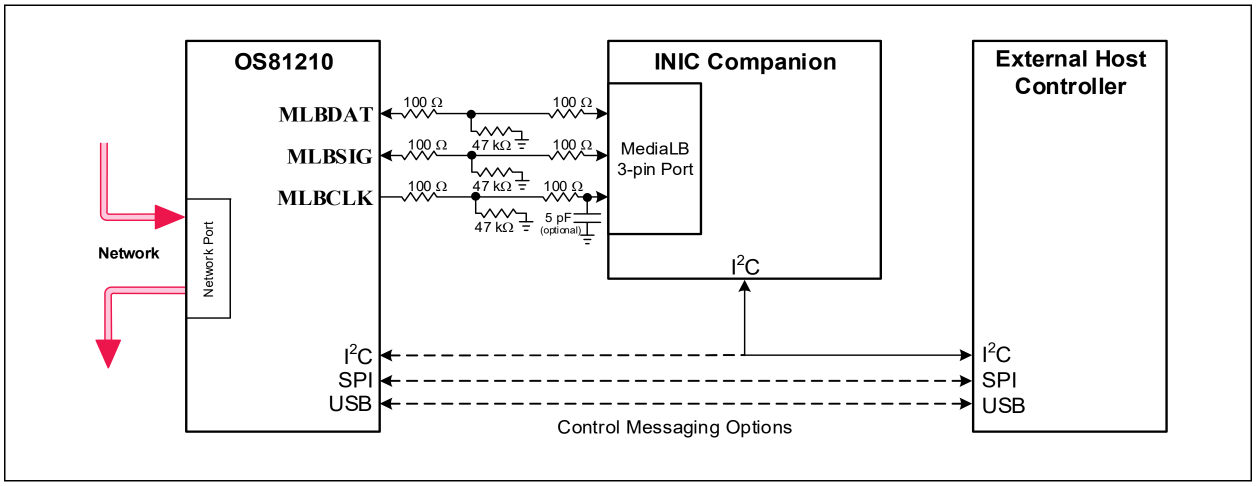
Standard MediaLB Debug Headers are defined to allow various MediaLB analysis and debug tools to connect directly into the application. MediaLB Debug Headers are designed to connect with *MediaLB Active Pods*, which are used with various MediaLB analysis and debug tools (see MediaLB Monitor Adapter User's Manual [8] and MediaLB Analyzer User's Manual [9]).

The specific header required depends on the MediaLB mode and speed. Refer to the MediaLB Specification [5] for more information about specified MediaLB Debug Headers. The header used for monitoring a MediaLB system should be placed as close as possible to the MediaLB interface to prevent unnecessary extension of signal traces. Layout examples demonstrating appropriate header placement are also included in the MediaLB Specification [5].

6.4 MediaLB Port Expansion

Applications requiring additional features not provided by the INIC may utilize an *I/O Companion* (IOC) device (e.g. OS85654/6 or OS85621/3) connected to INIC's MediaLB Port. See Section 3.3.9 "Port Expansion" for more information regarding the IOC family of devices. Communication between the EHC and the IOC is typically via the I^{2}C Control Port as depicted in Figure 6-5. However, if the EHC includes a MediaLB 3-pin interface then it may be attached to the bus in a 3-node system as shown in Figure 6-2.

FIGURE 6-5: MediaLB 3-PIN MODE TO PARALLEL INTERFACE



7.0 STREAMING PORT

The Streaming Port is used by external devices (e.g. ADCs, DACs, microphones, etc.) to source and sink streaming data with low overhead. The Streaming Port can exchange synchronous data in formats which are compatible with industry-standard serial interfaces including left-justified, right-justified, I^2S-compatible, time-division multiplex (TDM), or sequential formats. Mono microphones with a pulse density modulation (PDM) output are supported using the sequential format.

The OS81210 supports two Streaming Ports (Streaming Port A and Streaming Port B). Each Streaming Port provides two serial data pins: **SRXA0** and **SRXA1** for Streaming Port A, and **SRXB0** and **SRXB1** for Streaming Port B. In addition to the two serial data pins, Streaming Port A provides a bit clock (**SCKA**) and frame synchronization signal (**FSYA**). Streaming Port B islinked to Streaming Port A in which the **FSYA/SCKA** clock signals are shared by all the data pins of the two ports. The direction of the Streaming Port A and Streaming Port B data pins can be configured independently, as shown in Table 7-1 and Table 7-2.

TABLE 7-1: STREAMING PORT A CONFIGURATION OPTIONS

Streaming Port A Mode	I/O Options	SRXA0	SRXA1	Comments
Generic Streaming	Two Input Pins	Input (SRA0)	Input (SRA1)	All data pins share FSYA/SCKA. Data pins of Streaming Port B may also be available based on configu-ration.
	Two Output Pins	Output (SXA0)	Output (SXA1)	
	One Input Pin/ One Output Pin	Input (SRA0)	Output (SXA1)	

TABLE 7-2: STREAMING PORT B CONFIGURATION OPTIONS

Streaming Port B Mode	I/O Options	SRXB0	SRXB1	Comments
Generic Streaming	Two Input Pins	Input (SRB0)	Input (SRB1)	All data pins share FSYA/SCKA of Steaming Port A.
	Two Output Pins	Output (SXB0)	Output (SXB1)	
	One Input Pin/ One Output Pin	Input (SRB0)	Output (SXB1)	

Note: Streaming Port A must be created before Streaming Port B, as Streaming Port B cannot be the sole Streaming Port. With both Streaming Port A and B in Generic Steaming mode, Streaming Port B can be linked to Streaming Port A in which all data pins (**SRXA0**, **SRXA1**, **SRXB0**, **SRXB1**) utilize the Streaming Port A clocking signals (**FSYA** and **SCKA**).

Refer to the INIC Interface Specification [4] for more information on configuring the Streaming Port options and data formats.

7.1 Generic Streaming Data Exchange

All data pins on the Streaming Port share a common bit clock (**SCKA**) and a common start-of-frame synchronization signal (**FSYA**) which is often also used to delineate left/right channel boundaries. The bit clock (**SCKA**) operates at 64×Fs, 128×Fs, 256×Fs, or 512×Fs where Fs is the frame rate of **FSYA**. With both Streaming Port A and B in Generic Streaming mode, Streaming Port B can be linked to Streaming Port A and all data pins (**SRXA0**, **SRXA1**, **SRXB0**, **SRXB1**) share the Streaming Port A clocking signals (**FSYA** and **SCKA**).

When synchronous data is exchanged on the serial data pins, the **FSYA** and **SCKA** pins can be configured as inputs (Slave Mode) or outputs (Master Mode). As inputs, **FSYA** and **SCKA** must be synchronous to the network (derived from **RMCK**). In this configuration, the external **FSYA** source only needs to meet a minimum high/low time (see Section 15.7). As outputs, the **FSYA** and **SCKA** pins are driven based on the network frame rate. When configured as an output, the **FSYA** pin is driven by the INIC at a 50% duty cycle, or the INIC can drive a one-bit **FSYA** when using the TDM format. When using Master Mode, there is a clock mode where INIC can drive **FSYA** phase locked to the network frame. In this mode, the latency between an audio source to the network or the network to an audio sink is always 2 frames periods (41.6 µs at 48 kHz). When **FSYA** is not phase locked (regardless of Slave or Master mode), there is an arbitrary relation between **FSYA** and the network frame; therefore, the latency can vary between 1 to 4 frame periods.

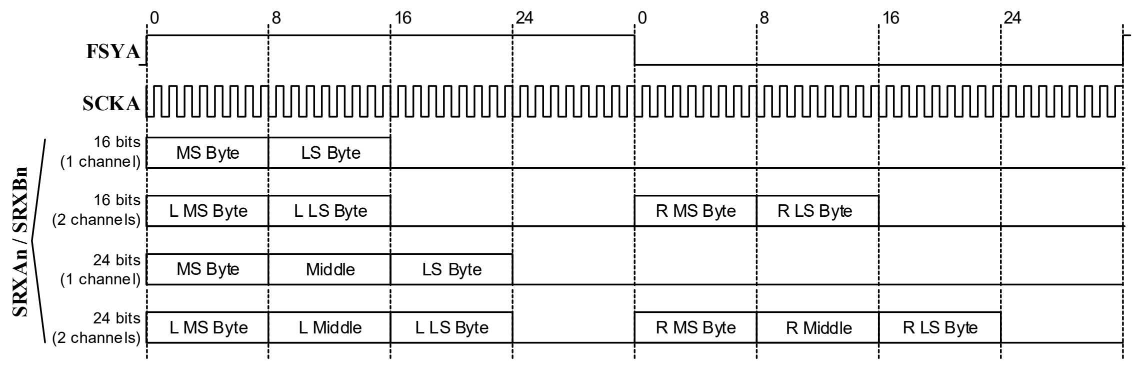
Refer to the INIC Interface Specification [4] for more information on configuring the Streaming Port options and data formats.

7.1.1 LEFT-JUSTIFIED ALIGNMENT

When using the left-justified alignment format, the first data bit occurs on the first clock after a transition on **FSYA**. The data and **FSYA** signals change on the falling edges of **SCKA** and are valid on the rising edges. Data is arranged MSB first, high byte first. The start of the frame is indicated by the rising edge of **FSYA**.

Figure 7-1 illustrates the data alignment and timing when the left-justified format is selected. For higher data bit rates, the channel data remains aligned with the edges of **FSYA** with more unused data bytes between channels.

FIGURE 7-1: LEFT-JUSTIFIED STREAMING DATA FORMAT

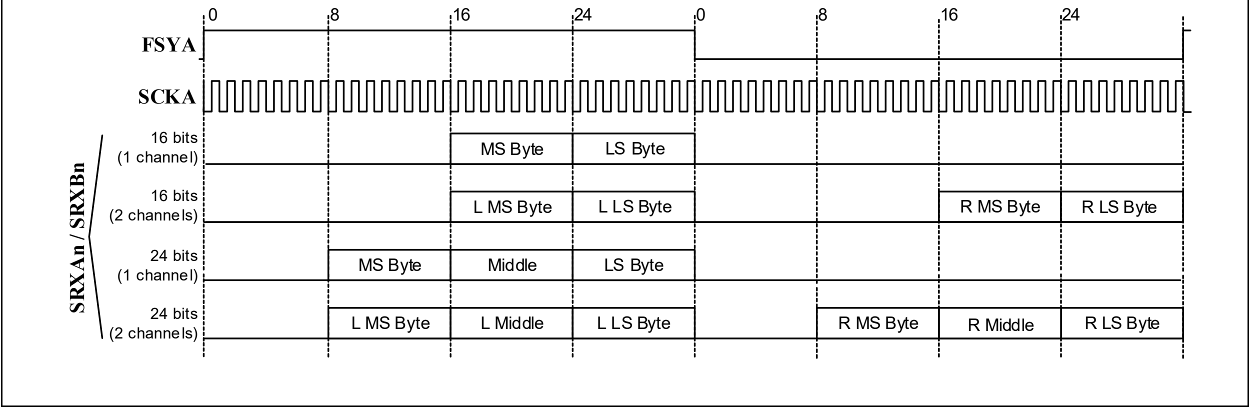


7.1.2 RIGHT-JUSTIFIED ALIGNMENT

When using the right-justified alignment format, the last data bit occurs on the last clock before a transition on **FSYA**. The data and **FSYA** signals change on the falling edges of **SCKA** and are valid on the rising edges. Data is arranged MSB first, high byte first. The start of the frame is indicated by the rising edge of **FSYA**.

Figure 7-2 illustrates the data alignment and timing when the right-justified format is selected. For higher data bit rates, the channel data remains aligned with the edges of **FSYA** with more unused data bytes between channels.

FIGURE 7-2: RIGHT-JUSTIFIED STREAMING DATA FORMAT

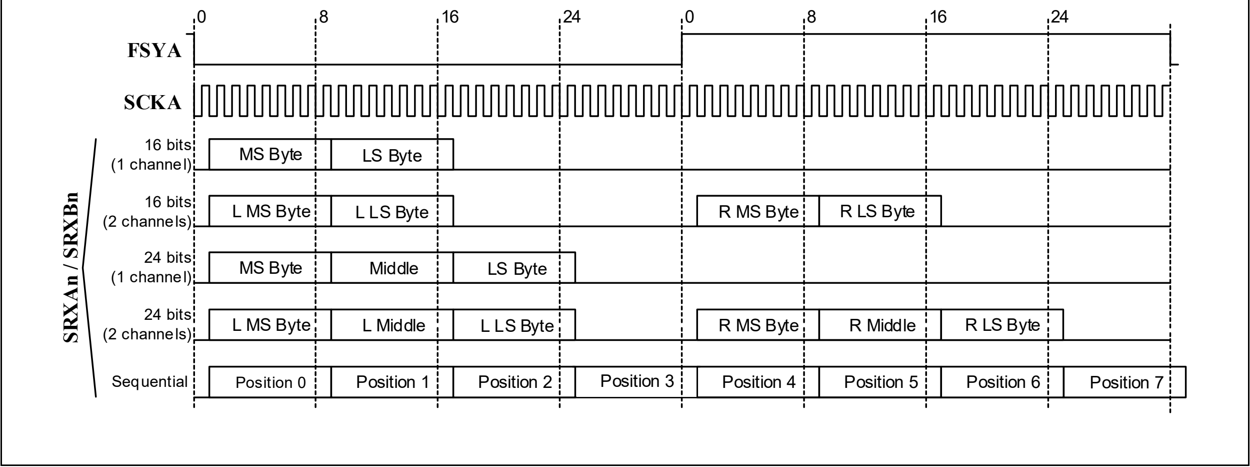


7.1.3 DELAYED-BIT ALIGNMENT

When using the delayed-bit alignment format, which is I^2S-compatible, there is a single **SCKA** clock delay between the start of frame (falling edge of **FSYA**) and the start of the frame data on **SRXn**. **FSYA** and **SRXn** change on the falling edges of **SCKA** and are valid on the rising edges. Data is arranged MSB first, high byte first.

Figure 7-3 illustrates the data alignment and timing when the delayed-bit format is selected. For higher data bit rates, the channel data remains bit-delayed from the edges of **FSYA** with more unused data bytes between channels.

FIGURE 7-3: DELAYED-BIT DATA FORMAT (I^2S COMPATIBLE)



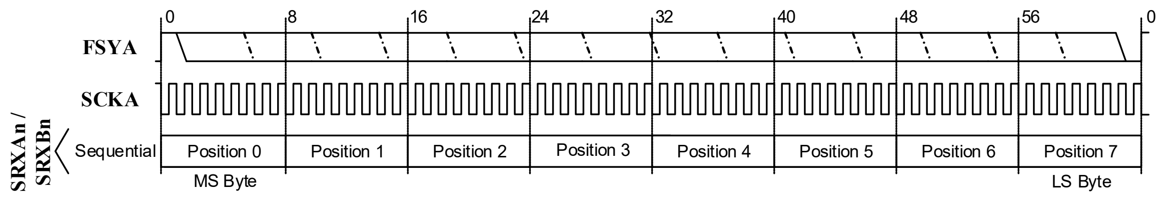
For a single (mono) channel using delayed-bit alignment format at higher **SCKA** rates, a sequential, delayed-bit alignment format is available (as shown in Figure 7-3). When using this format, byte positions in a frame remain bit-delayed and fixed; however, the actual number of byte positions *used* in the frame is flexible. This allows sockets of various sizes to be connected.

7.1.4 SEQUENTIAL ALIGNMENT

When using the sequential (non-delayed) alignment format, the data and FSYA signals change on the falling edges of SCKA and are valid on the rising edges. Data is arranged MSB first, high byte first. This format allows data bytes to exist at every byte position in the frame; however, the actual number of byte positions *used* is flexible. This allows a socket of custom size to be connected.

Figure 7-4 illustrates the byte positions and timing for the sequential (non-delayed) alignment format when SCKA is 64×Fs. For higher data bit rates, more byte positions are available.

FIGURE 7-4: SEQUENTIAL DATA FORMAT



Because the sequential alignment format does not place restrictions on socket size, this format can support a single (mono) channel. For example, with SCKA at 128×Fs, a single 16-bit data channel could be sent in the following byte positions:

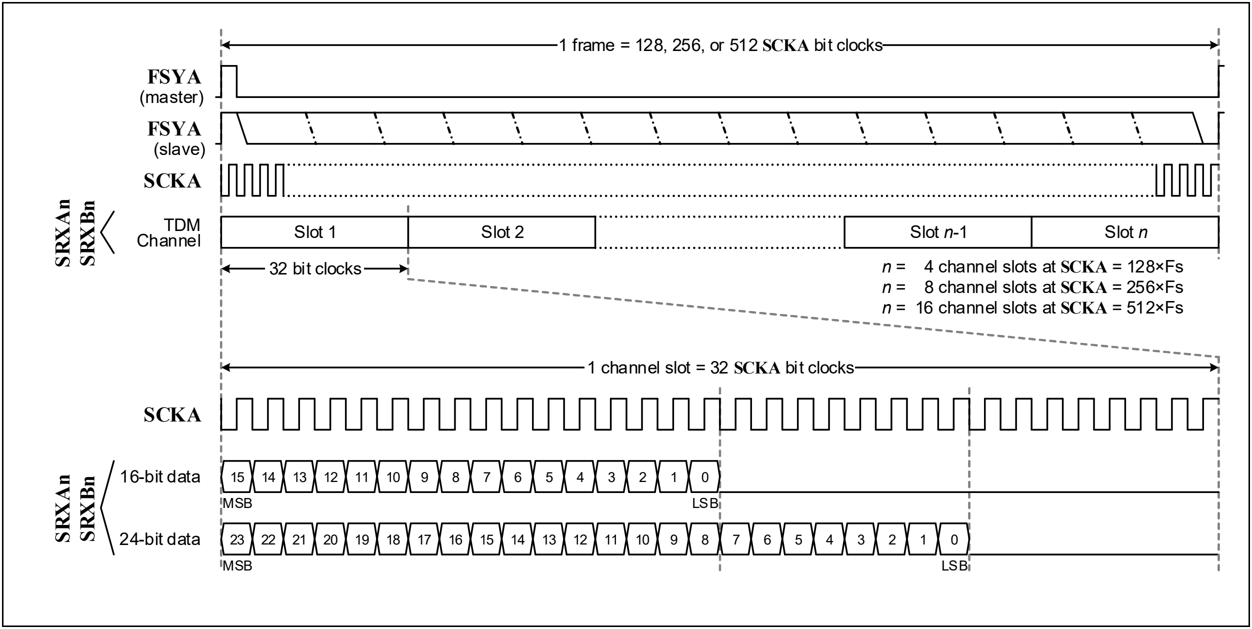
- Positions 0 and 1 (for left-justified alignment)
- Positions 6 and 7 (for right-justified alignment)

The sequential alignment format also supports pulse density modulation (PDM) mono microphone applications. Only SCKA is required to provide the timebase for PDM microphones. The microphone's continuous bit stream enters a single Streaming Port data pin. When using this feature, SCKA is typically configured at 64×Fs and all data byte positions of the frame will be filled by the microphone. Stereo PDM is not supported.

7.1.5 TIME-DIVISION MULTIPLEX

The time-division multiplex (TDM) format enables the routing of multiple data streams on a single data pin. The data and FSYA signals change on the falling edges SCKA and are valid on the rising edges. Data is transferred MSB first. Each frame of 128, 256, or 512 bits is composed of multiple 32-bit fixed-size *channel slots*. Figure 7-5 illustrates the TDM (non-delayed) alignment format. Identified in Figure 7-5 by the variable *n*, the number of channel slots per frame is dependent upon the SCKA bit clock rate. Four 32-bit channel slots are available per data pin at a clock rate of 128×Fs, eight channel slots are available at 256×Fs, and 16 channel slots are available at 512×Fs. All data is routed within each channel slot most-significant bit first and left-justified. Either 16-bit or 24-bit data may be transferred within the TDM channel slots and all channel slots for a pin are configured to transfer the same number of data bits per channel slot; i.e., it is not possible to mix routing of 16-bit and 24-bit data within channel slots of a single pin. However, not all channel slots within the frame must be used to route data. All unrouted transmit channel slots and channel slot bytes are transmitted as zeros.

FIGURE 7-5: TIME-DIVISION MULTIPLEX FORMAT


1 frame = 128, 256, or 512 SCKA bit clocks
FSYA (master)
FSYA (slave)
SCKA
SRXAn SRXBn
TDM Channel
Slot 1
Slot 2
Slot n-1
Slot n
32 bit clocks
n = 4 channel slots at SCKA = 128×Fs
n = 8 channel slots at SCKA = 256×Fs
n = 16 channel slots at SCKA = 512×Fs
1 channel slot = 32 SCKA bit clocks
16-bit data
24-bit data
MSB
LSB

The TDM alignment format may be configured for operation in one of three clock formats:

- TDM Non-Delayed - The first channel slot is aligned with the start of the frame identified by the rising edge of **FSYA** (Figure 7-5).
- TDM Delayed-Bit (bit-delayed only) - The first channel slot is delayed from the start of frame by one bit clock and a rising edge of **FSYA** signifies the start of the frame (Figure 7-6).
- TDM Delayed-Bit (I^{2}S compatible) - The first channel slot is delayed by one bit clock from the start of the frame coincident with a falling edge on **FSYA** (Figure 7-7).

FIGURE 7-6: TIME-DIVISION MULTIPLEX FORMAT (DELAYED-BIT ONLY)

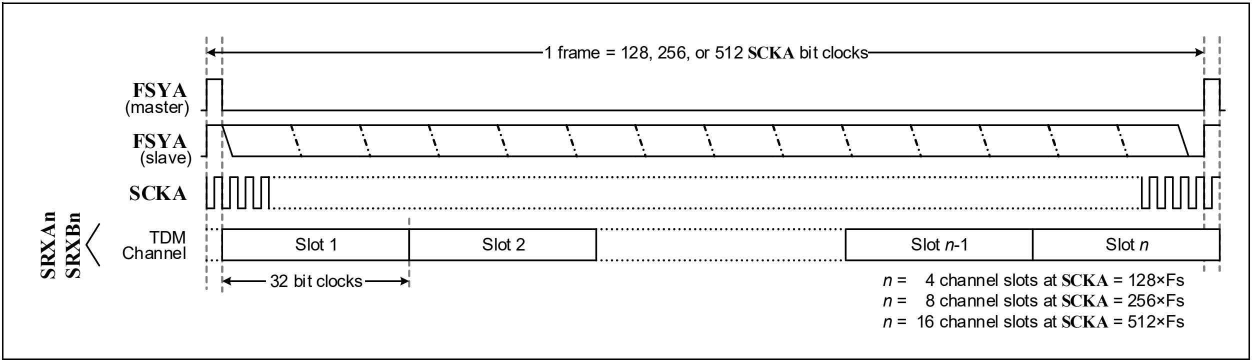
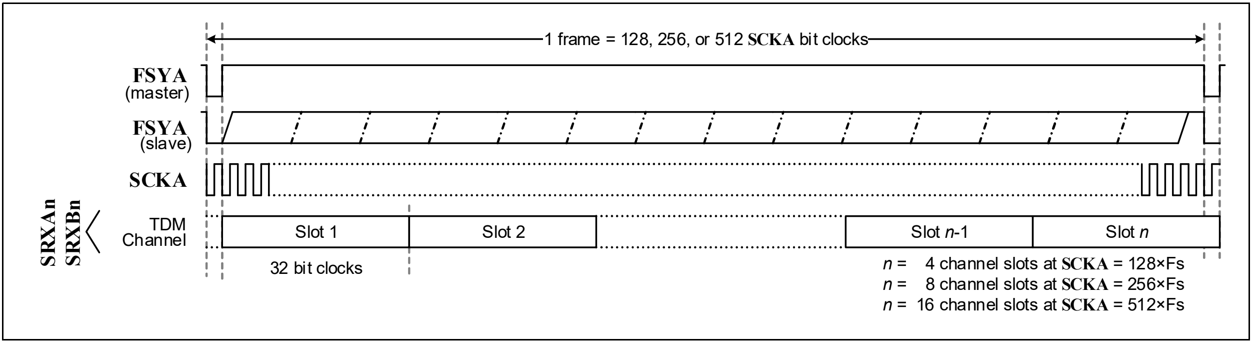

1 frame = 128, 256, or 512 SCKA bit clocks
FSYA (master)
FSYA (slave)
SCKA
SRXAn SRXBn
TDM Channel
Slot 1
Slot 2
Slot n-1
Slot n
32 bit clocks
n = 4 channel slots at SCKA = 128×Fs
n = 8 channel slots at SCKA = 256×Fs
n = 16 channel slots at SCKA = 512×Fs

FIGURE 7-7: TIME-DIVISION MULTIPLEX FORMAT (DELAYED-BIT - I^{2}S COMPATIBLE)


1 frame = 128, 256, or 512 SCKA bit clocks
FSYA (master)
FSYA (slave)
SCKA
SRXAn SRXBn
TDM Channel
Slot 1
Slot 2
Slot n-1
Slot n
32 bit clocks
n = 4 channel slots at SCKA = 128×Fs
n = 8 channel slots at SCKA = 256×Fs
n = 16 channel slots at SCKA = 512×Fs

8.0 USB PORT

The OS81210 USB Port supports on-PCB connection between the OS81210 and a USB 2.0-compliant host. Enabling the USB Port gives the EHC (USB Host) access to control messages, asynchronous packets, synchronous data, and A/V Packetized (AVP) isochronous streaming packets.

The USB Port supports communication of control messages between the OS81210 and the EHC. Control messages may be used to configure the OS81210, target the local INIC FBlocks (ICMs), or be routed by INIC between the EHC and the network control channel (RCMs or MCMs). Refer to the INIC Interface Specification [4] for more detailed information.

Note:	The USB Port operates as a self-powered USB device as defined in the Universal Serial Bus Specification 2.0 [15]. The OS81210 only supports USB 2.0 High-speed (480 Mbit/s) operation.

In addition to the bidirectional USB control pipe (Endpoint 0, IN/OUT), up to 14 unidirectional configurable endpoints are available (7 IN and 7 OUT), supporting bulk transfers. The maximum size of a bulk transfer is 512 bytes. Packets less than 512 bytes are completed with a single transfer. Multiple bulk transfers are required to transmit packets greater than 512 bytes. The end of the application packet is marked by a bulk transfer that is less than 512 bytes (also referred to as a short packet) or a zero length transfer (also referred to as a *zero length packet* or ZLP). Depending on size, an *Ethernet Frame* sent over an MEP to the INIC may require up to 3 bulk transfers on a configurable endpoint.

Refer to the INIC Interface Specification [4] for configuration options for the USB Port and vendor-specific descriptors. To guarantee proper operation, the USB communication tree must fulfill the INIC's endpoint requirements described in the INIC Interface Specification [4].

8.1 Physical Interfaces

The USB Port supports either a USB physical interface or an HSIC physical interface. Although these two physical interfaces do not share pins, they cannot be enabled simultaneously. At any given time, only a single physical interface can be operational. The active physical interface for the USB Port is selected in the *INIC Configuration String*.

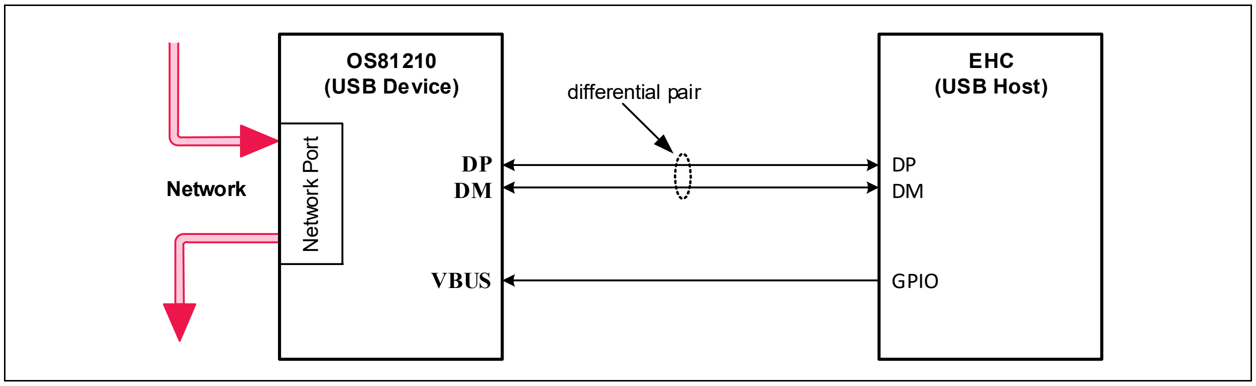
8.1.1 USB INTERFACE

The OS81210 USB Interface consists of three pins: **DM**, **DP**, and **VBUS**. Through the USB Interface, the OS81210 is a self-powered USB device and is intended for implementations contained on a single application board. The **DM** and **DP** differential signal pair are used to implement bidirectional communication between the OS81210 and the USB host (e.g. EHC). The **VBUS** input is used to detect when the host is present and ready for communication. In order to protect other devices in the system, the OS81210 USB Port is disabled until **VBUS** is high. This signal is typically controlled by a GPIO pin on the host. Alternatively, it may be pulled high to 3.3V switched power through an external resistor.

Note: The OS81210 USB physical interface is designed for on-PCB communication only and is not 5 V tolerant. Cabled USB applications are not supported for production applications.

Figure 8-1 shows a typical connection diagram for communication between an external USB Host and INIC using the USB Interface. Layout recommendations for the OS81210 USB physical interface can be found in Section 16.9.4 “USB Port”.

FIGURE 8-1: USB PORT TYPICAL CONNECTIONS - USB INTERFACE



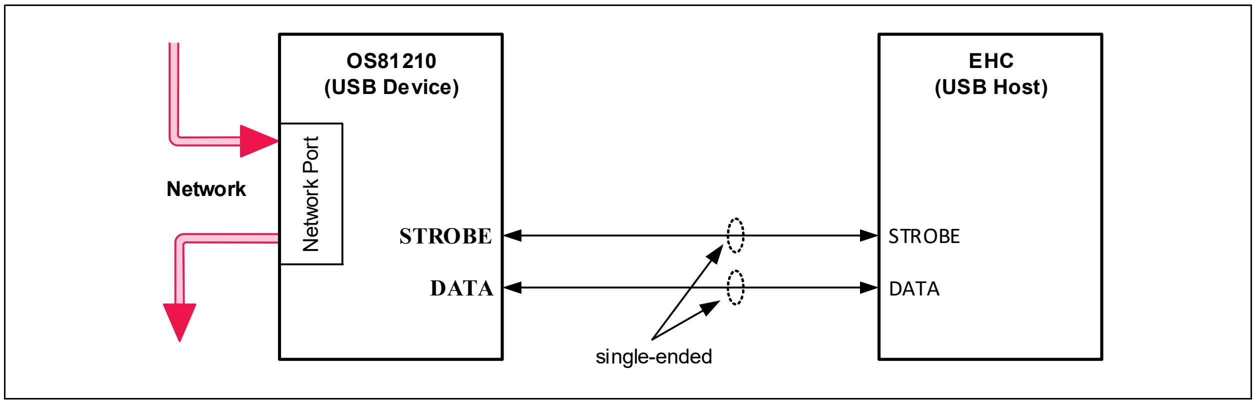
Refer to the Universal Serial Bus Specification 2.0 [15] for more information about the operation of the USB interface.

8.1.2 HSIC INTERFACE

The OS81210 HSIC Interface consists of the **DATA** and **STROBE** pins. The HSIC physical interface uses low-voltage signaling and single-ended signals to lower system cost and reduce power requirements compared to the standard USB physical interface. HSIC is specifically designed for on-PCB applications.

Figure 8-2 shows a typical connection diagram for communication between an external USB Host and INIC using the HSIC Interface. Layout recommendations for the OS81210 HSIC Interface can be found in Section 16.9.4 “USB Port”.

FIGURE 8-2: USB PORT TYPICAL CONNECTIONS - HSIC INTERFACE



Refer to the High-Speed Inter-Chip USB Electrical Specification 1.0 [16] for more information about the operation of the HSIC interface.

9.0 SPI PORT

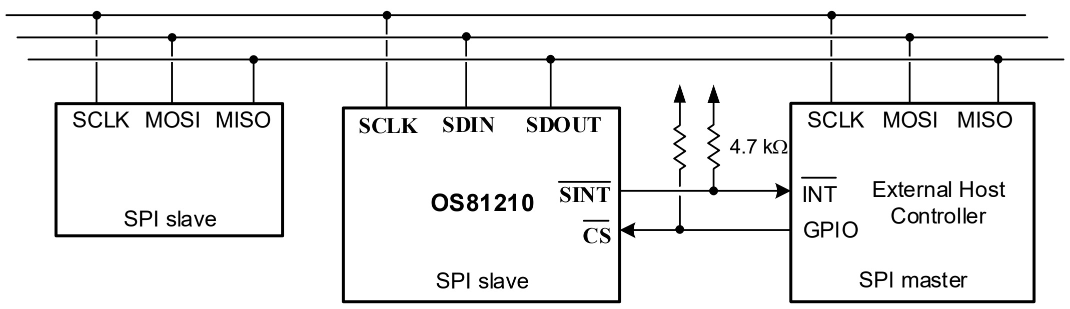
The *Serial Peripheral Interface* (SPI) Port is designed to interface with SPI Port available on many industry devices. In addition to the four standard SPI signals ($\overline{\text{CS}}$, **SCLK**, **SDIN**, and **SDOUT**), the SPI Port uses an additional interrupt line ($\overline{\text{SINT}}$) which is used for signaling to an External Host Controller (EHC). When the SPI Port is enabled, it operates as an SPI bus slave and can exchange data packets at a rate up to 25 Mbit/s (e.g. **SCLK**=512×Fs at Fs=48 kHz). The SPI Port supports the exchange of asynchronous and control packets (*MHP Frames/Ethernet Frames* and *Application Control Frames)* at data rates higher than possible with I^2C.

Table 9-1 shows the 5-pin SPI Port, including the four standard signals and the additional interrupt. Typical connections for the SPI Port are shown in Figure 9-1. External pull-up resistors are recommended on $\overline{\text{SINT}}$ and $\overline{\text{CS}}$ to maintain these signals in a known state while the interface is inactive (e.g. power-up/reset).

TABLE 9-1: SERIAL PERIPHERAL INTERFACE

Pin Name	Description	Direction (Slave Mode)
$\overline{\text{CS}}$	Chip Select (active low)	input
SDIN	Serial Data In (MOSI)	input
SDOUT	Serial Data Out (MISO)	output
SCLK	Clock	input
$\overline{\text{SINT}}$	Interrupt (active low)	output

FIGURE 9-1: SPI PORT PIN CONNECTIONS



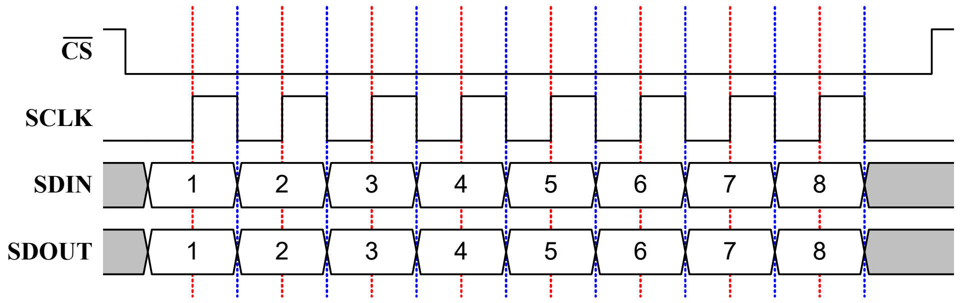
9.1 Clock Modes

Four different clock modes are available on the SPI Port. Refer to the INIC Interface Specification [4] for more information on configuring the SPI Port.

9.1.1 MODE 0

In Clock Mode 0, **SCLK** is *low* when the bus is idle. Data transitions on the *falling* clock edge and is latched on the *rising* clock edge. This mode is defined as (CPOL=0, CPHA=0) by the SPI Block Guide Version 4.01 [10].

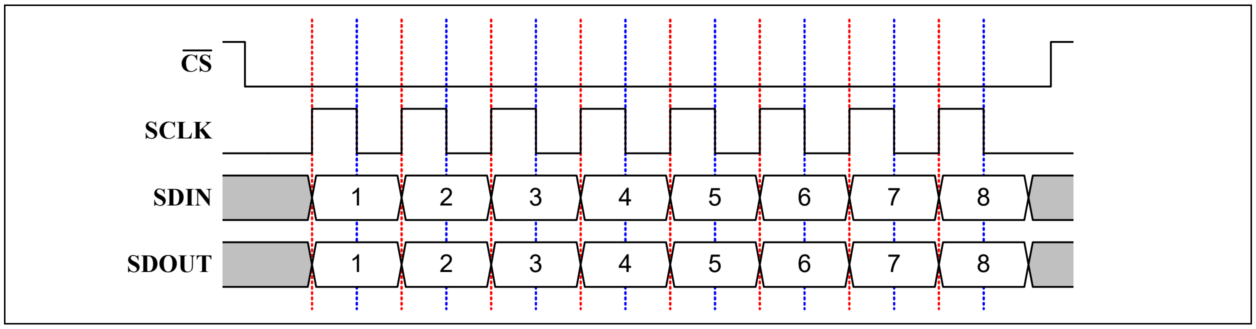
FIGURE 9-2: SPI CLOCK MODE 0



9.1.2 MODE 1

In Clock Mode 1, **SCLK** is *low* when the bus is idle. Data transitions on the *rising* clock edge and is latched on the *falling* clock edge. This mode is defined as (CPOL=0, CPHA=1) by the SPI Block Guide Version 4.01 [10].

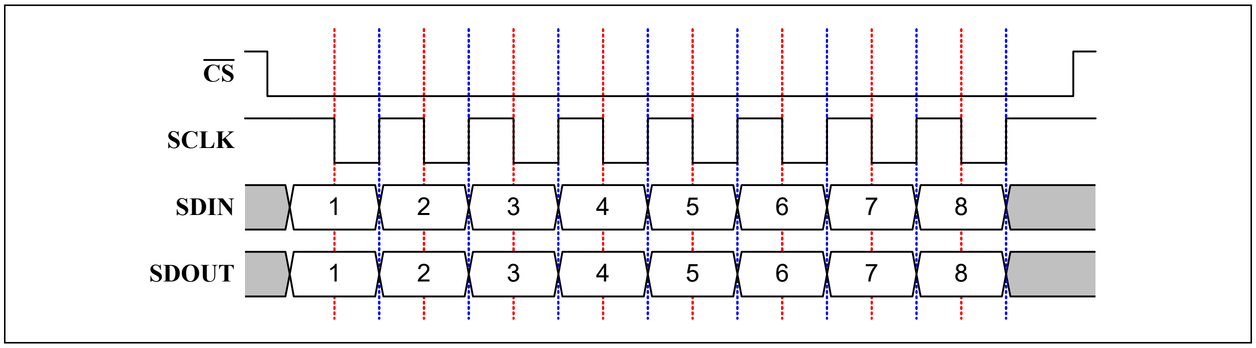
FIGURE 9-3: SPI CLOCK MODE 1



9.1.3 MODE 2

In Clock Mode 2, **SCLK** is *high* when the bus is idle. Data transitions on the *rising* clock edge and is latched on the *falling* clock edge. This mode is defined as (CPOL=1, CPHA=0) by the SPI Block Guide Version 4.01 [10].

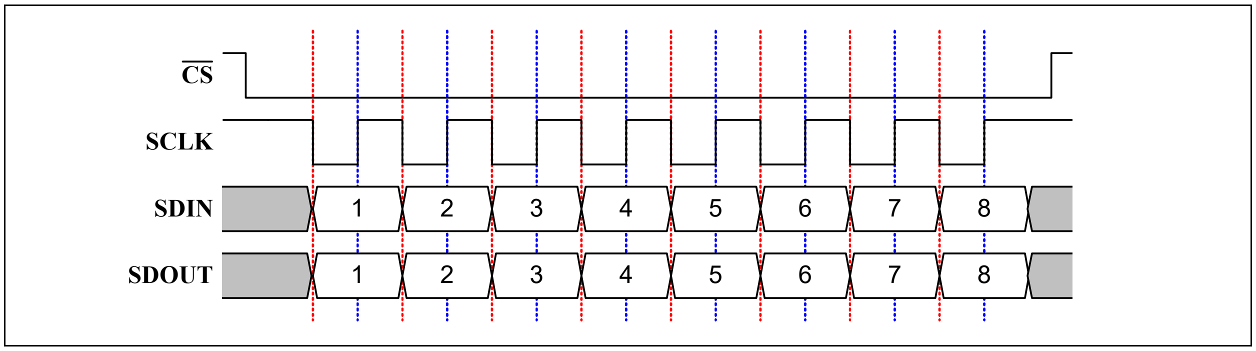
FIGURE 9-4: SPI CLOCK MODE 2



9.1.4 MODE 3

In Clock Mode 3, **SCLK** is *high* when the bus is idle. Data transitions on the *falling* clock edge and is latched on the *rising* clock edge. This mode is defined as (CPOL=1, CPHA=1) by the SPI Block Guide Version 4.01 [10].

FIGURE 9-5: SPI CLOCK MODE 3



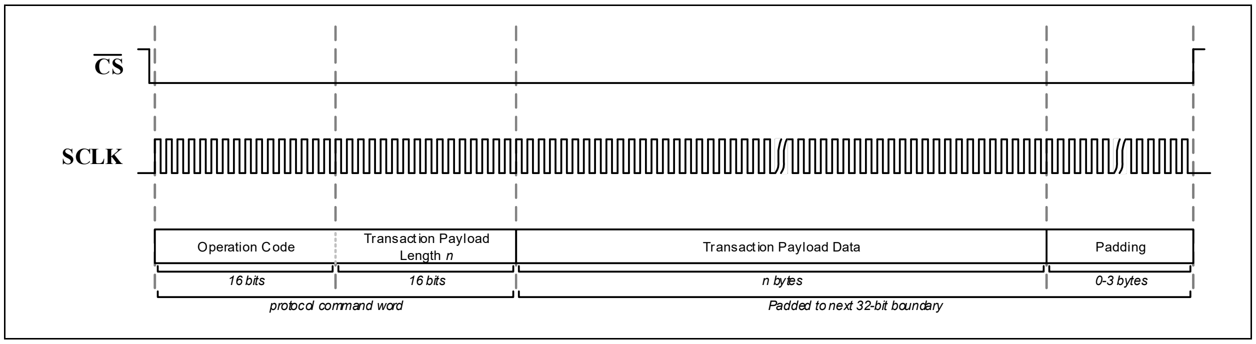
9.2 SPI Protocol

As illustrated in Figure 9-6, the SPI bus master (also referred to as the EHC) initiates communication with the SPI Port by driving $\overline{\text{CS}}$ low. A *protocol command word* is the first four bytes sent by the EHC on **SDIN** after the falling edge of $\overline{\text{CS}}$. This 32-bit command word defines the direction of the data transfer (e.g., read or write operation), the resource being accessed (registers or packet data channels), and the length of the *transaction payload data* that follows. The remaining bytes of the transaction contain the *transaction payload data* being read from or written to the SPI Port. Multiple transactions are not allowed while $\overline{\text{CS}}$ is asserted; i.e., only one transaction is allowed for each assertion of $\overline{\text{CS}}$. The $\overline{\text{CS}}$ pin must remain asserted through the entire transaction, only going high following the last bit clock of the transaction.

The SPI Port supports 32-bit word aligned communication. All SPI Port transactions must therefore be a multiple of 4 bytes in length. This is accomplished by adding one to three padding bytes of data at the end of the transaction, if necessary.

Data is always shifted most significant bit first into **SDIN** and out of **SDOUT**.

FIGURE 9-6: SPI TRANSACTION



Note:	During the time that the EHC is writing the 32-bit command word to the SPI Port on **SDIN**, there is no data for the INIC to output on **SDOUT**. The **SDOUT** pin is therefore driven low for the first 32-bits of every transaction. Similarly, when the EHC is performing a write transaction to the SPI Port, **SDOUT** remains driven low during the entire write transaction. The INIC only drives data onto the **SDOUT** pin following the *protocol command word* during read transactions.

9.2.1 PROTOCOL COMMAND WORD

A *protocol command word* is the first four bytes sent by the EHC on the **SDIN** data line after the falling edge of $\overline{\text{CS}}$. The 32-bit *protocol command word* consists of a 16-bit *operation code* (opcode) followed by a 16-bit *transaction payload length*. As shown in Table 9-2, the opcode defines the direction of the transaction data transfer, the SPI Port resource being accessed, and the resource address. The opcode read-not-write (**RNW**) bit indicates if the transaction is a read from or a write to the SPI port. The channel-not-register resource selector bit (**CNR**) selects the type of resource (control/status registers or logical packet channels) within the SPI to be accessed. When the **CNR** bit is clear, the transaction will access a set of SPI configuration and status registers. These registers are described in detail in Section 9.7. With **CNR** set, the transaction will access one of the packet communication channels. The opcode address field (**ADDR**) specifies the address of the SPI Port resource being accessed (register or packet communication channel address). The 16-bit *transaction payload length* field specifies the number of data bytes in the *transaction payload data* that follows, excluding any necessary padding.

Note:	The *transaction payload length* field must never be zero, otherwise an SPI protocol error will result.

Table 9-2 shows the structure of the 16-bit opcode portion of the *protocol command word*.

TABLE 9-2: SPI TRANSACTION OPCODE

RNW	—	—	CNR	ADDR<3:0>			
15	14	13	12	11	10	9	8

—	—	—	—	—	—	—	—
7	6	5	4	3	2	1	0

bit 15	**RNW**: SPI transaction direction (Read-Not-Write) `0` = Write Transaction `1` = Read Transaction
bit 14-13	**Unimplemented**: Write as '0'
bit 12	**CNR**: Channel-Not-Register resource selection bit. `0` = Address SPI control and status registers `1` = Address packet communication channel resources
bit 11-8	**ADDR<3:0>**: SPI resource address When **CNR** is '0', the **ADDR** field addresses the control and status registers. See Section 9.7. When **CNR** is '1', the **ADDR** field selects the packet communication channels: `0000` = Driver Control Interface (DCI) Channel access `0010` = Asynchronous Channel access (*MHP Frames/Ethernet Frames)* `0100` = Control Channel access (*Application Control Frames)* All other encoding values are reserved and should not be used.
bit 7-0	**Unimplemented**: Write as '0'

9.3 Control / Status Register Transactions

The SPI Port implements a register interface which the EHC uses to control, configure, and obtain status of the SPI hardware. These control and status registers are described in detail in Section 9.7.

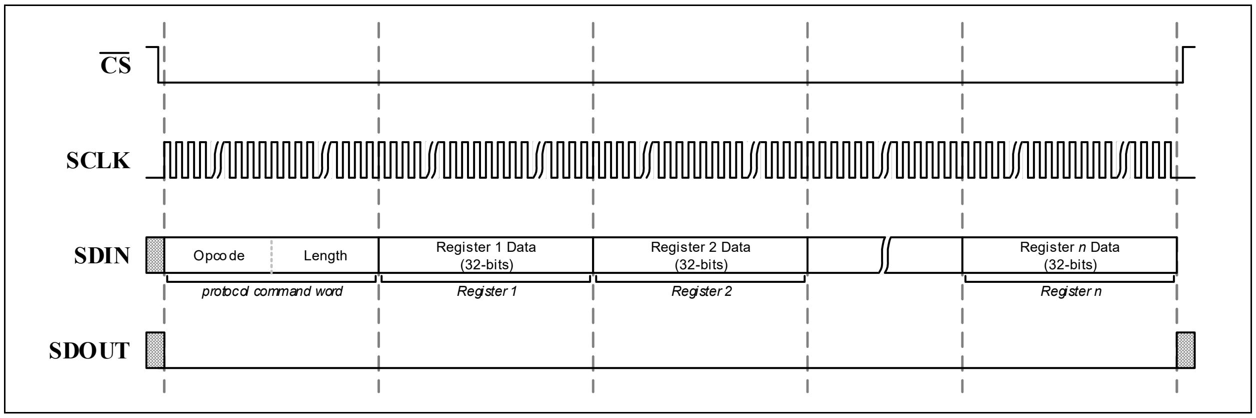
9.3.1 CONTROL / STATUS REGISTER WRITE TRANSACTION

When writing to SPI registers, the EHC constructs the *protocol command word* with the opcode **RNW** bit clear to indicate a write transaction, and the **CNR** bit clear to select addressing registers. The opcode address field, **ADDR**, selects the first register address to be written. For example, writing to register address 0x1 would result in a *protocol command word* opcode of 0x0100.

The transaction payload contains the data to be written to the registers. Each register is 32-bits in size, so four bytes of data are required in the transaction payload for each register written. Multiple registers with consecutive addresses may be written using a single write transaction as shown in Figure 9-7. A maximum of 8 consecutive registers may be written in a single transaction. Attempting to write more than 8 registers will result in the additional data being discarded.

The transaction payload must be a multiple of 4 bytes in length. The number of registers that will be written is equal to the size of the transaction payload (in bytes) divided by 4. As such, the *transaction payload length* field within the *protocol command word* must always be an integer multiple of four, therefore padding bytes are not necessary.

FIGURE 9-7: SPI REGISTER WRITE TRANSACTION



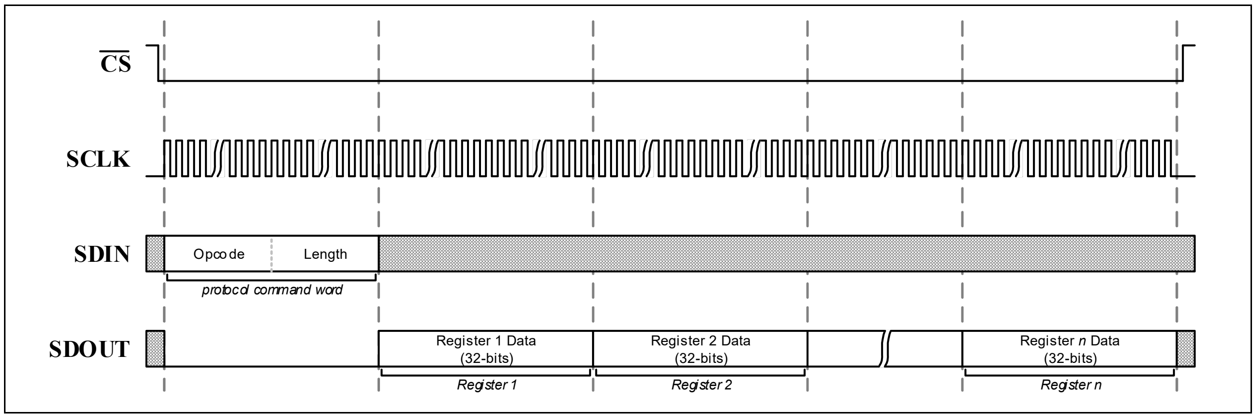
9.3.2 CONTROL / STATUS REGISTER READ TRANSACTION

Reading from SPI registers requires the EHC to construct the *protocol command word* with the opcode **RNW** bit set to indicate a read transaction, and the **CNR** bit clear to select addressing registers. The opcode address field, **ADDR**, selects the first register address to be read. For example, reading from register address 0x1 would result in a *protocol command word* opcode of 0x8100.

Following the *protocol command word* sent by the EHC on **SDIN**, the INIC will transmit the transaction payload onto **SDOUT**. The transaction payload contains the data read from the registers. Each register is 32-bits in size, so four bytes of data are transmitted in the transaction payload for each register read. Multiple registers with consecutive addresses may be read using a single read transaction as shown in Figure 9-8. A maximum of 8 registers may be read in a single transaction. Attempting to read more than 8 registers will result in zeros being read.

The transaction payload must be a multiple of 4 bytes in length. The number of registers that will be read is equal to the size of the transaction payload (in bytes) divided by 4. As such, the *transaction payload length* field within the *protocol command word* must always be an integer multiple of four.

FIGURE 9-8: SPI REGISTER READ TRANSACTION



9.4 Packet Transactions

The SPI Port supports the transfer of *MHP Frames/Ethernet Frames* and *Application Control Frames*. The transfer of packet messages through the SPI Port is accomplished by reading and writing one of the packet channel addresses. The EHC writes the *protocol command word* opcode **CNR** bit set as '1' to select addressing the packet channels. The address **ADDR** field is used to select the specific packet channel which will be read from or written to.

MHP Frames/Ethernet Frames and *Application Control Frames* are encapsulated within *Port Message*s which are transferred within the SPI transaction payload. The first two bytes of the *Port Message* contain the *Port Message Length* (PML). The PML contains the number of bytes remaining in the *Port Message*. Refer to the INIC Interface Specification [4] for more detailed information. When multiple packets are transferred in a single SPI transaction, the PML is used to locate the end of the current message and the beginning of the next message (and next PML).

9.4.1 PACKET WRITE TRANSACTION

The SPI Port is able to receive packets from the EHC when the packet receive status bit (**ARIS**, **CRIS**) is set within the CHSTS register. The receive status bit is set to ‘1’ when the SPI is able to receive at least one packet, and the total amount of buffer space available for receiving a packet is greater than the receive threshold (**ARTHR**, **CRTHR**). The receive threshold is typically configured to be equal to, or greater than the maximum size of packet the EHC may write to the SPI Port. The maximum size of an *MHP Frame* or *Ethernet Frame* is 1510 bytes, while a *Application Control Frame* is a maximum of 64 bytes in size. By configuring the receive threshold to the maximum packet size, the EHC is guaranteed to be able to write one maximal length packet each time the receive interrupt status bit is set.

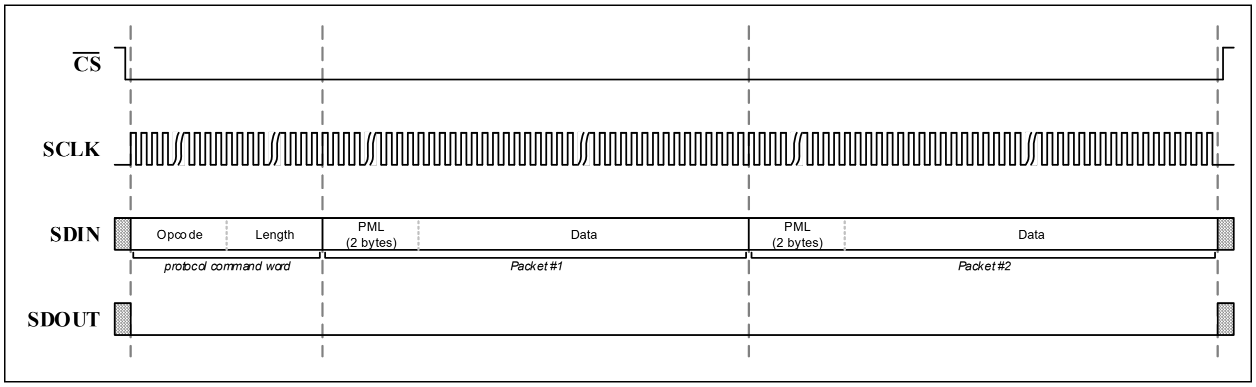
Prior to writing packets to the SPI Port, the EHC must determine how many packets and the total number of bytes that may be received by the SPI Port. This information is available by reading the receive buffer status register (ARXB, CRXB). The **RXPCNT** field specifies the maximum number of packets that the SPI Port is able to receive and the **RXFREE** field contains the maximum number of bytes that may be written by the EHC.

Note: The total number of bytes written, *plus one extra byte per packet*, must not exceed the number of bytes specified by **RXFREE** field.

The EHC writes packets to the SPI Port by constructing a *protocol command word* with the opcode **RNW** bit clear to indicate a write transaction and the **CNR** bit set to ‘1’ to select writing to the packet channels. The opcode address field, **ADDR**, selects the packet data channel (asynchronous or control) to which the packets will be written. For example, writing a *MHP Frame* or *Ethernet Frame* to the asynchronous channel requires a *protocol command word* opcode of 0x1200, whereas an opcode of 0x1400 results in writing a *Application Control Frame* to the control channel.

The transaction payload contains the packet data to be written to the packet channel. Multiple packets may be written using a single write transaction as shown in Figure 9-9. Only whole packets may be written; writing of partial (i.e., incomplete) packets is not permitted. When multiple packets are being written in a single transaction, padding is only applied at the end of the *transaction payload data*, not between packets.

FIGURE 9-9: SPI PACKET WRITE TRANSACTION



9.4.2 PACKET READ TRANSACTION

The SPI Port has data packets available for the EHC to read when the packet transmit status (**ATIS**, **CTIS**) bit is set within the CHSTS register. The transmit status bit is set automatically when the INIC has at least one packet available for reading. Once the EHC has determined that data packets are available for reading from the SPI Port, the transmit buffer status register (ATXB, CTXB) status register is read to determine the number of packets and total number of bytes the SPI Port has ready for the EHC to read. The **TXPCNT** field contains the number of packets available for the EHC to read. The **TXAVAIL** field reflects the total number of packet bytes required to read the packets.

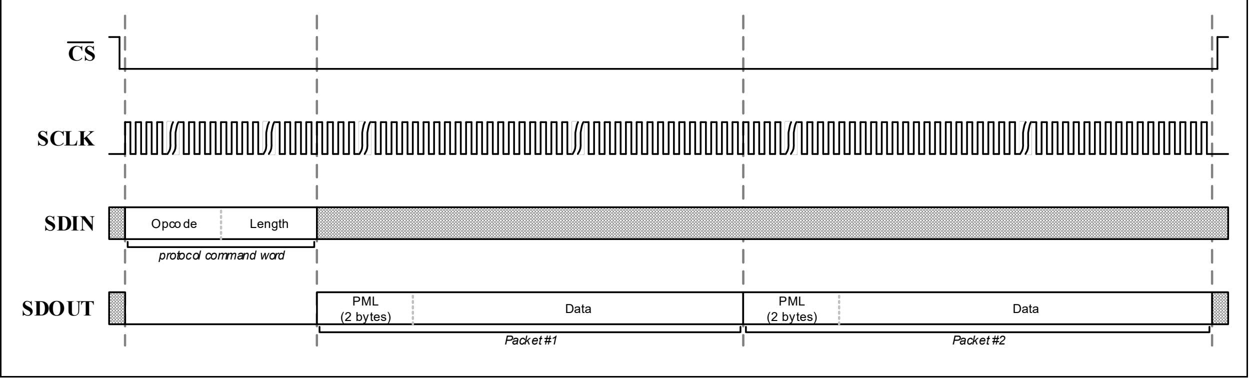
Reading packets from the SPI Port requires the EHC to construct a *protocol command word* with the opcode **RNW** bit set to indicate a read transaction and the **CNR** bit set to ‘1’ to select reading from the packet channels. The opcode address field, **ADDR**, selects the packet data channel (asynchronous or control) from which the packets will be read. For example, a *protocol command word* opcode of 0x9200 results in the reading of a *MHP Frame* or *Ethernet Frame* from the asynchronous channel and an opcode of 0x9400 results in reading a *Application Control Frame* from the control channel.

Following the write of the *protocol command word* by the EHC on SDIN, INIC transmits the transaction payload onto SDOUT. The transaction payload contains the packet data read from the logical packet data channel. Multiple packets may be read using a single read transaction as shown in Figure 9-10. Reading of partial (incomplete) packets is not permitted.

Note: The *transaction payload length* field must never be zero, otherwise an SPI protocol error will result, possibly resulting in the loss of data packets.

Note: As padding is only applied at the end of the *transaction payload data*. When multiple packets are being read in a single transaction, padding will not be inserted between packets. Therefore, successive packets in the transaction may not begin on a 32-bit aligned boundary.

FIGURE 9-10: SPI PACKET READ TRANSACTION



9.5 Driver Control Interface (DCI)

The EHC normally accesses all INIC configuration and status through the normal Control Message interface (e.g., MediaLB, I^2C Control Port). However, the Control Message interface may not always be available due to an interface error or device reset. In these conditions, the Driver Control Interface (DCI) enables the EHC driver to directly access low-level driver related configuration and status of the INIC when no other communication interface is available. The low-level configuration and status are implemented as a virtual register which the EHC accesses by writing *DCI Command* packets to the DCI channel. Each *DCI Command* either writes data to the DCI virtual registers, or requests data to be read from the registers and returned in a *DCI Response* packet.

Low-level INIC configuration and status accessible to the EHC via the DCI include:

- Retrieval and configuration of INIC status such as Node Address, Packet EUI-48 Address, Network Availability, etc.
- Packet channel synchronization in the event of a fatal channel error
- Device synchronization in the event the EHC or INIC resets or encounters a fatal error
- Bidirectional notification between the EHC and INIC on status changes or events including resource availability, channel and device synchronization, fatal errors, etc.

The DCI also implements an interface to communicate events between the INIC and EHC. The EHC may signal to the INIC that a *DCI Command* has been written is ready for processing by the INIC (**CMDREQ**), or that the EHC driver requires synchronization with the INIC (**SYNCREQ**). Three status flags are used by the INIC to signal events to the EHC. The **CMDDN** status indicates that the INIC has completed processing of a *DCI Command* and a *DCI Response* is ready for the EHC to read. The INIC uses the **NTF** status bit to notify the EHC that a DCI virtual register status has changed. Should the INIC detect an error condition, it will be signaled with the **ERR** status bit. The DCI may be configured to assert the **DEIS** bit and the $\overline{\text{SINT}}$ pin on the occurrence of any or all of the three INIC generated event status flags being asserted (see Figure 9-13).

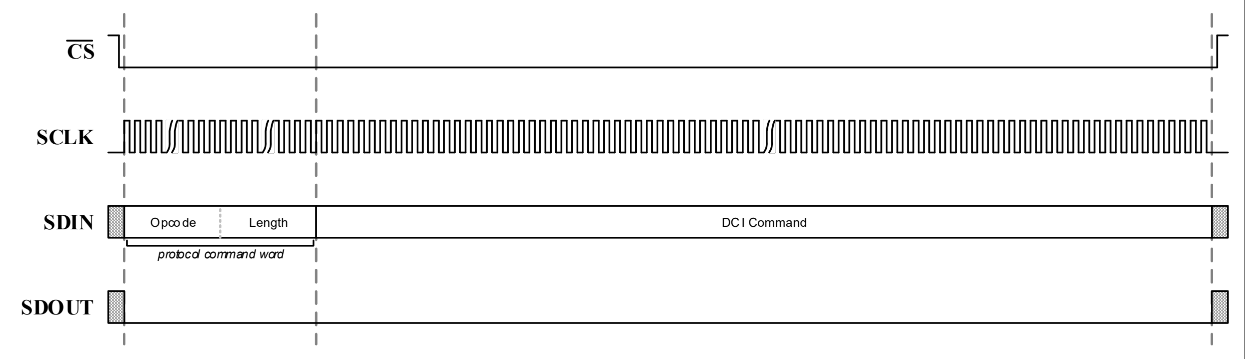
Refer to the INIC Interface Specification [4] for detailed information regarding the *DCI Command* and *DCI Response* packet formats as well as the virtual register mapping.

9.5.1 DCI COMMAND WRITE

The writing of *DCI Command* packets is performed in a manner similar to the writing of packet messages to the control or asynchronous channels. The INIC is ready to accept a *DCI Command* packet when the **CMDREQ**, **CMDDN**, and **SYNCREQ** bits in the DCCTL register are all ‘0’. The EHC writes a *DCI Command* packet to the SPI Port by constructing a *protocol command word* with the opcode **RNW** bit clear to indicate a write transaction and the **CNR** bit set to ‘1’ to select writing to the packet channels. The opcode address field, **ADDR**, is set accordingly to select the DCI data channel for writing. For example, writing a *DCI Command* packet requires a *protocol command word* opcode of 0x1000.

The transaction payload follows the *protocol command word* and contains the *DCI Command* packet to be written to the DCI channel as shown in Figure 9-11. The *DCI Command* is always an integer multiple of 4 bytes in length, therefore no padding bytes are necessary. Once the EHC has written the *DCI Command*, the INIC is notified of the pending command by setting **CMDREQ** to ‘1’ at which time the INIC will execute the posted *DCI Command* and clear **CMDREQ**.

FIGURE 9-11: DCI COMMAND PACKET WRITE TRANSACTION



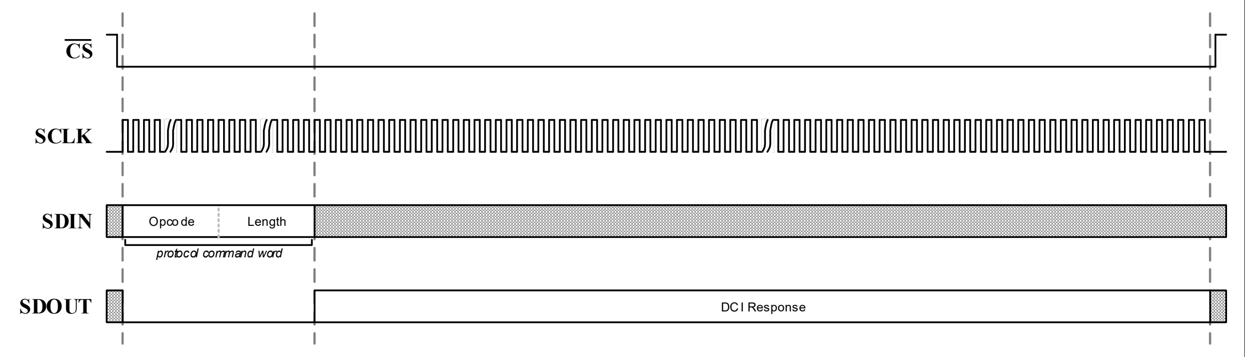
9.5.2 DCI RESPONSE READ

Once the INIC has executed a *DCI Command*, it will clear the **CMDREQ** bit. When INIC has responded to the command with a *DCI Response*, it will set the **CMDDN** bit to ‘1’ to alert the EHC that a response packet is to be read. If unmasked, the **DEIS** bit in the CHSTS register will also be set when a *DCI Response* packet is available for reading.

Reading a *DCI Response* packet from the SPI Port requires the EHC to construct a *protocol command word* with the opcode **RNW** bit set to indicate a read transaction and the **CNR** bit set to ‘1’ to select reading from the packet channels. The opcode address field, **ADDR**, is set to select reading from the DCI data channel. For example, a *protocol command word* opcode of 0x9000 results in the reading of *DCI Response* packets from the DCI data channel.

Following the write of the *protocol command word* by the EHC on **SDIN**, INIC transmits the transaction payload containing the *DCI Response* onto **SDOUT** (see Figure 9-12). The *DCI Response* is always an integer multiple of 4 bytes in length, therefore no padding bytes are transmitted. Once the EHC has read the *DCI Response*, the **CMDDN** bit is cleared by writing a ‘1’.

FIGURE 9-12: DCI RESPONSE PACKET READ TRANSACTION



9.6 Interrupt Functionality

As an alternative to polling the CHSTS register to determine a change if the SPI Port state, the EHC can instead configure and monitor $\overline{\text{SINT}}$. The $\overline{\text{SINT}}$ pin is a shared signal to indicate availability for packet read and write transactions on the asynchronous and control packet channels, and pending DCI events the EHC should react to.

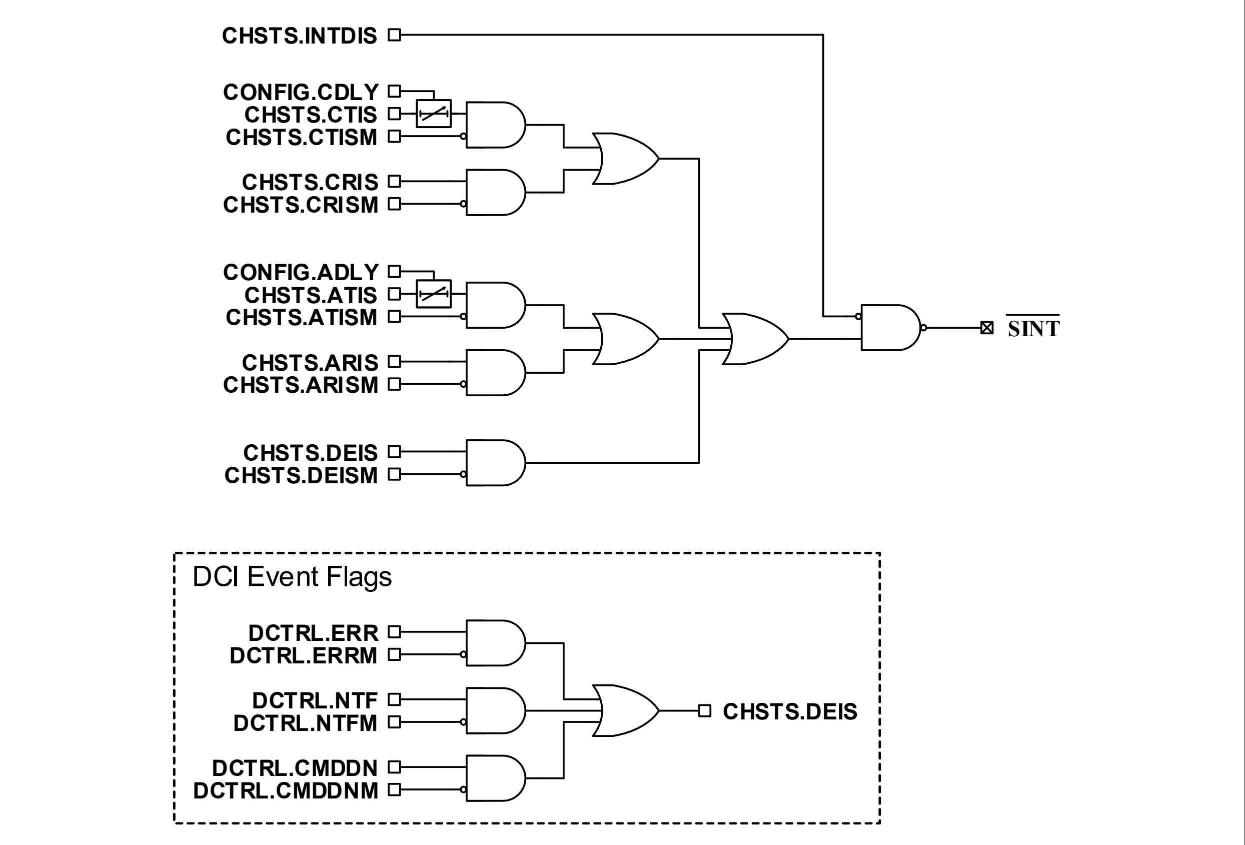
When an *Application Control Frame* or *MHP Frame/Ethernet Frame* becomes ready for reading or writing, the corresponding interrupt channel status bit within the CHSTS register will be set causing the $\overline{\text{SINT}}$ pin to be asserted. The interrupt channel status bit will remain pending until the event that caused the status to be set is resolved. When any of the interrupt channel status bits within the CHSTS register are set, the $\overline{\text{SINT}}$ pin is asserted to notify the EHC that the SPI Port is available for packet communication. In addition, the $\overline{\text{SINT}}$ pin may be used to indicate the assertion of the DCI event status, **DEIS**, indicating a need for the EHC to read the DCI virtual registers to determine the source and nature of the event.

Each interrupt source status bit that can generate an assertion of the $\overline{\text{SINT}}$ pin has a corresponding interrupt mask bit. When the interrupt mask bit is set, the associated interrupt channel status bit will not cause an assertion of the $\overline{\text{SINT}}$ pin. Additionally, assertion of the $\overline{\text{SINT}}$ pin may be disabled globally by setting the **CHSTS.INTDIS** bit. When disabled, $\overline{\text{SINT}}$ will remain driven high.

The logic for asserting the $\overline{\text{SINT}}$ pin is illustrated in Figure 9-13.

After it is asserted, $\overline{\text{SINT}}$ remains low until the SPI transaction starts. After the transaction completes and $\overline{\text{CS}}$ is de-asserted, the INIC will then check if there are new interrupts pending (from channel sources or DCI).

FIGURE 9-13: $\overline{\text{SINT}}$ INTERRUPT PIN LOGIC



9.6.1 INTERRUPT DELAY

When a burst of small *Packet Frames/Ethernet Data Frames* or *Control Frames* (referred to as *MHP Frames/ Ethernet Frames* or *Application Control Frames* at the EHC driver level) is received from the network for transmission to the EHC, the $\overline{\text{SINT}}$ pin may repeatedly become asserted in quick succession, possibly adding an increased load on the EHC to process the interrupts by reading each packet individually. An alternative is to limit the assertion rate of the $\overline{\text{SINT}}$ pin when asynchronous packets are ready for transmit by the INIC.

When the asynchronous or control packet transmit delay mode is enabled by setting **ADLY**, or **CDLY**, respectively, an internal timer is started when the first packet is ready for transmit to the EHC and the packet transmit interrupt status bit (**ATIS**, or **CTIS**) is set. Only after the timer has expired will the $\overline{\text{SINT}}$ pin assert. This delay allows the EHC to read and process multiple packets at once.The amount of delay is configured in **INTDLY**.

While the ability to delay assertion of $\overline{\text{SINT}}$ until multiple, small packets are ready for reading is useful, it is also desirable for the EHC to immediately read and process a large *Packet Frame* or *Ethernet Data Frame* containing application data from the asynchronous channel. Therefore, when the asynchronous transmit interrupt delay mode is enabled, the $\overline{\text{SINT}}$ pin may be asserted early before the delay expiration if the total number of bytes pending read by the EHC is greater than the asynchronous packet transmit threshold, **ATTHR**. The interrupt delay feature is exclusive to the asynchronous channel and is not extended to the control channel or DCI.

9.7 Control / Status Registers

A register interface is provided to permit the EHC software driver to configure the SPI Port and access port status. These registers are accessed using register read and write transactions as described in Section 9.3.

TABLE 9-3: SPI REGISTER SUMMARY

Address	Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
0x0	CONFIG	31:24	CRTHR<3:0>				ARTHR<3:0>			
		23:16	—	—	—	—	ATTHR<3:0>			
		15:8	—	—	—	—	—	—	CDLY	ADLY
		7:0	INTDLY<7:0>							
0x1	CHSTS	31:24	INTDIS	—	—	DEISM	CTISM	CRISM	ATISM	ARISM
		23:16	—	—	—	DEIS	CTIS	CRIS	ATIS	ARIS
		15:8	PCE	DCIES	—	—	—	—	—	—
		7:0	CRES	CRCA	CTES	CTCA	ARES	ARCA	ATES	ATCA
0x2	DBC	31:24	LEN<7:0>							
		23:16	BS<7:0>							
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	—	—	—
0x3	DCTRL	31:24	—	—	—	—	—	ERRM	NTFM	CMDDNM
		23:16	—	—	—	—	—	ERR	NTF	CMDDN
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	—	CMDREQ	SYNCREQ
0x4	ARXB	31:24	RXPCNT<15:8>							
		23:16	RXPCNT<7:0>							
		15:8	RXFREE<15:8>							
		7:0	RXFREE<7:0>							
0x5	ATXB	31:24	TXPCNT<15:8>							
		23:16	TXPCNT<7:0>							
		15:8	TXAVAIL<15:8>							
		7:0	TXAVAIL<7:0>							
0x6	CRXB	31:24	RXPCNT<15:8>							
		23:16	RXPCNT<7:0>							
		15:8	RXFREE<15:8>							
		7:0	RXFREE<7:0>							
0x7	CTXB	31:24	TXPCNT<15:8>							
		23:16	TXPCNT<7:0>							
		15:8	TXAVAIL<15:8>							
		7:0	TXAVAIL<7:0>							

REGISTER 9-1: CONFIG: SPI CONFIGURATION

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CRTHR<3:0>				ARTHR<3:0>			
31	30	29	28	27	26	25	24

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	ATTHR<3:0>			
23	22	21	20	19	18	17	16

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	CDLY	ADLY
15	14	13	12	11	10	9	8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTDLY<7:0>							
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 27-24 **CRTHR<3:0>**: Control Packet Receive Buffer Free Threshold
Specifies the minimum number of free bytes available in the SPI control packet buffer which the EHC may write before the **CRIS** status bit is set pending.
`0000` = 8 bytes
`0001` = 16 bytes
`0010` = 24 bytes
`0011` = 32 bytes
`0100` = 48 bytes
`0101` = 64 bytes
`0110` = 80 bytes
`0111` = 96 bytes
`1000` = 128 bytes
`1001` = 256 bytes
`1010` = 512 bytes

All other encoding values are reserved and should not be used.

bit 27-24 **ARTHR<3:0>**: Asynchronous Packet Receive Buffer Free Threshold
Specifies the minimum number of free bytes available in the SPI asynchronous packet buffer which the EHC may write before the **ARIS** status bit is set pending.
`0000` = 64 bytes
`0001` = 96 bytes
`0010` = 128 bytes
`0011` = 192 bytes
`0100` = 256 bytes
`0101` = 384 bytes
`0110` = 512 bytes
`0111` = 768 bytes
`1000` = 1024 bytes
`1001` = 1536 bytes
`1010` = 2048 bytes
`1011` = 3072 bytes

All other encoding values are reserved and should not be used.

bit 23-20	**Unimplemented:** Write as ‘0’
bit 19-16	**ATTHR<3:0>**: Asynchronous Packet Transmit Buffer Available Threshold Specifies the minimum number of bytes of asynchronous packet data available in the SPI asynchronous packet buffer available for the EHC to read. This field is only valid if the asynchronous packet transmit interrupt delay (**ADLY**) is enabled. `0000` = 64 bytes `0001` = 96 bytes `0010` = 128 bytes `0011` = 192 bytes `0100` = 256 bytes `0101` = 384 bytes `0110` = 512 bytes `0111` = 768 bytes `1000` = 1024 bytes `1001` = 1536 bytes `1010` = 2048 bytes `1011` = 3072 bytes All other encoding values are reserved and should not be used.
bit 15-10	**Unimplemented:** Write as ‘0’
bit 9	**CDLY**: Interrupt delay mode for control packet transmit `1` = If enabled and not masked, assertion of the $\overline{\text{SINT}}$ pin will be delayed from when the control transmit interrupt status (**CTIS**) is set. `0` = If enabled and not masked, the $\overline{\text{SINT}}$ pin will be immediately asserted when the control transmit interrupt status (**CTIS**) is set.
bit 8	**ADLY**: Interrupt delay mode for asynchronous packet transmit `1` = If enabled and not masked, assertion of the $\overline{\text{SINT}}$ pin will be delayed from when the asynchronous transmit interrupt status (**ATIS**) is set. `0` = If enabled and not masked, the $\overline{\text{SINT}}$ pin will be immediately asserted when the asynchronous transmit interrupt status (**ATIS**) is set.
bit 7-0	**INTDLY<7:0>**: Interrupt Delay Count. Specifies a period of time which to delay packet channel transmit ready interrupts. The delay is specified in number network frames. `00000000` = 1/Fs seconds (20.8 µs at Fs = 48 KHz) `00000001` = 2/Fs seconds (41.7 µs at Fs = 48 KHz) `00000010` = 3/Fs seconds (62.5 µs at Fs = 48 KHz) ... `01111111` = 128/Fs seconds (2.67 ms at Fs = 48 KHz) ... `11111111` = 256/Fs seconds (5.33 ms at Fs = 48 KHz) The delay counter is derived from an internal frame marker. Therefore, if a status interrupt bit becomes pending immediately prior to the next frame marker, the effective delay before $\overline{\text{SINT}}$ is asserted may be one frame period less than specified (20.8 µs at Fs = 48 KHz).

REGISTER 9-4: CHSTS: CHANNEL STATUS

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTDIS	—	—	DEISM	CTISM	CRISM	ATISM	ARISM
31	30	29	28	27	26	25	24

U-0	U-0	U-0	R-0	R-0	R-0	R-0	R-0
—	—	—	DEIS	CTIS	CRIS	ATIS	ARIS
23	22	21	20	19	18	17	16

R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
PCE	DCIES	—	—	—	—	—	—
15	14	13	12	11	10	9	8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CRES	CRCA	CTES	CTCA	ARES	ARCA	ATES	ATCA
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31	**INTDIS**: Global Interrupt Enable/Disable bit `0` = The $\overline{\text{SINT}}$ pin is enabled and controlled by the unmasked interrupt status bits. `1` = The $\overline{\text{SINT}}$ pin is disabled and driven high.
bit 30-29	**Unimplemented:** Write as ‘0’
bit 28	**DEISM**: DCI Event Interrupt Status Mask `0` = The $\overline{\text{SINT}}$ pin will be asserted when the DCI Event Interrupt Status (**DEIS**) bit is set. `1` = The DCI Event Interrupt Status (**DEIS**) bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 27	**CTISM**: Control Channel Transmit Interrupt Status Mask `0` = The $\overline{\text{SINT}}$ pin will be asserted when the Control Channel Transmit Status (**CTIS**) bit is set. `1` = The Control Channel Transmit Status (**CTIS**) bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 26	**CRISM**: Control Channel Receive Interrupt Status Mask `0` = The $\overline{\text{SINT}}$ pin will be asserted when the Control Channel Receive Status (**CRIS**) bit is set. `1` = The Control Channel Receive Status (**CRIS**) bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 25	**ATISM**: Asynchronous Channel Transmit Interrupt Status Mask `0` = The $\overline{\text{SINT}}$ pin will be asserted when the Asynchronous Channel Transmit Status (**ATIS**) bit is set. `1` = The Asynchronous Channel Transmit Status (**ATIS**) bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 24	**ARISM**: Asynchronous Channel Receive Interrupt Status Mask `0` = The $\overline{\text{SINT}}$ pin will be asserted when the Asynchronous Channel Receive Status (**ARIS**) bit is set. `1` = The Asynchronous Channel Receive Status (**ARIS**) bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 23-21	**Unimplemented:** Write as ‘0’
bit 20	**DEIS**: DCI Event Interrupt Status `0` = The DCI has no status event pending for the EHC to read. `1` = The DCI has a status event pending for the EHC to read.
bit 19	**CTIS**: Control Channel Transmit Interrupt Status `0` = The control packet channel has no packets pending for the EHC to read. `1` = Packets are available for the EHC to read from the control packet channel. The number of packets and total number of bytes available for the EHC to read are specified in the CTXB register.

bit 18	**CRIS**: Control Channel Receive Interrupt Status `0` = The control packet channel cannot receive packets written by the EHC. `1` = Packets may be written by the EHC to the control packet channel. The number of packets and total number of bytes that may be written by the EHC are specified in the CRXB register.
bit 17	**ATIS**: Asynchronous Channel Transmit Interrupt Status `0` = The asynchronous packet channel has no packets pending for the EHC to read. `1` = Packets are available for the EHC to read from the asynchronous packet channel. The number of packets and total number of bytes available for the EHC to read are specified in the ATXB register.
bit 16	**ARIS**: Asynchronous Channel Receive Interrupt Status `0` = The asynchronous packet channel cannot receive packets written by the EHC. `1` = Packets may be written by the EHC to the asynchronous packet channel. The number of packets and total number of bytes that may be written by the EHC are specified in the ARXB register.
bit 15	**PCE**: SPI Protocol Command Error `0` = No errors detected. `1` = An SPI protocol error was encountered. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 14	**DCIES:** DCI Error Status `0` = No errors detected. `1` = An error was detected while the EHC was writing a *DCI Command* or reading a *DCI Response*. Data loss may have occurred. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 13-8	**Unimplemented:** Write as ‘0’
bit 7	**CRES**: Control Receive Error Status `0` = No errors detected. `1` = An error was detected while the EHC was writing a packet to the control packet channel. Data loss may have occurred. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 6	**CRCA**: Control Receive Channel Availability `0` = The control packet channel is not available for the EHC to write packet data. `1` = The control packet channel has been enabled and is available to receive packet data from the EHC.
bit 5	**CTES**: Control Transmit Error Status `0` = No errors detected. `1` = An error was detected while the EHC was reading a packet from the control packet channel. Data loss may have occurred. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 4	**CTCA**: Control Transmit Channel Availability `0` = The control packet channel is not available for the EHC to receive packet data. `1` = The control packet channel has been enabled and is available to transmit packet data to the EHC.
bit 3	**ARES**: Asynchronous Receive Error Status `0` = No errors detected. `1` = An error was detected while the EHC was writing a packet to the asynchronous packet channel. Data loss may have occurred. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 2	**ARCA**: Asynchronous Receive Channel Availability `0` = The asynchronous packet channel is not available for the EHC to write packet data. `1` = The asynchronous packet channel has been enabled and is available to receive packet data from the EHC.
bit 1	**ATES**: Asynchronous Transmit Error Status `0` = No errors detected. `1` = An error was detected while the EHC was reading a packet from the asynchronous packet channel. Data loss may have occurred. This bit is sticky when set and the EHC must write a ‘1’ to clear the error state.
bit 0	**ATCA**: Asynchronous Transmit Channel Availability `0` = The asynchronous packet channel is not available for the EHC to receive packet data. `1` = The asynchronous packet channel has been enabled and is available to transmit packet data to the EHC.

Note: Do not attempt to write to the DBC register.

REGISTER 9-5: DBC: DCI BUFFER CONFIGURATION

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
LEN<7:0>							
31	30	29	28	27	26	25	24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BS<7:0>							
23	22	21	20	19	18	17	16

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
15	14	13	12	11	10	9	8

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24	**LEN<7:0>**: Length of *DCI Command* written or pending *DCI Response*
bit 23-16	**BS<15:0>**: DCI Buffer Size
bit 15-0	**Reserved**: Reserved

REGISTER 9-6: DCTRL: DCI CONTROL

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	ERRM	NTFM	CMDDNM
31	30	29	28	27	26	25	24

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
—	—	—	—	—	ERR	NTF	CMDDN
23	22	21	20	19	18	17	16

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
15	14	13	12	11	10	9	8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	CMDREQ	SYNCREQ
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-27	**Unimplemented:** Write as ‘0’
bit 26	**ERRM**: DCI Error Interrupt Status Mask 0 = The $\overline{\text{SINT}}$ pin will be asserted when the DCI Error (**ERR**) status bit is set. 1 = The DCI Error (**ERR**) status bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 25	**NTFM**: DCI Notification Interrupt Status Mask 0 = The $\overline{\text{SINT}}$ pin will be asserted when the DCI Notification (**NTF**) status bit is set. 1 = The DCI Notification (**NTF**) status bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 24	**CMDDNM**: DCI Command Done Interrupt Status Mask 0 = The $\overline{\text{SINT}}$ pin will be asserted when the DCI Command Done (**CMDCN**) status bit is set. 1 = The DCI Command Done (**CMDDN**) status bit is masked and prevented from asserting $\overline{\text{SINT}}$ when set.
bit 23-19	**Unimplemented:** Write as ‘0’
bit 18	**ERR**: DCI Error Status 0 = No error detected. 1 = A fatal error was detected on the DCI by the INIC. The EHC must write a ‘1’ to clear.
bit 17	**NTF**: DCI Notification Status 0 = The EHC clears this bit by writing a ‘1’ to acknowledge the status change notification. 1 = The INIC sets this bit to notify the EHC of a DCI status change. The EHC should respond by reading the DCI virtual registers.
bit 16	**CMDDN**: DCI Command Done Status 0 = The EHC clears this bit by writing a ‘1’ after reading the *DCI Response*. 1 = The INIC sets this bit to indicate that a *DCI Response* packet is available for reading from the DCI channel by the EHC.
bit 15-2	**Unimplemented:** Write as ‘0’
bit 0	**CMDREQ**: Command Request 0 = No command. The INIC clears this bit when it has executed the *DCI Command*. 1 = The EHC writes a ‘1’ setting this bit to request the INIC to execute a *DCI Command* packet that has been written to the DCI channel.
bit 0	**SYNCREQ**: Driver Synchronization Request 0 = No synchronization requested. The INIC clears this bit. 1 = The EHC writes a ‘1’ setting this bit to request DCI synchronization.

REGISTER 9-7: ARXB: ASYNCHRONOUS PACKET RECEIVE BUFFER STATUS

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXPCNT<15:8>							
31	30	29	28	27	26	25	24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXPCNT<7:0>							
23	22	21	20	19	18	17	16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFREE<15:8>							
15	14	13	12	11	10	9	8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFREE<7:0>							
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **RXPCNT<15:0>**: Number of asynchronous packets available to be written by EHC
bit 15-0 **RXFREE<15:0>**: Number of free buffer bytes available to be written by EHC

REGISTER 9-8: ATXB: ASYNCHRONOUS PACKET TRANSMIT BUFFER STATUS

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXPCNT<15:8>							
31	30	29	28	27	26	25	24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXPCNT<7:0>							
23	22	21	20	19	18	17	16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXAVAIL<15:8>							
15	14	13	12	11	10	9	8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXAVAIL<7:0>							
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **TXPCNT<15:0>**: Number of asynchronous packets available to be read by EHC
bit 15-0 **TXAVAIL<15:0>**: Number of bytes available to be read by EHC

REGISTER 9-9: CRXB: CONTROL PACKET RECEIVE BUFFER STATUS

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXPCNT<15:8>							
31	30	29	28	27	26	25	24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXPCNT<7:0>							
23	22	21	20	19	18	17	16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFREE<15:8>							
15	14	13	12	11	10	9	8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFREE<7:0>							
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **RXPCNT<15:0>**: Number of control packets available to be written by EHC
bit 15-0 **RXFREE<15:0>**: Number of free buffer bytes available to be written by EHC

REGISTER 9-10: CTXB: CONTROL PACKET TRANSMIT BUFFER STATUS

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXPCNT<15:8>							
31	30	29	28	27	26	25	24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXPCNT<7:0>							
23	22	21	20	19	18	17	16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXAVAIL<15:8>							
15	14	13	12	11	10	9	8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TXAVAIL<7:0>							
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16	**TXPCNT<15:0>**: Number of control packets available to be read by EHC
bit 15-0	**TXAVAIL<15:0>**: Number of bytes available to be read by EHC

10.0 EXTERNAL POWER MANAGEMENT

The OS81210 INIC supports external power management logic for controlling Electronic Control Unit (ECU) power by providing two inputs (**PS0** and **PS1**) and one output (**PWROFF**). The **PS0** and **PS1** inputs communicate power status from the ECU external power management logic to the INIC. When a power management event occurs, it should be communicated to the INIC through **PS0** and **PS1** by the external power management circuitry. The functional device behavior that occurs in response to an appropriate external power management status event is described in the INIC Interface Specification [4] and can be configured depending on the application requirements (e.g. OEM specifications). The **PWROFF** output pin is driven low during normal operating conditions and released when the OS81210 INIC is ready to be powered-off by the external power management logic.

If external power management is not used, **PS0** and **PS1** should be pulled to ground (to indicate U_{Normal}) or power management should be disabled within the *INIC Configuration String* (see the INIC Interface Specification [4]). When power management is disabled, the **PS0** and **PS1** pins are available for use as GPIO signals.

Table 10-1 describes the power management states defined by the ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications [1] and ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification [2] that should be communicated to the INIC through **PS0** and **PS1**.

TABLE 10-1: EXTERNAL POWER MANAGEMENT STATES

PS1	**PS0**	**Status**
0	0	U_{Normal}
0	1	Switch-To-Power (STP)
1	0	$U_{Critical}$, U_{Super}
1	1	U_{Low}

The INIC recognizes an external power management state when **PS0** and **PS1** remain at a valid logic level for a minimum time, t_{pshold} (see Section 15.4.2 “Reset and Configuration”); however, the *INIC Software Stack* may use longer hold times for **PS0** and **PS1** states before responding to an external power management event. In addition, the *INIC Software Stack* controls how INIC responds to external power management events (e.g. driving **PWROFF** and/or forcing network shutdown).

A Switch-To-Power (STP) event is recognized as the INIC samples the **PS0** and **PS1** pins at power-up (or reset) and on the *transition* to **PS[1:0]** = 01. If these pins continue to indicate an STP status (**PS[1:0]** = 01) after an STP event has already been recognized, then the U_{Normal} state is assumed.

Figure 10-1 illustrates a typical application in which the external power management circuitry is powered by a continuous supply and manages the switched power to the rest of the node. This circuitry also monitors the activity signal (**BSTATUS**) coming from the OS81210 network activity detector. When network activity is detected, the circuitry enables the switched supply and drives the **PS[1:0]** pins to the proper state, as depicted in Figure 15-1. When switched power is applied to INIC, it could take up to t_{stp} time for INIC to recognize the **PS[1:0]** state and **PWROFF** to become valid. When INIC is ready to be shut down (such as when U_{Low} state is recognized), it releases the **PWROFF** pin (pulled high through an external resistor).

Additionally, **ERR** is driven high and the network output is switched off when the U_{Low} state is recognized. As long as the U_{Low} state is indicated on **PS[1:0]**, the network port outputs remain disabled, regardless of activity on the network port inputs.

If the EHC requires time after power-up (or reset) for initialization, it could utilize a ***PWRHOLD*** line that the power management circuitry detects. The use of ***PWRHOLD*** would suspend node power-down until the EHC is ready.

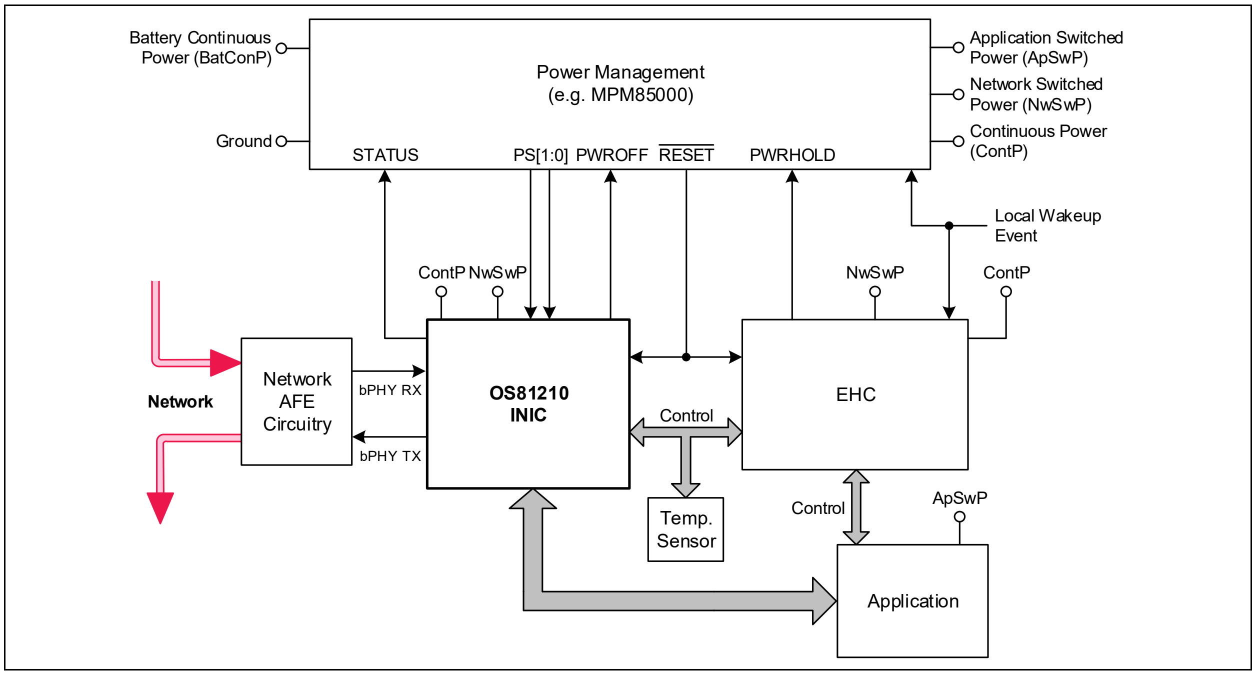
The INIC Interface Specification [4] contains more detailed information on configuring various parameters of the External Power Management function, such as the behavior of **PWROFF**.

INIC supports implementation of a low-power mode for an ECU that requires power conservation while keeping INIC powered and active on the network. A message to enter a low-power mode is initiated from the network and passed by INIC to the EHC. When the EHC receives this message, it may power down certain areas of the ECU and, if supported, the EHC may then enter into a low-power or sleep mode. INIC will wake the EHC by driving $\overline{\textbf{INT}}$ low when any message initiated from the network targets the EHC. This EHC wake feature is only supported when the I^2C port is configured as a slave device and selected as the configuration interface.

The Microchip *Automotive Power Management Device* (MPM85000) performs all power management functionality required of network ECUs including wake detection, power supply monitoring and status (**PS[1:0]**), reset generation, and optional *Switch To Power* (STP) pulse detection.The MPM85000 also supplies a 3.3 V continuous power supply output for the OS81210 **VDDAUn** pins. Integrating both the OS81210 and MPM85000 on a network node minimizes application circuitry and software complexity, allowing the application designer to focus on the application. Contact Microchip for more information on the MPM85000.

Figure 10-1 illustrates the fundamental hardware architecture of a network ECU.

FIGURE 10-1: TYPICAL POWER MANAGEMENT CIRCUITRY



11.0 CLOCK MANAGER

The Clock Manager generates internal clocks and optionally drives an external clock (**RMCK**) on the RMCK Port. Refer to Chapter 12.0 "RMCK Port" for more information on the functionality and configuration of **RMCK**.

11.1 PLL Lock Status

After power-up, the internal analog PLL locks to an external clock reference (**XTI/XTO**) and provides the primary time-base for the OS81210. A digital PLL performs clock and data recovery at the Network Port. For proper data and clock recovery of the incoming network bit stream, a valid and continuous reference clock must always be available.

When network lock is initially achieved, it takes three frames to synchronize streaming data. Therefore, streaming data will not be transferred properly between the network and any open source or sink ports for three frames when going from the unlock to lock state. While the PLL is unlocked, only control messages targeting the local INIC can be communicated with the EHC.

Note: A valid and continuous external reference clock (**XTI/XTO**) must be present for normal operation. If the reference clock is disabled during operation, INIC may become inoperable and/or the state of the **ERR** pin may become invalid. A reset and valid reference clock is required to recover from this condition.

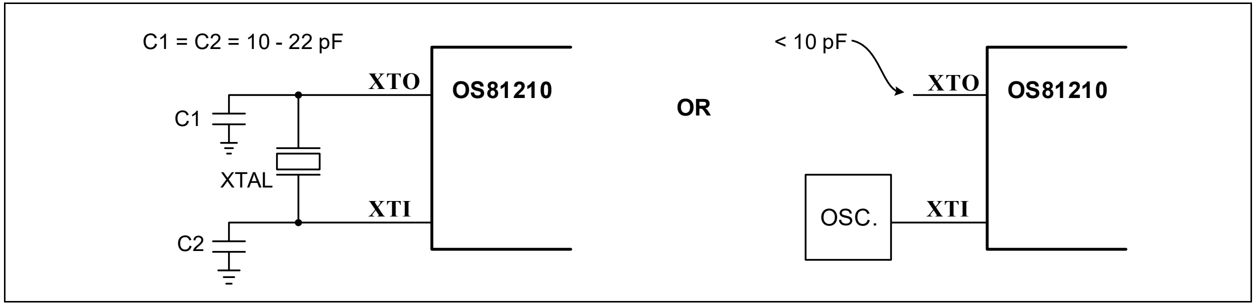
11.2 Crystal Pins (XTI/XTO)

The crystal oscillator should be in a fundamental, parallel resonant mode. Figure 11-1 depicts the external circuitry connected to the OS81210 oscillator circuit. Since the internal inverter/amplifier is operated in its linear region, external series resistors should not be used as they will lower the gain and could cause start-up problems. Several factors must be considered when selecting a crystal including load capacitance, oscillator margin, cut, and operating temperature. For more information on crystals, see Section 16.8 "Crystal Oscillator Selection". The crystal frequency must be 384 times the desired operational network frequency (Fs).

If an external clock is used in lieu of a crystal oscillator, it must be connected to **XTI**. The clock must be stable prior to the rising edge of $\overline{\textbf{RST}}$ and remain stable for proper operation. In addition, **XTO** should float and have minimal capacitance (see Figure 11-1).

Note: In timing-master applications, any jitter present at **XTI/XTO** is transferred through the Clock Manager to the various OS81210 clock outputs (e.g. **RMCK**, **MLBCLK**, **BTXP/BTXN**, etc.). To ensure network physical layer compliance at SP1 and optimal operation of all external application interfaces, jitter at **XTI** must be minimized. A local crystal oscillator (and not an external clock source) is strongly recommended.

FIGURE 11-1: CRYSTAL OSCILLATOR INPUT



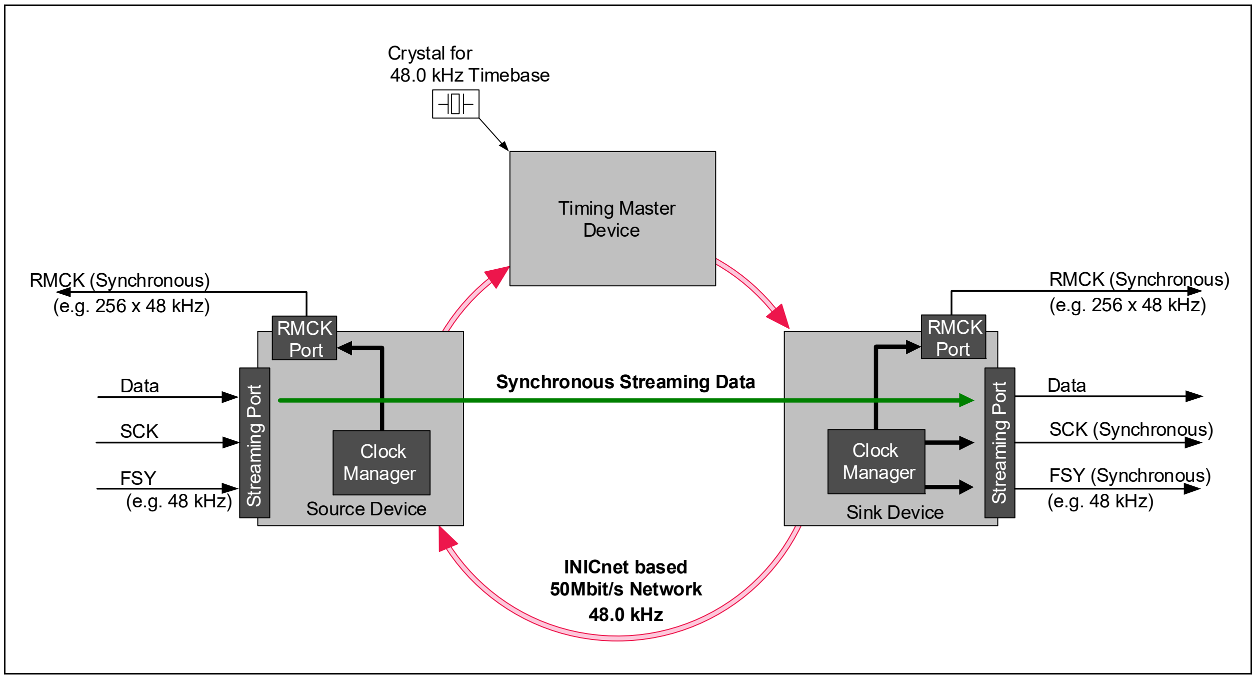
12.0 RMCK PORT

The external interface of the RMCK Port consists of a single pin (**RMCK**) which can be configured as a clock output to support synchronization of external devices to a clock multiple synchronous to the network frame rate. Refer to the INIC Interface Specification [4] for more information on configuring the RMCK Port.

12.1 Synchronous Output Clock

The **RMCK** pin output is derived by dividing an internal 3072×Fs clock that is phase locked to the network frame rate and therefore synchronous to the network. For synchronous streaming data exchanged at the network frequency, the Streaming Port and RMCK Port can be connected to an external device, such as an ADC or DAC, as shown in Figure 12-1.

FIGURE 12-1: RMCK PORT WITH SYNCHRONOUS OUTPUT CLOCK - EXAMPLE



13.0 GENERAL PURPOSE INPUT/OUTPUT (GPIO) PORT

The OS81210 allows certain pins to be reprogrammed from their default functionality to support general purpose input/output (GPIO) functionality. Each **GPn** and its default function when using standard factory firmware is listed in Table 13-1. The user may override the default function via the *INIC Configuration String* and/or custom firmware. See the INIC Interface Specification [4] for more details on the alternate configuration options.

TABLE 13-1: OS81210 GPIO PIN AVAILABILITY

Pin	GPIO	Default Function	Notes
54	GP0	$\overline{\text{INT}}$	The I^{2}C Port is not available in Slave Mode when **GP0** is used.
38	GP1	PS0	The External Power Management functionality is not available when either **GP1** or **GP2** are used.
39	GP2	PS1	
5	GP3	SDIN	The SPI Port is not available when any of **GP3**, **GP4**, **GP5**, **GP6** or **GP7** are used.
6	GP4	SDOUT	
7	GP5	SCLK	
8	GP6	$\overline{\text{SINT}}$	
9	GP7	$\overline{\text{CS}}$	
19	GP8	MUTE	Mute indication for the external application is not available when **GP8** is used.
16	GP10	GP10	Pin dedicated to GP10
40	GP12	GP12	Pin dedicated to GP12
40	GP13	GP13	Pin dedicated to GP13
40	GP14	GP14	Pin dedicated to GP14

Note: The OS81210 $\overline{\textbf{INT}}$ pin is also used as part of the communication interface during the device update process. During *Boot Mode*, this pin is configured as an open-drain output and may be driven low for flow control signaling. Therefore, any external circuitry used for GPIO functionality must be able to withstand INIC actively driving **GP0** low.

13.1 GPIO Classes

GPIO pins are highly configurable which provides the user with the flexibility to support various applications. The following GPIO pin modes are available:

- *Output (Push-Pull)* - High Level or Low Level
- *Output (Open-Drain)* - High Impedance (Hi-Z) or driven low. Trigger: Rising Edge, Falling Edge, High Level, Low Level
- *Input* - Trigger: Rising Edge, Falling Edge, High Level, Low Level
- *Sticky Input* - Trigger: Rising Edge, Falling Edge
- *Debounced Input* - Trigger: Rising Edge, Falling Edge, High Level, Low Level

At OS81210 power-up/reset, all GPIO pins are high-impedance inputs. During GPIO initialization, the user specifies the pin configuration for each GPIO pin that is used and configures any other relevant details, such as debounce settings. Refer to Chapter 15.0 “Electrical Characteristics” for GPIO details such as voltage levels and timing. Further details for each GPIO mode are included in the following subsections and the INIC Interface Specification [4].

13.1.1 PUSH-PULL OUTPUT

A *Push-Pull Output* provides a user-controlled output signal that is actively driven both high and low. The default output state of the pin can be configured as either high or low level.

All GPIO pins are high-impedance inputs following power-up and reset; therefore, an external pull-up resistor is required to hold the pin at a logic high level before the GPIO is configured as a *Push-Pull Output*.

13.1.2 OPEN-DRAIN OUTPUT

An *Open-Drain Output* provides a user-controlled output signal that is actively driven low and set to high-impedance during the logic high state. An external pull-up resistor is required to establish the logic high level when INIC releases the pin. This GPIO mode is useful for ganged signals for which more than one device can drive the line active (low).

Event notification (triggers) can also be configured on outputs similarly to inputs. For *Open-Drain Outputs*, this functionality is also used to notify the higher-level application that another device has driven the line active low.

All GPIO pins are high-impedance inputs following power-up and reset; therefore, an external pull-up resistor is required to hold the pin at a logic high level before the GPIO is configured as an *Open-Drain Output*.

13.1.3 INPUT

A GPIO pin configured as an input can generate a notification (trigger) when a logic level (high-level or low-level) is detected or when a signal transition (edge) is detected.

13.1.4 STICKY INPUT

A *Sticky Input* allows the user to capture very brief pulses that may occur using dedicated hardware mechanisms and is appropriate for high speed inputs where active conditions may only exist for brief periods of time. The captured value is referred to as the sticky state and is independent of the actual current state of the hardware pin. Once the active condition has been captured, the input is ’stuck’ and INIC does not report any further event conditions. Therefore, the captured state must be manually cleared by a higher-level application to re-enable the reporting of event conditions. The main advantage of a *Sticky Input* is the ability to capture narrow pulses that cannot be detected when using other GPIO modes.

Edge-based trigger events on a *Sticky Input* pin remain valid as long as the input signal remains at the active level. Once the event is cleared, INIC will immediately report another event if the active level remains present on the GPIO pin. This behavior is similar to that of *Level-Sensitive Input* GPIO pins; however, in the case of *Sticky Inputs*, this action may result in additional edge-based events being reported when no further edges have actually occurred.

Some considerations must be made with the use of *Sticky Input* pins. One is that the captured state must be manually cleared by a higher-level application in order to re-enable the reporting of subsequent event notifications. Additionally, there are complexities (described above) associated with the handling of an input event that is maintained at the active level rather than appearing as a narrow pulse.

13.1.5 DEBOUNCED INPUT

A *Debounced Input* is an *Input* that does not immediately report an event. Rather, the value on the pin is sampled again after a period of time and compared to the initially sampled value. If the values match, the signal is assumed to be stable (valid), the value is captured, and an event is reported. If the values do not match, the original edge event is considered non-qualified and is ignored.

A *Debounced Input* GPIO is useful for capturing events generated by mechanical switches that normally suffer from contact bounce. The time delay allows the signal to settle before reporting the event thereby only reporting a single, clean event to the monitoring application.

Refer to the INIC Interface Specification [4] for more information on configuring a *Debounced Input*.

13.2 GPIO Triggers

13.2.1 LEVEL-SENSITIVE TRIGGERS

A *Level-Sensitive Trigger* monitors the state of a GPIO pin and generates an event notification when either a high-level or low-level logic is detected. The event notification remains valid as long as the signal stays active (e.g., high or low, depending on configuration).

The *Level-Sensitive Trigger* is appropriate for signals where active levels are maintained until serviced. An advantage of this trigger is that subsequent event triggers are only signaled once the controlling device re-enables event notifications, provided the active level has been maintained. This behavior is desirable in application scenarios where:

- a peripheral attached to the OS81210 GPIO pin does not de-assert its hardware interrupt between events, or
- interrupt lines from multiple hardware peripherals are ganged together (e.g. using open-drain logic) onto a single OS81210 GPIO pin.

When using a *Level-Sensitive Trigger*, a higher-level application must manually clear events in order to re-enable the reporting of subsequent event notifications. Details for re-enabling of subsequent event notifications is found within the INIC Interface Specification [4].

13.2.2 EDGE-SENSITIVE TRIGGERS

An *Edge-Sensitive Trigger* generates an event notification when signal transitions are detected. This type of trigger is appropriate for detecting debounced transition events on an input signal or open-drain output signal in which subsequent signal transitions are immediately reported to the controlling application.

14.0 JTAG PORT

The OS81210 implements a dedicated, 4-pin Test Access Port (TAP) interface (**TDI**, **TDO**, **TCK** and **TMS**), as defined by the IEEE 1149.1 Test Access Port standard. For more information on the IEEE 1149.1 Test Access Port standard, see IEEE Standard Test Access Port and Boundary-Scan Architecture [11].

The test-data input (**TDI**) and test-data output (**TDO**) pins provide the means of shifting data into and out of the OS81210 boundary-scan register (BSR). The TAP interface is controlled by the test clock (**TCK**) and the test-mode select (**TMS**) pins. When **TMS** is low, the OS81210 TAP interface is enabled and the rising edge of **TCK** clocks in data from **TMS** and **TDI**. The TAP state machine is asynchronously reset at power-up and when **TMS** is held high for five rising edges of **TCK**. After reset, the **TDO** pin is high impedance. **TDO** is driven by the TAP controller only when in the *Shift-IR* or *Shift-DR* state.

For proper operation, all OS81210 TAP interface pins should be pulled high through resistors. A 4.7 kΩ pull-up resistor is recommended for **TCK**, **TDI**, and **TDO** per the IEEE 1149.1 standard. A 47 kΩ resistor is recommended for **TMS**.

The OS81210 TAP interface supports all the mandatory boundary-scan instructions, as specified in the IEEE 1149.1 standard. Supported instructions include:

- BYPASS - Selects the bypass register (BYR) to be connected for serial access between **TDI** and **TDO**.
- EXTEST - Selects the boundary-scan resister (BSR) to be connected for serial access between **TDI** and **TDO**.
- SAMPLE/PRELOAD - Captures a snapshot of the data at the digital pins during normal operation, and loads a data pattern into the boundary-scan register (BSR) prior to a new boundary-scan operation.
- IDCODE - (optional IEEE 1149.1 instruction) Used to read the Manufacturer ID from the identification register (IDR).

The OS81210 must be placed in *Boot Mode* before accessing the JTAG Port. In *Boot Mode*, the *INIC Software Stack* is disabled and the OS81210 waits for manual access through the JTAG Port. In this mode, standard communication channels (e.g. I^{2}C Port, MediaLB Port, USB Port) are unavailable. The following sequence should be followed when using the OS81210 JTAG Port:

- Place the OS81210 in *Boot Mode*:
  - Hold the OS81210 in reset (drive $\overline{\textbf{RST}}$ pin low),
  - Hold the $\overline{\textbf{BOOT}}$ pin low,
  - Release the OS81210 from reset ($\overline{\textbf{RST}}$ pin high),
- Perform JTAG operations,
- Release the $\overline{\textbf{BOOT}}$ pin, and
- Reset the OS81210 to resume a normal mode of operation.

Note: The $\overline{\textbf{BOOT}}$ pin may be used in the boundary scan after the configuration pin hold time (t_{cphrs}) given in Section 15.4.2 "Reset and Configuration" is met.

The Boundary Scan Description Language (BSDL) file for the OS81210 is available from Microchip upon request.

14.1 Instruction Register (IR)

The JTAG Instruction Register (IR) is accessed when the TAP controller is in the *Select-IR-scan*, *Capture-IR*, *Shift-IR*, *Exit1-IR*, *Pause-IR*, *Exit2-IR*, or *Update-IR* state.

REGISTER 14-1: IR: JTAG INSTRUCTION REGISTER

R/W-0	R/W-0	R/W-0	R/W-1
IR[3]	IR[2]	IR[1]	IR[0]
3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 3-0 **IR<3:0>**: Instruction Code. Default value = `0001`. Supported instructions include:

`0000` = EXTEST	Enables EXTEST operation
`0001` = SAMPLE/PRELOAD	Enables SAMPLE or PRELOAD operation
`0010` = IDCODE	Enables shifting out of the Manufacturer ID
`1111` = BYPASS	Enables BYPASS
All other codes	Not supported

14.2 Identification Register (IDR)

The JTAG Identification Register (IDR) is accessed when the TAP controller is in the *Select-DR-scan*, *Capture-DR*, *Shift-DR*, *Exit1-DR*, *Pause-DR*, *Exit2-DR*, or *Update-DR* state.

REGISTER 14-2: IDR: JTAG IDENTIFICATION REGISTER

R	R	R	R	R-0	R-0	R-0	R-0
REV[0]	REV[1]	REV[2]	REV[3]	FEAT[0]	FEAT[1]	FEAT[2]	FEAT[3]
31	30	29	28	27	26	25	24

R	R	R	R	R	R	R	R
PID[0]	PID[1]	PID[2]	PID[3]	PID[4]	PID[5]	PID[6]	PID[7]
23	22	21	20	19	18	17	16

r-0	r-0	r-0	r-0	R-0	R-1	R-0	R-0
—	—	—	—	MID[0]	MID[1]	MID[2]	MID[3]
15	14	13	12	11	10	9	8

R-0	R-1	R-0	R-0	R-0	R-1	R-0	R-1
MID[4]	MID[5]	MID[6]	MID[7]	MID[8]	MID[9]	MID[10]	1
7	6	5	4	3	2	1	0

Legend:			
R = Readable bit	W =Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

The 32-bit JTAG IDR register is read out least significant bit first, bit 0 to bit 31. The bits below are listed in reverse order from the way they are read out.

bit 31-28	**REV<3:0>**: Hardware Revision. Default value depends on actual hardware revision. `0001` = Revision A0A `0011` = Revision B1A `0100` = Revision B2B – The data sheet is applicable to this hardware revision.
bit 27-24	**FEAT<3:0>**: Feature Identifier. Default value depends on device grade. `0000` = Grade A (standard)
bit 23-16	**PID<15:0>**: Part Identifier. `30h` = OS81210
bit 15-12	**Reserved**: Read as ‘0000’
bit 11-1	**MID<10:0>**: 11-bit Manufacturer ID. `222h` = Microchip Manufacturer ID
bit 0	Least significant bit (must be set per IEEE 1149.1 specification). Default value = `1`.

15.0 ELECTRICAL CHARACTERISTICS

Specifications are subject to change without notice.

15.1 Absolute Maximum Ratings

TABLE 15-1: ABSOLUTE MAXIMUM RATINGS

Parameter [1]	Min	Max	Unit
Storage Temperature	-65	150	°C
Junction Temperature Under Bias		150	°C
Power Supply Voltage: [2]			
Peripheral (**VDD12**)	-0.2	1.9	V
Core (**VDDCn**)	-0.2	2.5	V
Analog (**VDDA18**)	-0.2	2.5	V
bPHY Analog (**VDDE18**)	-0.2	2.5	V
USB Analog (**VDDUSB18**)	-0.2	2.5	V
Peripheral (**VDDPn**)	-0.2	3.9	V
Analog (**VDDA33**)	-0.2	3.9	V
bPHY Analog (**VDDE33**)	-0.2	3.9	V
USB Analog (**VDDUSB33**)	-0.2	3.9	V
Analog Continuous Power (**VDDAUn**)	-0.2	3.9	V
Power Supply Ramp Rate:			
Analog Continuous Power (**VDDAUn**)	16.67	6000	µs/V
Peripheral (**VDDPn**)	16.67	6000	µs/V
Power Dissipation: [3]		994	mW
DC Current to any Pin Except Power		±10	mA
Maximum Input Voltage:			
STROBE, DATA	-0.5	**VDD12**+0.3	V
DM, DP	-0.5	**VDDPn**+0.3	V
BRXN, **BRXP**	-0.5	**VDDAUn**+0.3	V
XTI	-0.5	**VDDA33**+0.3	V
Other digital pins	-0.5	**VDDPn**+0.3	V

Note 1: Stresses above those listed in this table may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions, above those indicated in the operation listings of this specification, is not implied. Exposure to maximum rating conditions may affect device reliability.

2: The OS81210 power supplies must be sequenced in a way such that the lower voltage supplies never exceed the higher voltage supplies. Specifically, **VDD12** and **VDDCn** must never be more than 0.5 V above **VDDPn**; and **VDDA33**, **VDDE33**, **VDDA18**, and **VDDE18** must never be more than 0.5 V above **VDDAUn**. These requirements apply to power-up, power-down, and normal operation.

3: Maximum power dissipation is calculated using the maximum active mode power supply currents (listed in Section 15.3) and the maximum power supply voltages under operating conditions (listed in Section 15.2 “Operating Conditions”).

15.2 Operating Conditions

TABLE 15-2: OPERATING CONDITIONS

Parameter	Symbol	Min	Max	Unit
Junction Temperature	T_J	-40	125	°C
Power Supply Voltage:				
Peripheral (**VDD12**)		1.10	1.30	V
Core (**VDDCn**)		1.71	1.89	V
Analog (**VDDA18**)		1.71	1.89	V
bPHY Analog (**VDDE18**)		1.71	1.89	V
USB Analog (**VDDUSB18**)		1.71	1.89	V
Peripheral (**VDDPn**)		3.135	3.465	V
Analog (**VDDA33**)		3.135	3.465	V
bPHY Analog (**VDDE33**)		3.135	3.465	V
USB Analog (**VDDUSB33**)		3.135	3.465	V
Analog Continuous Power (**VDDAUn**)		3.135	3.465	V
Voltage applied to pins:				
STROBE, **DATA**		0	1.3	V
DM, **DP**		0	1.6	V
BRXN, **BRXP**		0	**VDDAUn**	V
XTI		0	**VDDA33**	V
Other digital pins		0	**VDDPn**	V
Operating Network Frame Rate[1]	Fs	47.9	48.1	kHz

Note 1: The targeted operating network frame rate (Fs) is 48 kHz.

15.3 DC Characteristics (other than MediaLB Port and USB Physical Interface)

T_J = -40 to 125 °C; VDDCn, VDDA18, VDDE18, VDDUSB18 = 1.8 V ±5 %; VDDPn, VDDA33, VDDE33, VDDAUn, VDDUSB33 = 3.3 V ±5 %; VDD12 = 1.2 V ±0.1 V; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-3: DC CHARACTERISTICS (OTHER THAN MediaLB PORT AND USB PORT)

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Low-Level Input Voltage: **XTI** [1] **STROBE**, **DATA** All other digital pins	V_{IL}			 0.75 0.35×**VDD12** 0.8	 V V V	
High-Level Input Voltage: **XTI** [1] **STROBE**, **DATA** All other digital pins	V_{IH}	 2.0 0.65×**VDD12** 2.0			 V V V	
Input Leakage Current	I_L			±10	µA	0 < Vin < **VDDPn**
Input Hysteresis	ΔV_{hyst}	200			mV	
Low-Level Output Voltage: **STROBE**, **DATA** All other digital pins	V_{OL}			 0.25×**VDD12** 0.4	V	I_{OL} = 4 mA
High-Level Output Voltage: **STROBE**, **DATA** All other digital pins	V_{OH}	 0.75×**VDD12** **VDDPn**-0.4			V	I_{OH} = -4 mA
Digital Input Pin Capacitance				10	pF	
Active Mode Current: [2,3]						
1.2 V Supply Current: Peripheral (**VDD12**)	I_{DDP12}			20	mA	
1.8 V Supply Current: Core (**VDDCn**) Analog (**VDDA18** + **VDDUSB18**) bPHY Analog (**VDDE18**)	 I_{DDC18} I_{ANA18} I_{ANAE18}			 185 82 1	 mA mA mA	
3.3 V Supply Current: Peripheral (**VDDPn**) Analog (**VDDA33** + **VDDUSB33**) bPHY Analog (**VDDE33**) Analog Continuous (**VDDAUn**)[4]	 I_{DDP33} I_{ANA33} I_{ANAE33} I_{VDDAU}			 36 40 6 51	 mA mA mA mA	 outputs unloaded
Sleep Mode Current: [5]						
3.3 V Supply Current: Analog Continuous (**VDDAUn**)	$I_{VDDAU-SL}$			65	µA	

Note 1: Applicable only when an external clock driver is used.

2: The supply current specified for **VDDCn**, **VDDPn**, and **VDDE33** is the sum total of all power pins for the power supply.

3: The maximum current is calculated by loading each pin to the corresponding maximum loading (I_{OL}) and each INIC port is enabled at the following clock speed (MediaLB **MLBCLK** = 1024×Fs, Streaming Port **SCKA** = 512×Fs with all data pins configured as outputs, SPI **SCLK** = 25 MHz, I^2C **SCL** = 400 kHz, **RMCK** = 512×Fs).

4: During a bulk current injection (BCI) event with a common mode swing of 0.9 V_{PK-PK} on the **BRXP** and **BRXN** pins, I_{VDDAU} can increase by 6.4 mA (RMS).

5: Sleep Mode is when **VDDAUn** = 3.3 V and all other power supplies are at 0 V. In this mode, the bPHY physical interface is suspended in a low power state and remains idle until valid network activity is detected.

15.4 AC Characteristics (other than MediaLB Port and USB Physical Interface)

15.4.1 GENERAL SIGNAL AND CLOCKS

T_J = -40 to 125 °C; VDDCn,VDDA18,VDDUSB18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn,VDDUSB33 = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-4: AC CHARACTERISTICS (OTHER THAN MediaLB PORT AND USB PORT)

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Output rise time:	t_{hdr}					(V_{OL} to V_{OH})
ERR, **TDO**				12	ns	
RMCK, **$\overline{\text{SRXA}}$n**, **SRXBn**, **FSYA**, **SCKA**, **MUTE**				5	ns	
SDOUT, **$\overline{\text{SINT}}$**				5	ns	
BSTATUS				12	ns	
GPn				5	ns	
Output fall time:	t_{hdf}					(V_{OH} to V_{OL})
ERR, **PWR$\overline{\text{OFF}}$**				12	ns	
SDA, **SCL**, **$\overline{\text{INT}}$**				12	ns	
DSDA, **DSCL**, **$\overline{\text{DINT}}$**, **TDO**				12	ns	
RMCK, **$\overline{\text{SRXA}}$n**, **SRXBn**, **FSYA**, **SCKA**, **MUTE**				5	ns	
SDOUT, **$\overline{\text{SINT}}$**				5	ns	
BSTATUS				12	ns	
GPn				5	ns	
GPIO Debounce Time Delay	$t_{dbdelay}$	1		65,535	ms	
GPIO Debounce Time Resolution	t_{dbres}		1		ms	
Minimum Pulse Width (Normal Input)	$t_{minpwnrml}$	200			µs	
Minimum Pulse Width (Sticky Input)	$t_{minpwstky}$	60			ns	
Maximum **MUTE** pin delay from event[1]	t_{muteon}			2	ms	
Clocks:						
RMCK Recovered Master Clock Output	f_{rmck}	18,000		1536×Fs	Hz	when enabled
External clock/crystal frequency [2]	f_{osc}		384×Fs		Hz	
External clock/crystal deviation [2]	Δf_{osc}			±200	ppm	

Note 1: The t_{muteon} parameter is the maximum time from an event occurring that can corrupt data to the **MUTE** pin being asserted high.

2: For a 50 Mbit/s network, the external clock/crystal must meet the physical layer specifications for Time Base Deviation ΔFs and network bit rate tolerance within ±200 ppm over all conditions.

15.4.2 RESET AND CONFIGURATION

T_J = -40 to 125 °C; VDDCn,VDDA18,VDDUSB18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn,VDDUSB33 = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-5: RESET AND CONFIGURATION AC CHARACTERISTICS

Parameter	Symbol	Min	Typ	Max	Unit	Comments
PS[1:0] hold time for event detection[1,2,3]	t_{pshold}	2.2			ms	
PS[1:0] event detection to hardware response [2]	t_{pshw}		2	30	ms	
$\overline{\textbf{RST}}$ rising to **PS[1:0]** state recognition [3]	t_{stp}		20	100	ms	
$\overline{\textbf{RST}}$ pulse width [4,6]	t_{rspw}	150			ns	
$\overline{\textbf{BOOT}}$ setup to $\overline{\textbf{RST}}$ rising [4,6]	t_{cpsrs}	0			ns	
$\overline{\textbf{BOOT}}$ hold time from $\overline{\textbf{RST}}$ rising [4,6]	t_{cphrs}	2.5			ms	
$\overline{\textbf{RST}}$ rising to **PWROFF** asserted low	t_{pal}		10	100	µs	
$\overline{\textbf{BRST}}$ hold time after ≥ 90 % VDD [5]	t_{bhold}	1			ms	
$\overline{\textbf{BRST}}$ asserted from ≤ 90 % VDD [5]	t_{bal}	0		100	µs	

Note 1: t_{pshold} is the minimum time required for the INIC to detect an external power management state change at the **PS[1:0]** pins. Events that do not meet this qualification time will be ignored.

2: Once a power state change is detected and qualified (see t_{pshold}), any hardware responses (e.g. disabling the network outputs and releasing the **PWROFF** pin when U_{LOW} is recognized) occur within t_{pshw}. Subsequent power state changes may override pending hardware responses. Therefore, to guarantee that a particular hardware response occurs, t_{pshold} should be at least t_{pshw}(max). Refer to Chapter 10.0 "External Power Management" for more information. The *INIC Software Stack* may take a longer time to report an external power management event to the EHC.

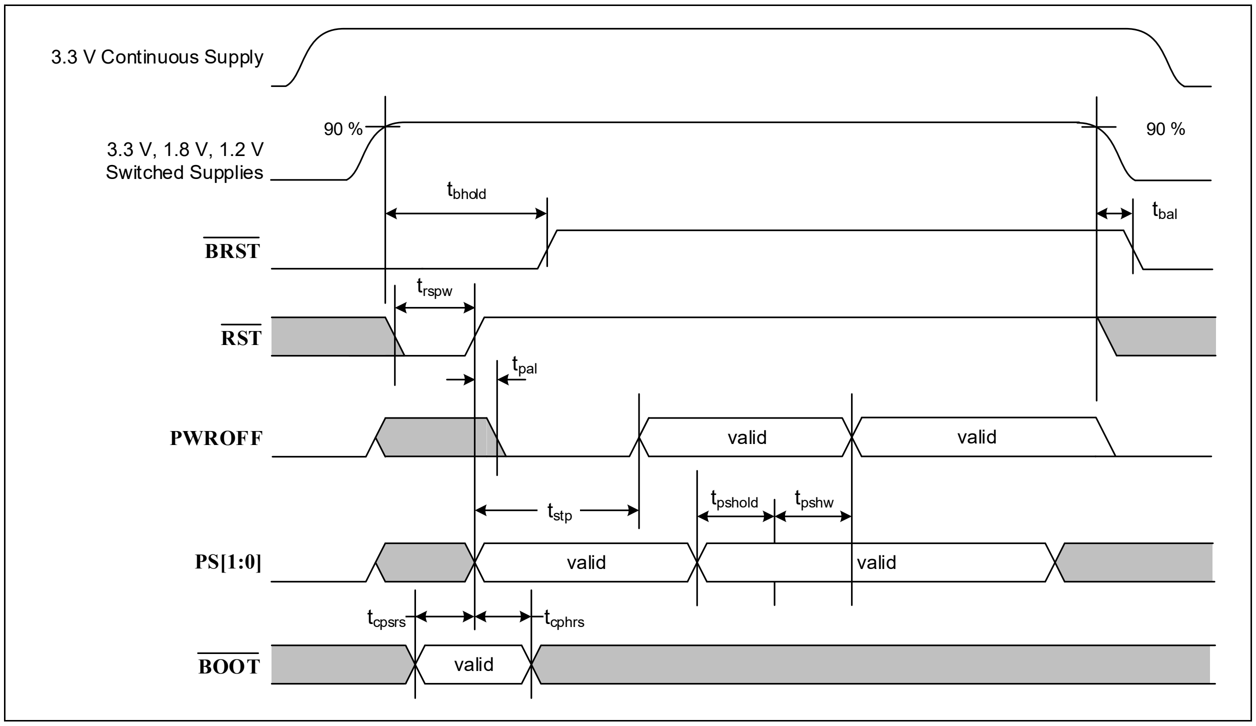
3: Immediately following power-up (or reset), INIC detects and responds to **PS[1:0]** within t_{stp}. Therefore t_{pshold} be at least t_{stp}(max) after power-up (or reset) to guarantee the initial power state detected.

4: After the 3.3 V, 1.8 V, and 1.2 V (when applicable) power supplies have stabilized, INIC must be reset for normal operation. A proper reset is when all supplies are ≥ 90 % VDD.

5: In order to prevent glitches in the network transmit signals during startup and shutdown, the $\overline{\textbf{BRST}}$ pin must be held low for a minimum hold time of t_{bhold} once all supplies are ≥ 90 % VDD, and must be asserted low within t_{bal} after all supplies are ≤ 90 % VDD.

6: The $\overline{\textbf{RST}}$ pulse width indicates the minimum time required to reset the part; however, the $\overline{\textbf{BOOT}}$ pin can take longer to settle to its default state, based on the trace capacitance and size of the on-board pull-up or pull-down resistor. Therefore, the $\overline{\textbf{RST}}$ pulse width must be long enough to allow the external $\overline{\textbf{BOOT}}$ pin to achieve its default state.

FIGURE 15-1: POWER-UP AND RESET TIMING



15.4.3 BALANCED MEDIA PHYSICAL LAYER

T_J = -40 to 125 °C; VDDCn,VDDA18,VDDE18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDE33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-6: ELECTRICAL PHYSICAL LAYER AC CHARACTERISTICS

Parameter	Symbol	Min	Typ	Max	Unit	Comment
$\overline{\text{BRST}}$ rising to *bPHY Tx* output enabled	t_{bphyon}			100	µs	
$\overline{\text{BRST}}$ rising to *bPHY Tx* output disabled	$t_{bphyoff}$			2	µs	
$\overline{\text{RST}}$ rising to *bPHY Tx* output active[1]	$t_{inicinit}$	50		60	ms	
Activity on *bPHY Rx* to *bPHY Tx* output enabled [2]	$t_{NtwStartup}$			6	ms	$\overline{\text{RST}}$ = 3.3 V, $\overline{\text{BRST}}$ = 3.3 V
Loss of activity on *bPHY Rx* to *bPHY Tx* output disabled [3]	$t_{NtwShutdown}$			15	ms	$\overline{\text{RST}}$ = 3.3 V, $\overline{\text{BRST}}$ = 3.3 V
Activity on *bPHY Rx* to BSTATUS falling	t_{statf}	50		700	µs	
Loss of activity on *bPHY Rx* to BSTATUS rising - normal shutdown	t_{statr}			100	µs	
Loss of activity on *bPHY Rx* to BSTATUS rising - Sudden Signal Off [4]	$t_{statrsso}$		180		ms	

Note 1: When INIC's BTXP/BTXN output is enabled, the INIC transceiver becomes *visible* and INIC enters the network (e.g. obtains a node address).

2: The $t_{NtwStartup}$ parameter is only valid if $t_{inicinit}$ has elapsed, otherwise the time from activity on *bPHY Rx* until *bPHY Tx* enabled is determined by $t_{inicinit}$. $t_{NtwStartup}$ also assumes activity has been absent for at least the minimum $t_{Restart}$ time (refer to the *MOST Specification*). The $t_{NtwStartup}$ parameter represents the INIC component of the ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications [1] and ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification [2] t_{WakeUp} parameter.

3: The $t_{NtwShutdown}$ parameter assumes $t_{inicinit}$ has previously expired. $t_{NtwShutdown}$ represents the INIC component of the ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications [1] and ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification [2] $t_{ShutDown}$ parameter.

4: During an unexpected loss of activity on the *bPHY Rx* (such an a sudden-signal-off (SSO) event), the BSTATUS reaction time will increase.

FIGURE 15-2: BALANCED MEDIA PHYSICAL LAYER RESET TIMING

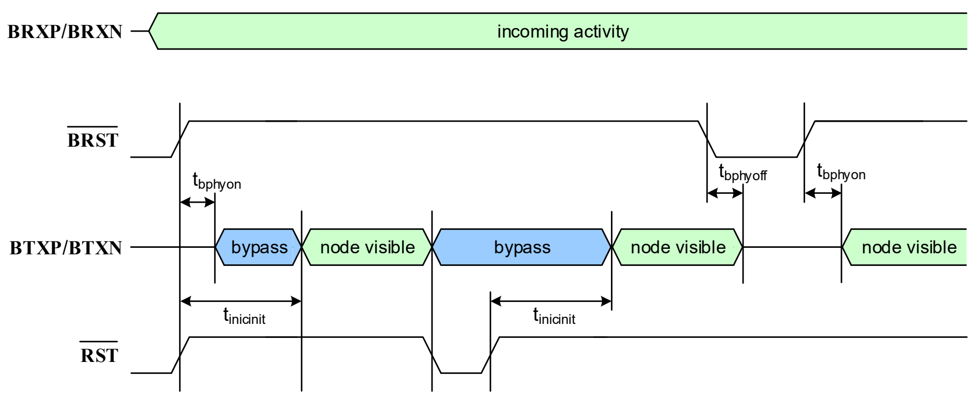


FIGURE 15-3: BALANCED MEDIA PHYSICAL LAYER STARTUP AND SHUTDOWN TIMING

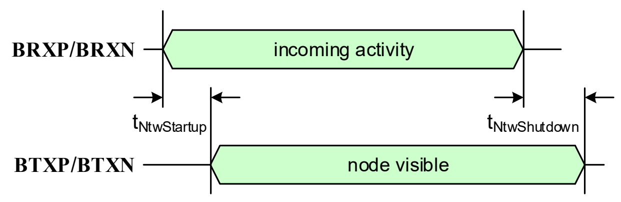
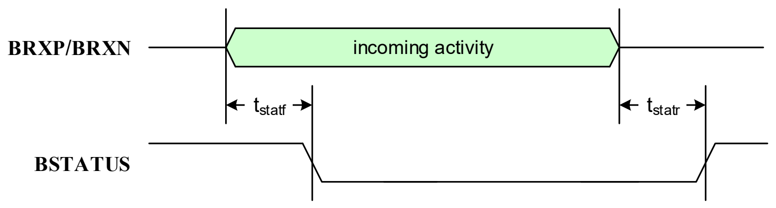


FIGURE 15-4: BALANCED MEDIA PHYSICAL LAYER STATUS TIMING



15.5 Network Characteristics

15.5.1 BALANCED MEDIA PHYSICAL LAYER

T_J = -40 to 125 °C; VDDCn,VDDA18,VDDE18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDE33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-7: ELECTRICAL PHYSICAL LAYER ELECTRICAL CHARACTERISTICS

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Transmitter:						
BTX Common-Mode output voltage	V_{CMT}		1.65		V	See Figure 15-5
Rise time, fall time[1]	t_{rtx},t_{ftx}	4.6		6.9	ns	
Transferred jitter [2]	t_{jtx}			50	ps (RMS)	
RMS signal amplitude	V_{RMSTX}	533		921	mV (RMS)	See Figure 15-5
Eye-mask [3]	See Figure 15-6					
Receiver						
BRX Common-Mode input voltage	V_{CMR}		1.65		V	
Input RMS amplitude for ON State	$V_{RMSRXON}$	110			mV (RMS)	See Note 4
Input RMS amplitude for OFF State	$V_{RMSRXOFF}$			35	mV (RMS)	See Note 5
Eye-mask [3]	See Figure 15-7					

Note 1: Rise and fall time parameters are based upon 10% - 90% signal amplitude limits.

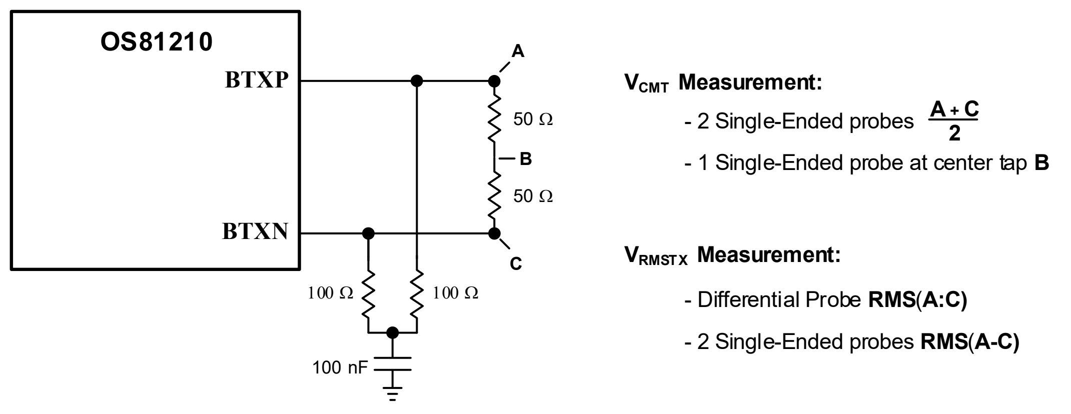
2: Using the jitter filter specified in ISO 21806 (Parts 12-13): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) 50 Mbit/s Balanced Media Physical Layer Specification [12].

3: Using Golden PLL specified in ISO 21806 (Parts 12-13): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) 50 Mbit/s Balanced Media Physical Layer Specification [12].

4: Using the VRMSRXON frequency range (12.288 MHz - 24.576 MHz) specified in ISO 21806 (Parts 12-13): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) 50 Mbit/s Balanced Media Physical Layer Specification [12].

5: Using the VRMSRXOFF frequency range (0 kHz - 10 kHz) specified in ISO 21806 (Parts 12-13): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) 50 Mbit/s Balanced Media Physical Layer Specification [12].

FIGURE 15-5: TRANSMITTER TEST CONDITIONS


OS81210
BTXP
BTXN
A
50 Ω
B
50 Ω
C
100 Ω
100 Ω
100 nF
VCMT Measurement:
- 2 Single-Ended probes (A+C)/2
- 1 Single-Ended probe at center tap B
VRMSTX Measurement:
- Differential Probe RMS(A:C)
- 2 Single-Ended probes RMS(A-C)

15.5.1.1 BTXP/BTXN - Transmit Eye Mask Pattern

The eye mask pattern in Figure 15-6 defines the minimum transmit eye opening area for **BTXP/BTXN**. The OS81210 **BTXP/BTXN** differential signal that is transmitted does not intrude on any of the keep-out areas defined for this eye mask.

FIGURE 15-6: BTXP/BTXN TRANSMIT EYE MASK PATTERN

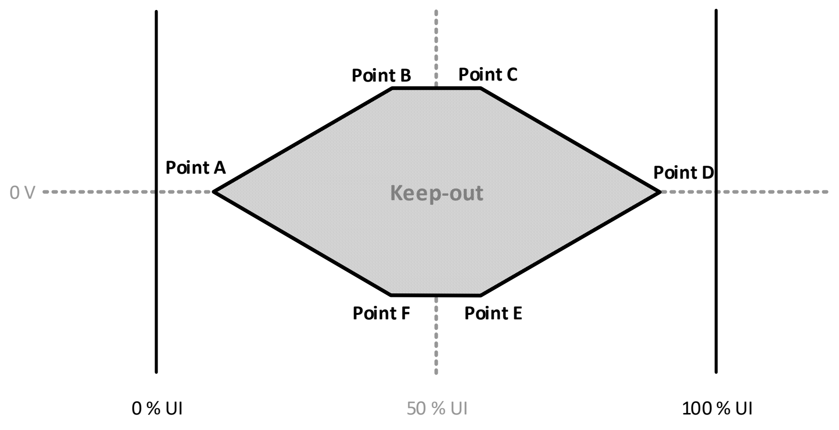


Table 15-8 defines the boundaries of the **BTXP/BTXN** eye mask shown in Figure 15-6.

TABLE 15-8: BTXP/BTXN TRANSMIT EYE MASK BOUNDARIES

Parameter	Voltage	Time
Point A	0.00V	7.5 % UI
Point B	+0.50V	45.0 % UI
Point C	+0.50V	55.0 % UI
Point D	0.00V	92.5 % UI
Point E	-0.50V	55.0 % UI
Point F	-0.50V	45.0 % UI

15.5.1.2 BRXP/BRXN - Receive Eye Mask Pattern

The OS81210 **BRXP/BRXN** receiver will tolerate an input signal that runs up to the keep-out boundaries of the eye mask pattern in Figure 15-7. A differential signal received at **BRXP/BRXN** that fills the eye mask (up to the keep-out boundary) represents the worst-case signal that is tolerable by the OS81210 receiver.

FIGURE 15-7: BRXP/BRXN RECEIVE EYE MASK PATTERN

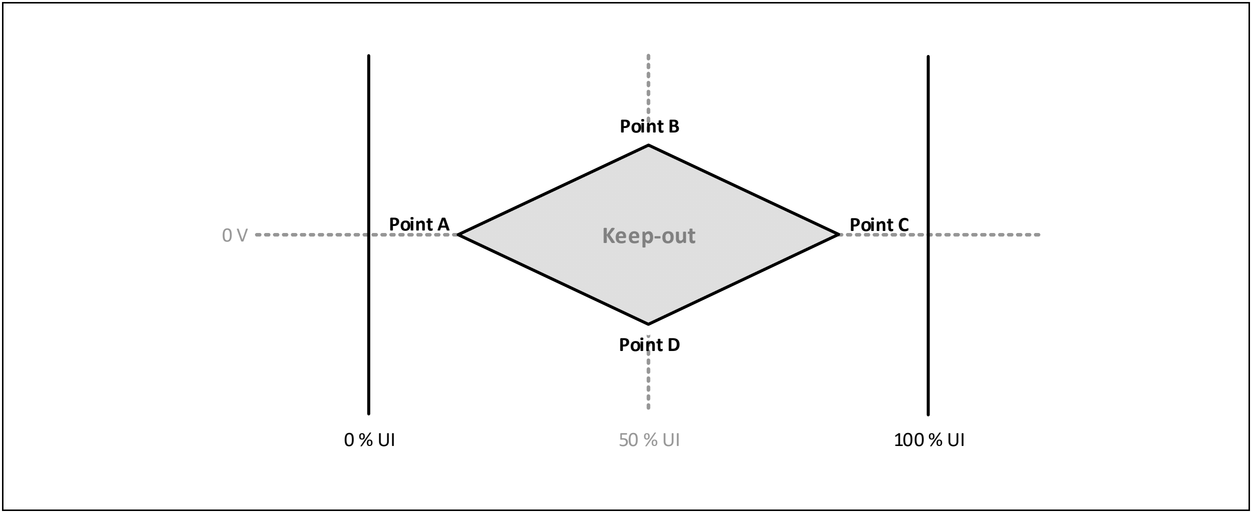


Table 15-9 defines the boundaries of the **BRXP/BRXN** eye mask shown in Figure 15-7.

TABLE 15-9: BRXP/BRXN RECEIVE EYE MASK BOUNDARIES

Parameter	Voltage	Time
Point A	0.00V	30.0 % UI
Point B	+0.050V	50.0 % UI
Point C	0.00V	70.0 % UI
Point D	-0.050V	50 % UI

15.6 MediaLB Port (3-pin)

Note: Refer to the MediaLB Specification [5] for full system-level specifications defining the interaction between the MediaLB Controller (i.e. OS81210) and attached MediaLB Devices.

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 60 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-10: MediaLB PORT (3-PIN 256×FS OR 512×FS)

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Low-level input voltage	V_{IL}			0.7	V	
High-level input voltage	V_{IH}	1.8			V	
Input leakage current	I_L			±1	µA	0 < Vin < VDDPn
Low-level output voltage	V_{OL}			0.4	V	I_{OL} = 8 mA
High-level output voltage	V_{OH}	2.4			V	I_{OH} = -8 mA
MLBCLK operating frequency [1,2]	f_{mck}	12.2624	12.288	12.3136 12.4672	MHz MHz	256×Fs 256×Fs, PLL unlocked
		24.5248	24.576	24.6272 24.9344	MHz MHz	512×Fs 512×Fs, PLL unlocked
MLBCLK rise time	t_{mckr}			3	ns	V_{IL} to V_{IH}
MLBCLK fall time	t_{mckf}			3	ns	V_{IH} to V_{IL}
MLBCLK cycle time	t_{mckc}		81 40		ns ns	256×Fs 512×Fs
MLBCLK low time	t_{mckl}	30 14	35.5 16.5		ns ns	256×Fs 512×Fs
MLBCLK high time	t_{mckh}	30 14	36.5 16.5		ns ns	256×Fs 512×Fs
MLBSIG/MLBDAT receiver input valid to MLBCLK falling	t_{dsmcf}	1			ns	
MLBSIG/MLBDAT receiver input hold from MLBCLK low	t_{dhmcf}	t_{mdzh}			ns	
MLBSIG/MLBDAT output high impedance from MLBCLK low [3]	t_{mcfdz}	0		t_{mckl}	ns	
Bus hold from MLBCLK low [3]	t_{mdzh}	4			ns	
MLBSIG/MLBDAT output valid from MLBCLK high	t_{mcrdv}			8	ns	

Note 1: The OS81210 can shut off MLBCLK to place MediaLB in a low-power state.

2: When the PLL is locked, MLBCLK frequencies are based on the operating network frame rate (Fs) specification given in Section 15.2 "Operating Conditions". During PLL unlock, Fs may drift up to 48.7 kHz.

3: The MediaLB driver can release MLBSIG/MLBDAT (e.g. high impedance) as soon as MLBCLK is low; however, the logic state of the final driven bit on the line must remain on the bus for t_{mdzh}. Therefore, coupling must be minimized while meeting the maximum capacitive load listed.

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 40 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-11: MediaLB PORT (3-PIN 1024×FS)

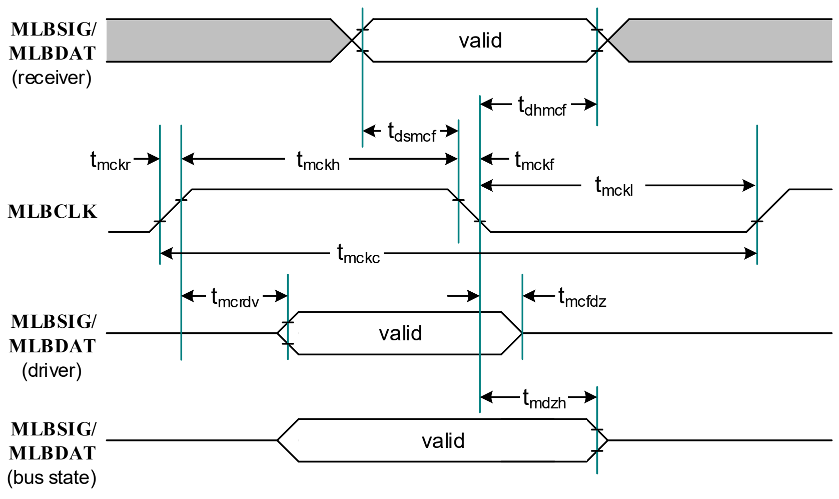
Parameter	Symbol	Min	Typ	Max	Unit	Comments
Low-level input voltage	V_{IL}			0.7	V	
High-level input voltage	V_{IH}	1.8			V	
Input leakage current	I_L			±1	μA	0 < Vin < VDDPn
Low-level output voltage	V_{OL}			0.4	V	I_{OL} = 8 mA
High-level output voltage	V_{OH}	2.4			V	I_{OH} = -8 mA
MLBCLK operating frequency [1,2]	f_{mck}	49.0496	49.152	49.2544 49.8688	MHz MHz	1024×Fs 1024×Fs, PLL unlocked
MLBCLK rise time	t_{mckr}			1	ns	V_{IL} to V_{IH}
MLBCLK fall time	t_{mckf}			1	ns	V_{IH} to V_{IL}
MLBCLK cycle time	t_{mckc}		20.3		ns	
MLBCLK low time	t_{mckl}	6.1	7.3		ns	
MLBCLK high time	t_{mckh}	9.3	10.2		ns	
MLBSIG/MLBDAT receiver input valid to MLBCLK falling	t_{dsmcf}	1			ns	
MLBSIG/MLBDAT receiver input hold from MLBCLK low	t_{dhmcf}	t_{mdzh}			ns	
MLBSIG/MLBDAT output high impedance from MLBCLK low [3]	t_{mcfdz}	0		t_{mckl}	ns	
Bus hold from MLBCLK low [3]	t_{mdzh}	2			ns	
MLBSIG/MLBDAT output valid from MLBCLK high	t_{mcrdv}			7	ns	

Note 1: The OS81210 can shut off MLBCLK to place MediaLB in a low-power state.

2: When the PLL is locked, MLBCLK frequencies are based on the operating network frame rate (Fs) specification given in Section 15.2 “Operating Conditions”. During PLL unlock, Fs may drift up to 48.7 kHz.

3: The MediaLB driver can release MLBSIG/MLBDAT (e.g. high impedance) as soon as MLBCLK is low; however, the logic state of the final driven bit on the line must remain on the bus for t_{mdzh}. Therefore, coupling must be minimized while meeting the maximum capacitive load listed.

FIGURE 15-8: MediaLB 3-PIN TIMING



15.7 Streaming Port

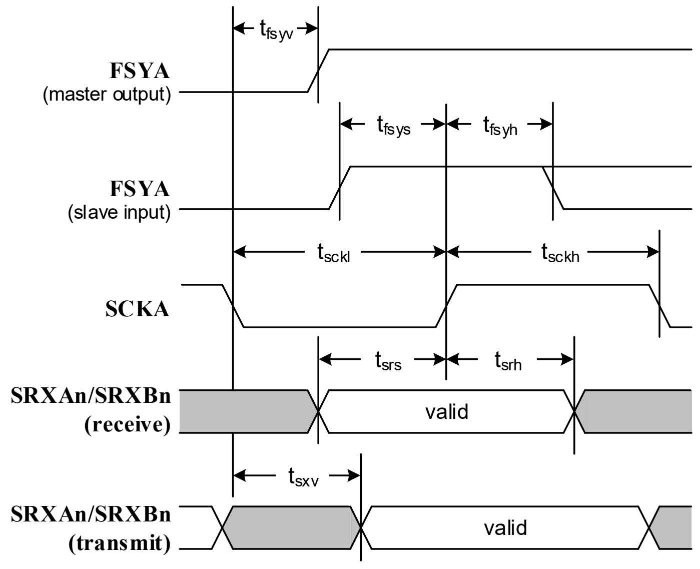
T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF (64×Fs to 256×Fs); Load Capacitance = 20 pF (512×Fs); PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-12: STREAMING PORT

Parameter	Symbol	Min	Typ	Max	Unit	Comments
FSYA frequency [1]	f_{fsy}	47.9	48	48.1	kHz	
SCKA frequency [1]	f_{sck}	64×Fs		512×Fs	Hz	
SRXAn/**SRXBn** valid to **SCKA** rising [2,4]	t_{srs}	2			ns	
SRXAn/**SRXBn** hold from **SCKA** rising [2,4]	t_{srh}	5			ns	
SCK and FSY Outputs (Master Mode):						
SCKA low time	t_{sckl}	50 50 35 18.5			ns ns ns ns	64×Fs 128×Fs 256×Fs 512×Fs
SCKA high time	t_{sckh}	50 50 35 17.5			ns ns ns ns	64×Fs 128×Fs 256×Fs 512×Fs
SCKA falling to **FSYA** valid [2,3]	t_{fsyv}	-5		5	ns	
SRXAn/**SRXBn** valid from **SCKA** falling [2,4]	t_{sxv}	-2		9	ns	
SCK and FSY Inputs (Slave Mode):						
SCKA low time	t_{sckl}	15			ns	
SCKA high time	t_{sckh}	15			ns	
FSYA valid to **SCKA** rising [2,3]	t_{fsys}	5			ns	
FSYA hold from **SCKA** rising [2,3]	t_{fsyh}	5			ns	
SRXAn/**SRXBn** valid from **SCKA** falling [2,4]	t_{sxv}	4		12.5	ns	

Note 1: If **SCKA** and **FSYA** are inputs, they must be frequency locked to the master clock (**RMCK** output clock).
2: The **SCKA** edge at which data is stable and not changing is the rising edge for all supported formats.
3: **FSYA** polarity depends on the data format selected when the port is opened.
4: The MSB of **SRXAn**/**SRXBn** is the first or the second bit after **FSYA** changes, based on the format selected.

FIGURE 15-9: STREAMING PORT TIMING



15.8 USB Port

15.8.1 USB PHYSICAL INTERFACE

Note:	The OS81210 USB physical interface is compatible with other USB 2.0 compliant devices. For more information regarding USB, please refer to the Universal Serial Bus Specification 2.0 [15].

15.8.2 HSIC PHYSICAL INTERFACE

Note:	The OS81210 HSIC physical interface is compatible with other HSIC compliant devices. For more information regarding HSIC, please refer to the High-Speed Inter-Chip USB Electrical Specification 1.0 [16].

15.9 SPI Port

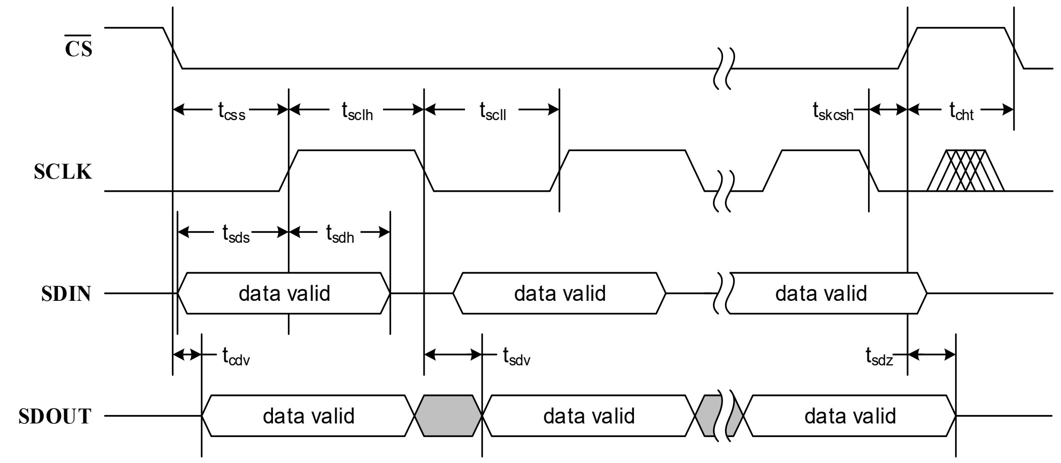
15.9.1 CLOCK MODE 0

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-13: SPI PORT (CLOCK MODE 0)

Parameter	SPI Slave			
	Symbol	**Min**	**Max**	**Unit**
SCLK frequency	f_{scl}		25	MHz
SCLK low time	t_{scll}	15		ns
SCLK high time	t_{sclh}	15		ns
SCLK low to $\overline{\mathbf{CS}}$ high	t_{skcsh}	5		ns
$\overline{\mathbf{CS}}$ low to **SCLK** rising	t_{css}	8		ns
$\overline{\mathbf{CS}}$ low to **SDOUT** valid	t_{cdv}		12	ns
$\overline{\mathbf{CS}}$ high time	t_{cht}	1		µs
SDIN valid to **SCLK** rising	t_{sds}	2		ns
SDIN hold from **SCLK** rising	t_{sdh}	5		ns
SCLK falling to **SDOUT** valid	t_{sdv}		11	ns
$\overline{\mathbf{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}		8	ns

FIGURE 15-10: SPI TIMING (CLOCK MODE 0)



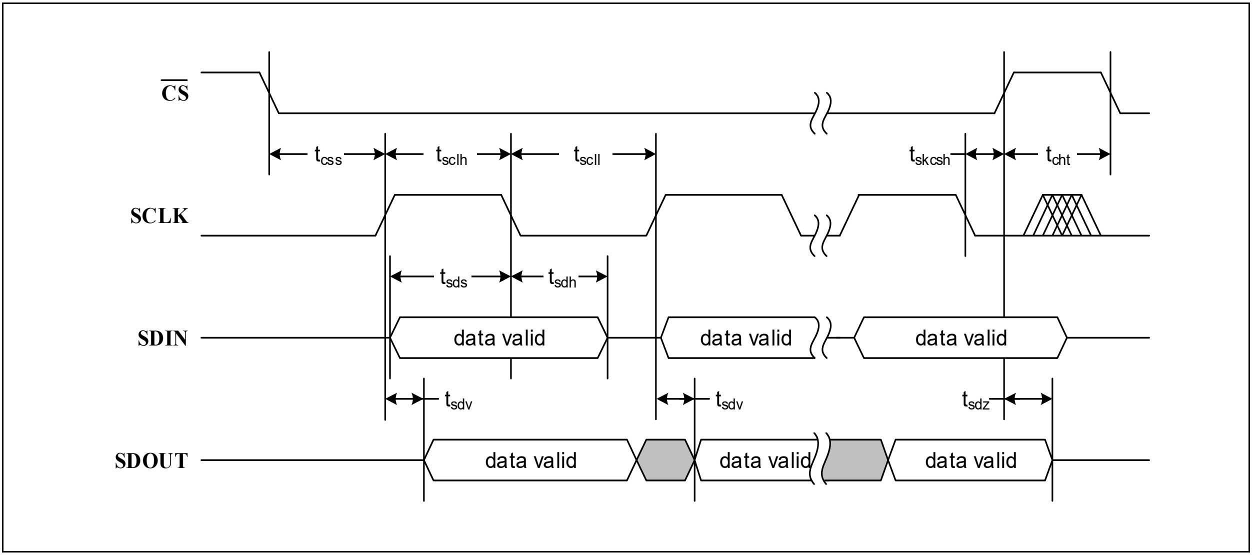
15.9.2 CLOCK MODE 1

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-14: SPI PORT (CLOCK MODE 1)

Parameter	SPI Slave			
	Symbol	**Min**	**Max**	**Unit**
SCLK frequency	f_{scl}		25	MHz
SCLK low time	t_{scll}	15		ns
SCLK high time	t_{sclh}	15		ns
SCLK low to $\overline{\textbf{CS}}$ high	t_{skcsh}	5		ns
$\overline{\textbf{CS}}$ low to **SCLK** rising	t_{css}	8		ns
$\overline{\textbf{CS}}$ high time	t_{cht}	1		µs
SDIN valid to **SCLK** falling	t_{sds}	2		ns
SDIN hold from **SCLK** falling	t_{sdh}	5		ns
SCLK rising to **SDOUT** valid	t_{sdv}		11	ns
$\overline{\textbf{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}		8	ns

FIGURE 15-11: SPI TIMING (CLOCK MODE 1)



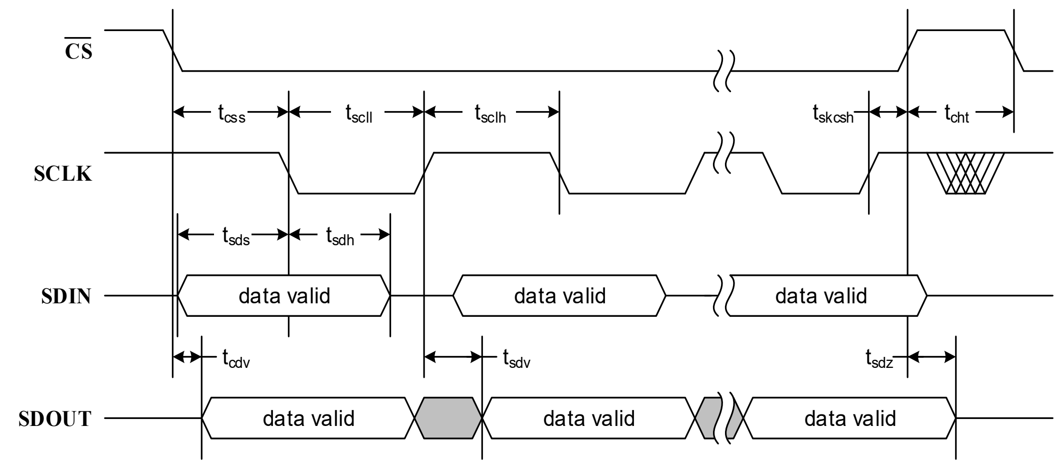
15.9.3 CLOCK MODE 2

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-15: SPI PORT (CLOCK MODE 2)

Parameter	SPI Slave			
	Symbol	**Min**	**Max**	**Unit**
SCLK frequency	f_{scl}		25	MHz
SCLK low time	t_{scll}	15		ns
SCLK high time	t_{sclh}	15		ns
SCLK high to $\overline{\text{CS}}$ high	t_{skcsh}	5		ns
$\overline{\text{CS}}$ low to **SCLK** falling	t_{css}	8		ns
$\overline{\text{CS}}$ low to **SDOUT** valid	t_{cdv}		12	ns
$\overline{\text{CS}}$ high time	t_{cht}	1		µs
SDIN valid to **SCLK** falling	t_{sds}	2		ns
SDIN hold from **SCLK** falling	t_{sdh}	5		ns
SCLK rising to **SDOUT** valid	t_{sdv}		11	ns
$\overline{\text{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}		8	ns

FIGURE 15-12: SPI TIMING (CLOCK MODE 2)



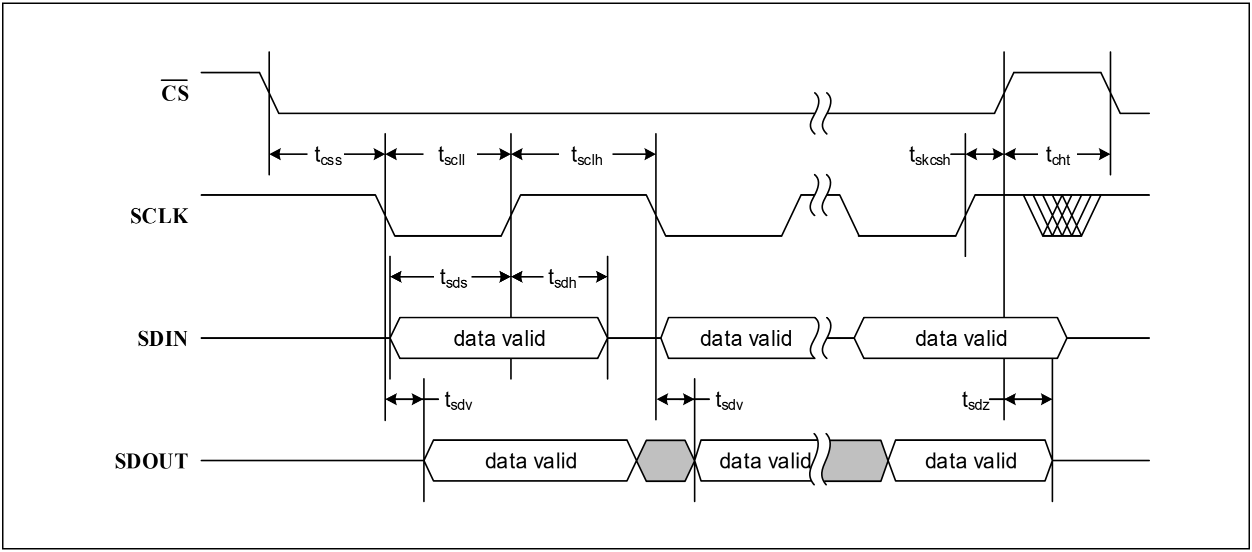
15.9.4 CLOCK MODE 3

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-16: SPI PORT (CLOCK MODE 3)

Parameter	SPI Slave			
	Symbol	**Min**	**Max**	**Unit**
SCLK frequency	f_{scl}		25	MHz
SCLK low time	t_{scll}	15		ns
SCLK high time	t_{sclh}	15		ns
SCLK high to $\overline{\textbf{CS}}$ high	t_{skcsh}	5		ns
$\overline{\textbf{CS}}$ low to **SCLK** falling	t_{css}	8		ns
$\overline{\textbf{CS}}$ high time	t_{cht}	1		µs
SDIN valid to **SCLK** rising	t_{sds}	2		ns
SDIN hold from **SCLK** rising	t_{sdh}	5		ns
SCLK falling to **SDOUT** valid	t_{sdv}		11	ns
$\overline{\textbf{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}		8	ns

FIGURE 15-13: SPI TIMING (CLOCK MODE 3)



15.10 I^{2}C Port

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-17: I^{2}C PORT (SLAVE MODE)

Parameter	Symbol	Min	Max	Unit
Slave Mode:				
SCL frequency	f_{scl}		400	kHz
Bus free between transmissions (SDA high time between stop and start)	t_{buf}	1.3		µs
Start condition hold time (SDA falling to SCL falling)	t_{stah}	0.6		µs
SCL low	t_{scll}	1.3		µs
SCL high	t_{sclh}	0.6		µs
SDA input hold from SCL falling	t_{sdah}	0	900	ns
SDA output valid from SCL falling	t_{ov}		900	ns
SDA input valid to SCL rising	t_{sdas}	100		ns
(repeated) start condition setup time	t_{stas}	0.6		µs
SDA and SCL rise time [1]	t_r		300	ns
SDA and SCL fall time [1]	t_f		300	ns
Stop condition setup time (SCL rising to SDA rising)	t_{stps}	0.6		µs

Note 1: Rise and fall time specifications are based on the V_{IL} and V_{IH} values given in the I2C-Bus Specification [7] and not the V_{IL} and V_{IH} specifications given in Table 15-3.

TABLE 15-18: I^{2}C PORT (MASTER MODE)

Parameter	Symbol	Min	Max	Unit
Master Mode (Standard-mode):				
SCL frequency	f_{scl}		100	kHz
Bus free between transmissions (SDA high time between stop and start)	t_{buf}	4.7		µs
Start condition hold time (SDA falling to SCL falling)	t_{stah}	4.0		µs
SCL low	t_{scll}	4.7		µs
SCL high	t_{sclh}	4.0		µs
SDA input hold from SCL falling	t_{sdah}	0	3.45	µs
SDA output valid from SCL falling	t_{ov}		3.45	µs
SDA input valid to SCL rising	t_{sdas}	250		ns
(repeated) start condition setup time	t_{stas}	4.7		µs
SDA and SCL rise time [1]	t_r		1000	ns
SDA and SCL fall time [1]	t_f		300	ns
Stop condition setup time (SCL rising to SDA rising)	t_{stps}	4.0		µs
Master Mode (Fast-mode):				
SCL frequency	f_{scl}		400	kHz
Bus free between transmissions (SDA high time between stop and start)	t_{buf}	1.3		µs
Start condition hold time (SDA falling to SCL falling)	t_{stah}	0.6		µs
SCL low	t_{scll}	1.3		µs
SCL high	t_{sclh}	0.6		µs

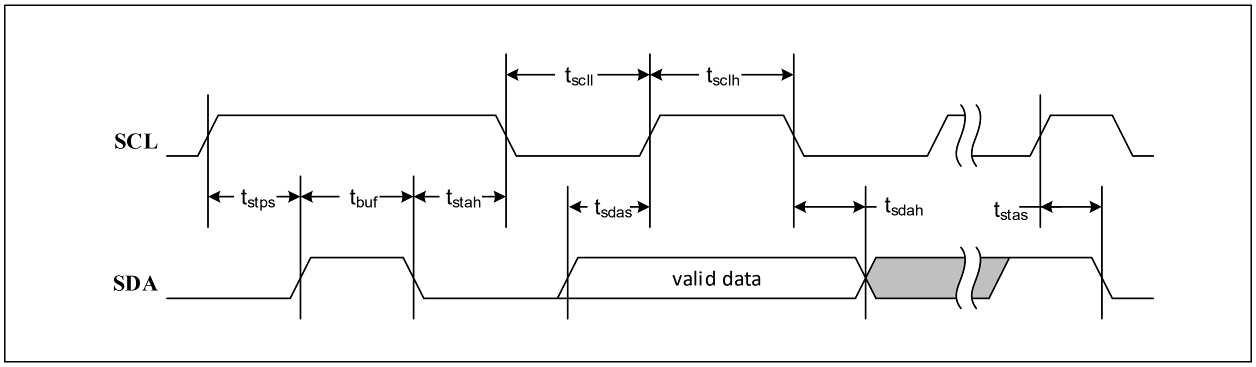
Note 1: Rise and fall time specifications are based on the V_{IL} and V_{IH} values given in the I2C-Bus Specification [7] and not the V_{IL} and V_{IH} specifications given in Table 15-3.

TABLE 15-18: I²C PORT (MASTER MODE) (CONTINUED)

Parameter	Symbol	Min	Max	Unit
SDA input hold from SCL falling	t_{sdah}	0	900	ns
SDA output valid from SCL falling	t_{ov}		900	ns
SDA input valid to SCL rising	t_{sdas}	100		ns
(repeated) start condition setup time	t_{stas}	0.6		µs
SDA and SCL rise time [1]	t_r		300	ns
SDA and SCL fall time [1]	t_f		300	ns
Stop condition setup time (SCL rising to SDA rising)	t_{stps}	0.6		µs

Note 1: Rise and fall time specifications are based on the V_{IL} and V_{IH} values given in the I2C-Bus Specification [7] and not the V_{IL} and V_{IH} specifications given in Table 15-3.

FIGURE 15-14: I²C PORT TIMING



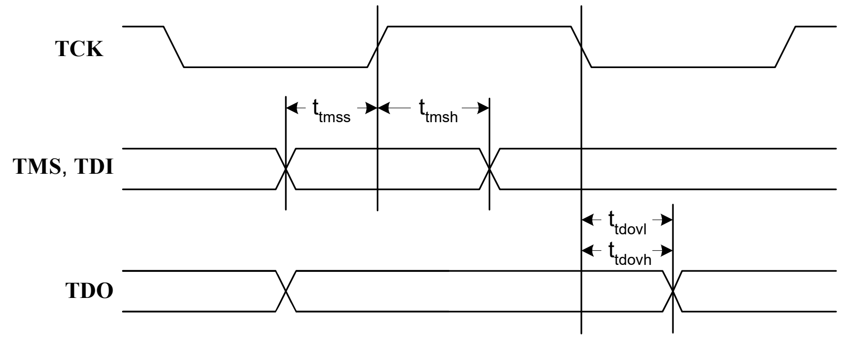
15.11 JTAG Port

T_J = -40 to 125 °C; VDDCn,VDDA18 = 1.8 V ±5 %; VDDPn,VDDA33,VDDAUn = 3.3 V ±5 %; ePAD GND = 0.0 V; Load Capacitance = 30 pF; PLL locked at 48 kHz; unless otherwise noted.

TABLE 15-19: JTAG PORT

Parameter	Symbol	Min	Max	Unit
Maximum **TCK** Frequency	f_{tck}		10	MHz
TDO output valid high from **TCK** falling edge	t_{tdovl}		25	ns
TDO output valid low from **TCK** falling edge	t_{tdovh}		25	ns
TMS and **TDI** input setup to **TCK** rising edge	t_{tmss}	5		ns
TMS and **TDI** input hold from **TCK** rising edge	t_{tmsh}	5		ns

FIGURE 15-15: JTAG TIMING



16.0 APPLICATION INFORMATION

16.1 Power

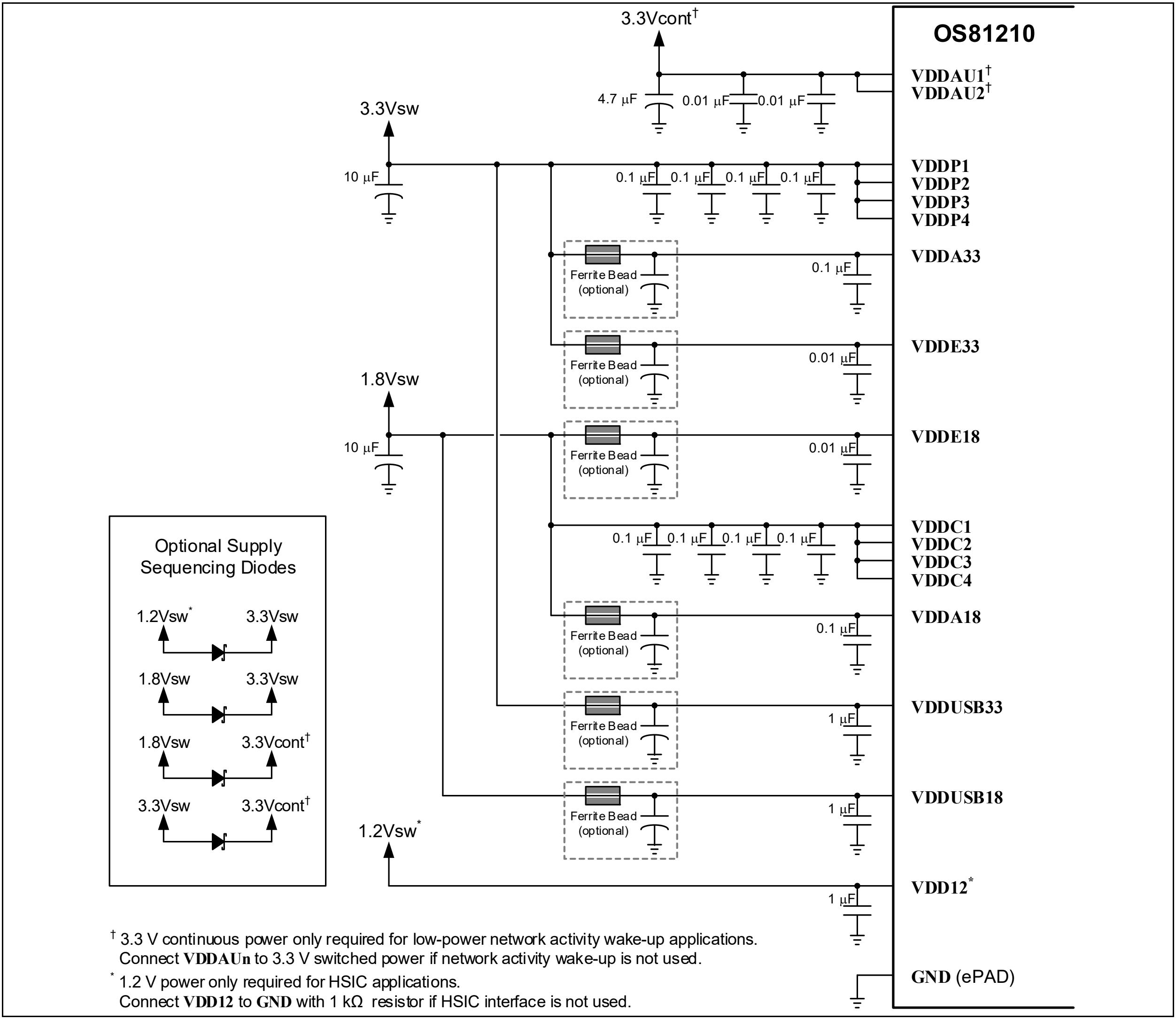
The OS81210 requires 3.3 V power for the **VDDPn**, **VDDA33**, **VDDUSB33**, and **VDDE33** pins and 1.8 V power for the **VDDCn**, **VDDA18**, **VDDE18**, and **VDDUSB18** pins. When the HSIC physical interface is used, 1.2 V power is required for the **VDD12** pin. Additionally, for INIC to support network activity detection while in a low power mode, a continuous 3.3 V power supply should be connected to **VDDAUn** to power the internal bPHY activity detector.

Typical network ECUs support a *Sleep Power State* where continuous power is always connected to the ECU to allow the device to wake the ECU due to network activity. For networks requiring network activity detection, **VDDAUn** should be connected to 3.3 V continuous power to support the *Sleep Power State*, as shown in Figure 16-1. When using the OS81210 in a network ECU that does not implement a *Sleep Power State* and wake from network activity, **VDDAUn** should be connected to 3.3 V switched power. In this case, the INIC does not require 3.3 V continuous power.

The 1.8 V switched power supply must never exceed the 3.3 V switched supply by more than 0.5 V. When a continuous 3.3 V supply is used to implement a *Sleep Power State*, the 1.8 V and 3.3 V switched supplies must never exceed the 3.3 V continuous power supply by more than 0.5 V. When used, the HSIC 1.2 V power supply must never exceed the 3.3 V switched supply by more than 0.5 V. Furthermore, digital input pins must never be driven to more than the **VDDPn** supply, and analog input pins should never be driven to more than the **VDDAUn** supply. These requirements are applicable to power-up and power-down, as well as normal operational conditions.

Figure 16-1 illustrates a typical power arrangement for the OS81210 with the power supply architecture and decoupling required to minimize jitter effects.

FIGURE 16-1: POWER SUPPLY CONNECTION DIAGRAM


3.3Vcont†
OS81210
VDDAU1†
VDDAU2†
4.7 µF
0.01 µF
0.01 µF
3.3Vsw
10 µF
0.1 µF
0.1 µF
0.1 µF
0.1 µF
VDDP1
VDDP2
VDDP3
VDDP4
Ferrite Bead (optional)
0.1 µF
VDDA33
0.01 µF
VDDE33
1.8Vsw
10 µF
0.01 µF
VDDE18
VDDC1
VDDC2
VDDC3
VDDC4
0.1 µF
VDDA18
1 µF
VDDUSB33
1 µF
VDDUSB18
1.2Vsw*
1 µF
VDD12*
GND (ePAD)
Optional Supply Sequencing Diodes
1.2Vsw*
3.3Vsw
1.8Vsw
3.3Vsw
1.8Vsw
3.3Vcont†
3.3Vsw
3.3Vcont†
† 3.3 V continuous power only required for low-power network activity wake-up applications.
Connect VDDAUn to 3.3 V switched power if network activity wake-up is not used.
* 1.2 V power only required for HSIC applications.
Connect VDD12 to GND with 1 kΩ resistor if HSIC interface is not used.

16.2 Optional Components

Figure 16-1 shows Schottky diodes connected between the various power supply rails and ferrite beads on the analog power supply pins. The use of these components is optional and should be considered on a case-by-case basis depending on application requirements and limitations.

As noted in Section 16.1 "Power", the 1.8 V switched supply must never exceed the 3.3 V switched supply by more than 0.5 V. The1.8 V and 3.3 V switched supplies must also never exceed the 3.3 V continuous supply by more than 0.5 V in applications implementing a *Sleep Power State*. Furthermore, in HSIC applications, the 1.2 V supply must never exceed the 3.3 V switched supply by more than 0.5 V. One approach to satisfying these power sequencing requirements is to connect a Schottky diode between the supplies to prevent the lower voltage supply from exceeding the higher supply by more than a forward diode drop. See Figure 16-1. If used, these diodes must be sized appropriately; the forward voltage drop must be less than 0.5 V when the diode conducts current to protect the OS81210 in a power supply fault condition.

Ferrite beads may optionally be added to **VDDA18**, **VDDA33**, **VDDE18**, **VDDE33**, **VDDUSB33**, and **VDDUSB18** for increased noise immunity in EMI-sensitive applications. Depending on the properties of the ferrite bead, their combination with the decoupling capacitors may cause a resonance peaking at lower frequencies leading to an undesired amplification of low-frequency noise in the system resulting in increased jitter and electromagnetic radiation. Since the ferrite bead selection is highly dependent on the noise present in the system, which varies from design to design, the addition of a large bulk capacitor, typically 10 µF, is recommended to be placed on the INIC side of the ferrite bead. See Figure 16-1. During the prototype phase, it is recommended to include the option for the bulk capacitors at the ferrite beads should the need arise to populate them to reduce the effects of resonance peaking in the circuit.

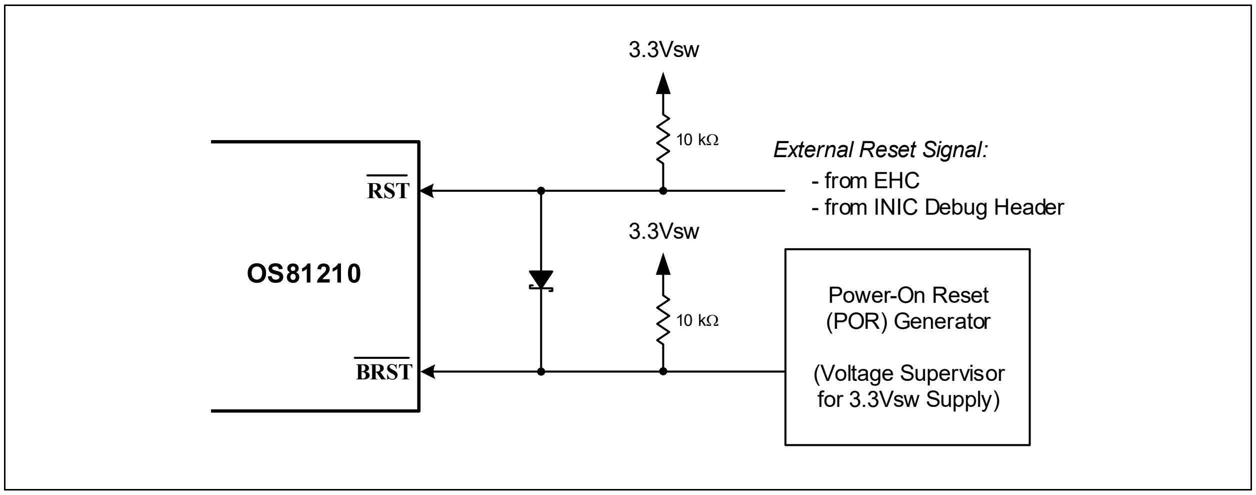
16.3 Reset

Once all power supplies have stabilized, the OS81210 must be reset ($\overline{\textbf{RST}}$ low) for normal operation. A proper reset is when all supplies are ≥90 % of their target voltage; however, electrical specifications are applicable only when supplies are within ±5 % of their target voltage. Once initialized, INIC is typically reset by the EHC only when a fatal error occurs, or when loading the *INIC Configuration String* (see the INIC Interface Specification [4]).

If the external Power-On-Reset (POR) circuit does not drive the $\overline{\textbf{RST}}$ line high (e.g. open-drain output), an external pull-up resistor to 3.3 V should be used. If the external POR circuit drives the $\overline{\textbf{RST}}$ line high and low (e.g. push-pull output), a series resistor should be used between the POR circuit and the OS81210 $\overline{\textbf{RST}}$ pin in lieu of the pull-up. This series resistor allows INIC to be reset by something other than the POR circuit, such as the EHC or the INICkit Tool [3]. The pull-up or series resistor should be at least 10 kΩ and the $\overline{\textbf{RST}}$ pulse width should be long enough to allow the **BOOT** configuration pin to achieve its default state (see Section 15.4.2 "Reset and Configuration").

Diode isolation is recommended on the OS81210 $\overline{\textbf{RST}}$ line. An example reset circuit is shown in Figure 16-2. The POR from the reset generator influences the INIC $\overline{\textbf{RST}}$ pin; however, the Schottky diode ensures that an application reset (i.e. from the EHC) does not disturb the physical layer. The physical layer reset signal acts as a transmitter enable/disable signal that ensures glitch-free operation at the physical layer output during power supply ramp up/down. The physical layer reset signal must be connected to the OS81210 $\overline{\textbf{BRST}}$ pin.

FIGURE 16-2: RESET ARCHITECTURE



16.4 Physical Layer

The INIC is used in 50 Mbit/s balanced media physical layer (bPHY) network applications. The integrated OS81210 bPHY Network Port supports the electrical transmission of a network bit-stream between two or more compatible network devices using low-cost unshielded twisted pair (UTP) wire.

The bPHY interface consists of:

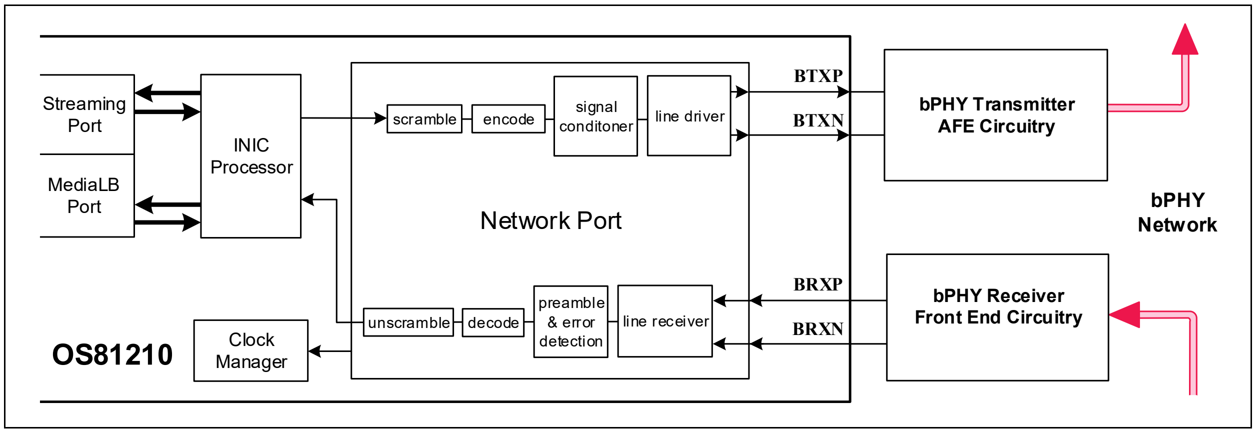
- **BTXP** - positive transmitter output
- **BTXN** - negative transmitter output
- **BRXP** - positive receiver input
- **BRXN** - negative receiver input
- $\overline{\textbf{BRST}}$ - network reset input
- **BSTATUS** - network activity status output

Figure 16-3 illustrates the connection between the INIC Network Port and the bPHY network. A large portion of the transmitter and receiver front end circuitry is used to reduce electromagnetic emissions and increase the overall system immunity to EMI. In order for data transmission to be reliable over a wide operating rage, all signals along the transmission path need to be relatively noise free, with minimal jitter and pulse width distortion. As a result, both the transmitter and receiver front end circuits are designed for reducing electrical emissions, increasing system immunity to EMI, and minimizing reflections.

When $\overline{\textbf{BRST}}$ is driven low, the transmitter output is disabled. $\overline{\textbf{BRST}}$ should only be asserted by the power-on-reset generator and should not be connected directly to the INIC $\overline{\textbf{RST}}$ pin. Refer to Figure 16-2 for the recommended reset architecture. **BSTATUS** will be driven low when a valid network signal is detected and driven high when a qualified signal is not present.

Note:	Contact Microchip for the OS8121x Physical Layer Reference Design Application Note [13] which describes the Analog Front-End (AFE) circuitry implementation in detail.

FIGURE 16-3: bPHY INTERFACE - BLOCK DIAGRAM



When transmitting data onto the network, the Network Port accepts data bytes from the INIC Processor, scrambles the data to reduce data-dependent jitter, conditions the signal for transmission over UTP, and transmits the encoded bit-stream onto **BTXP/BTXN**. These pins are driven by an internal digital-to-analog converter, which drives a low-pass filtered version of the bit-stream by sourcing current into external load resistors. The **BTXP/BTXN** pins interface with the transmitter front end circuitry which provides an analog low-pass filter.

When the OS81210 receives the serial network bit-stream at **BRXP/BRXN**, it converts the differential input to a single-ended signal. The Network Port detects the preamble in the received bit-stream, detects transmission errors, and works in conjunction with the Clock Manager to recover the clock from the bit-stream. The Network Port then decodes and unscrambles the bit-stream, and sends the network data to the INIC Processor as data bytes.

When the OS81210 is configured as a timing-slave, the Network Port is synchronized to the bit-stream recovered clock. When configured as the timing-master, the Network Port controls network timing, based on the external clock input. The transmitted network frames are aligned to this clock.

16.5 Other Application Pins

For normal operation, the $\overline{\textbf{BOOT}}$ pin should be pulled to 3.3 V. To support the *Customer Configuration Interface*, the on-board pull-up resistor on $\overline{\textbf{BOOT}}$ must not be less than 10 kΩ; however, the actual resistor value should be chosen based on the input leakage of the INIC pin and on the other devices connected to the line. The JTAG Port pins **TMS**, **TCK**, **TDI**, and **TDO** require pull-up resistors to 3.3 V, regardless of whether or not the JTAG interface is used.

During the INIC initialization time, $t_{inicinit}$, the **ERR** pin becomes active. The **ERR** pin is driven high to indicate a Network Unlock, which occurs when two consecutive frame preambles are not received. At all other times, the **ERR** pin is driven low to indicate Network Lock. Network Lock occurs when three consecutive preambles are received with the correct timing. In a typical application, the **ERR** pin is used to control an LED to indicate Network Lock.

The **MUTE** pin is an optional output that should be pulled high by an on-board resistor (10 kΩ recommended) when used. If an event occurs that can corrupt the data for any registered synchronous streaming sink connection, the **MUTE** pin is driven low to inform external devices when muting is required. Refer to the INIC Interface Specification [4] for more information.

When an external crystal is used, the **XTI/XTO** pins should be connected to the crystal with no series resistor placed between the pins. The crystal used must be 384×Fs and comply with the specifications outlined in Section 16.8 “Crystal Oscillator Selection”. If an external clock is used in lieu of a crystal oscillator, it must be connected to **XTI** and support the drive levels listed in Chapter 15.0 “Electrical Characteristics” on page 71. In this scenario, **XTO** should float and have a stub capacitance less than 10 pF.

Jitter at **XTI** must be minimized in all applications. Driving an external clock signal is more prone to noise coupling events (e.g. crosstalk); therefore, a local crystal oscillator is the preferred method of providing the INIC clock reference. This directive is especially relevant when INIC operates as the network timing-master.

When external power management is used, the **PWROFF** pin should be pulled high through an external resistor. After initialization, INIC drives **PWROFF** low during normal operating conditions and releases **PWROFF** when the INIC Processor is ready to be shutdown (due to network inactivity or **PS[1:0]** = U_{Low}). Refer to Chapter 10.0 “External Power Management” on page 61 for more information.

16.6 Configuration and Debug

16.6.1 CONFIGURATION STRING

The OS81210 provides *One-Time Programmable* (OTP) memory that is used for storing power-up configuration settings. OTP memory is divided into two sections, which allows configuration settings to be written twice. When OTP memory is written once, the first section stores the configuration data and the second section is unused. When OTP memory is written again, the second section stores the configuration data and all data previously written into the first section is ignored. After power-up/reset, configuration settings are loaded from the appropriate OTP section into RAM.

The initial power-up configuration settings stored in OTP memory are collectively referred to as the *INIC Configuration String*. The *INIC Configuration String* defines the port(s) which will be automatically opened and includes initial hardware configuration information such as:

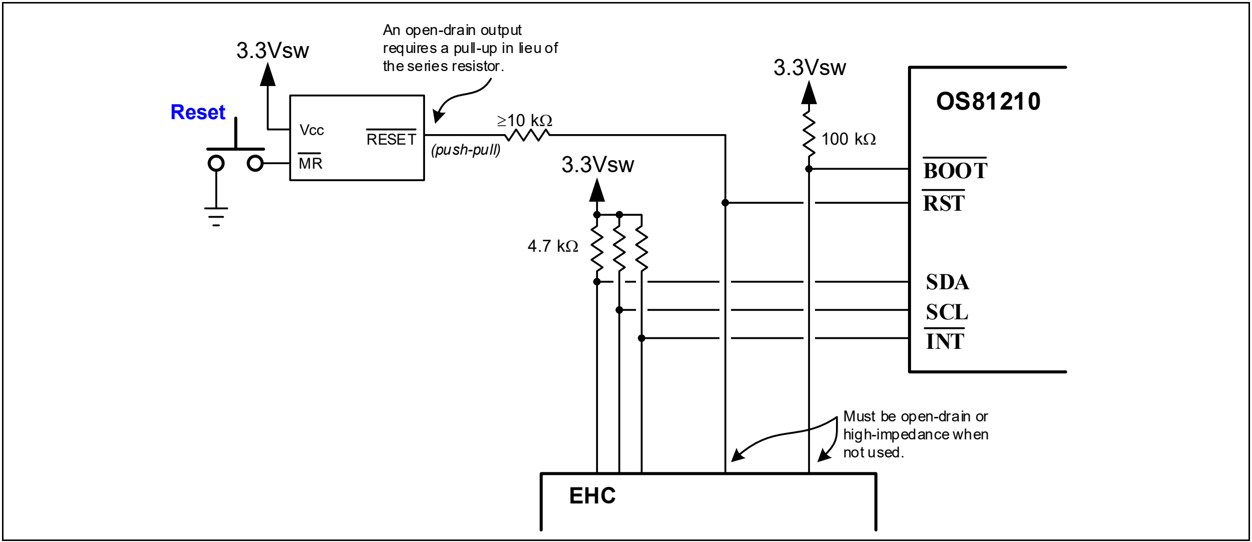
- I^2C Port address and mode,
- Default port for INIC-EHC communication,
- MediaLB Port speed,
- USB Port physical layer interface,
- **RMCK** configuration,
- GPIO usage, and
- Network device mode (timing-master/timing-slave).

If OTP memory is not initialized, factory default settings for the *INIC Configuration String* included in the INIC firmware are used in its place. The OTP memory only needs to be programmed when the factory defaults are not adequate for the application. Refer to the INIC Interface Specification [4] for more information on the factory default values and customization of the *INIC Configuration String*.

16.6.2 EHC VIA I^{2}C PORT

The I^{2}C Port can be used by the EHC to program the *INIC Configuration String*. This approach is suitable for production-level applications. To support this feature, the EHC must be connected to the INIC I^{2}C Port (**SDA**, **SCL**, and $\overline{\textbf{INT}}$) and be capable of controlling the $\overline{\textbf{BOOT}}$ and $\overline{\textbf{RST}}$ pins. The required connections are shown in Figure 16-4.

FIGURE 16-4: CONNECTION DIAGRAM FOR I^{2}C PORT INTERFACE



3.3Vsw
An open-drain output requires a pull-up in lieu of the series resistor.
Reset
Vcc
RESET
MR
≥10 kΩ
(push-pull)
3.3Vsw
3.3Vsw
100 kΩ
4.7 kΩ
OS81210
BOOT
RST
SDA
SCL
INT
Must be open-drain or high-impedance when not used.
EHC

The sequence required of the EHC to program the OS81210 INIC is as follows:

- Hold the OS81210 in reset (drive $\overline{\textbf{RST}}$ pin low),
- Hold the $\overline{\textbf{BOOT}}$ pin low,
- Release the OS81210 from reset ($\overline{\textbf{RST}}$ pin high),
- Erase and program OS81210 INIC, as described in the Device Update Process User's Guide [14],
- Release the $\overline{\textbf{BOOT}}$ pin, and
- Reset the OS81210 to resume a normal mode of operation.

16.6.3 INICkit VIA JTAG PORT

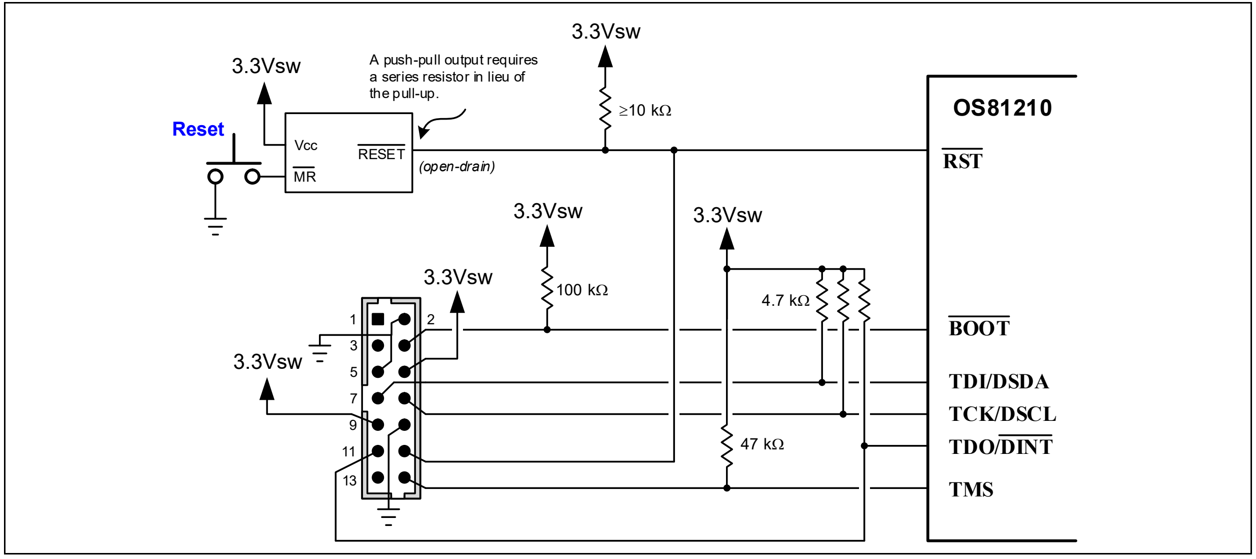
The OS81210 JTAG Port can be converted into a configuration interface for use with the Microchip INICkit Tool [3]. *INICkit* requires that the application PCB provide a standard 14-pin (2x7) 2 mm header (such as Molex 87332-1420 or equivalent). The interface between the *INICkit tool* and the OS81210 is also referred to as the *Customer Configuration Interface*.

In addition to writing the *INIC Configuration String* into OTP memory, *INICkit* provides debugging capabilities by allowing users access to INIC (via PC software) for:

- Exploring internal INIC properties and states while it is operating in the target platform,
- Viewing internal INIC states and routing configurations graphically, and
- Creating an INIC data memory snapshot as a file dump.

Figure 16-5 illustrates the necessary connections between the OS81210 and the configuration/debug header that accommodates *INICkit*. To allow the header to drive the $\overline{\textbf{BOOT}}$ and $\overline{\textbf{RST}}$ pins during configuration, the pull-up resistors on the lines should be greater than 10 kΩ. If the $\overline{\textbf{BOOT}}$ and $\overline{\textbf{RST}}$ lines are also connected to the EHC (as described in Section 16.6.2 "EHC via I2C Port"), the EHC must use open-drain logic or set its outputs as high-impedance when the *INICkit tool* connects to the OS81210 via the configuration/debug header. It is strongly recommended to include the footprint for the configuration/debug header even if the header is not populated.

FIGURE 16-5: CONFIGURATION/DEBUG HEADER



Contact Microchip for more information on the INICkit Tool [3].

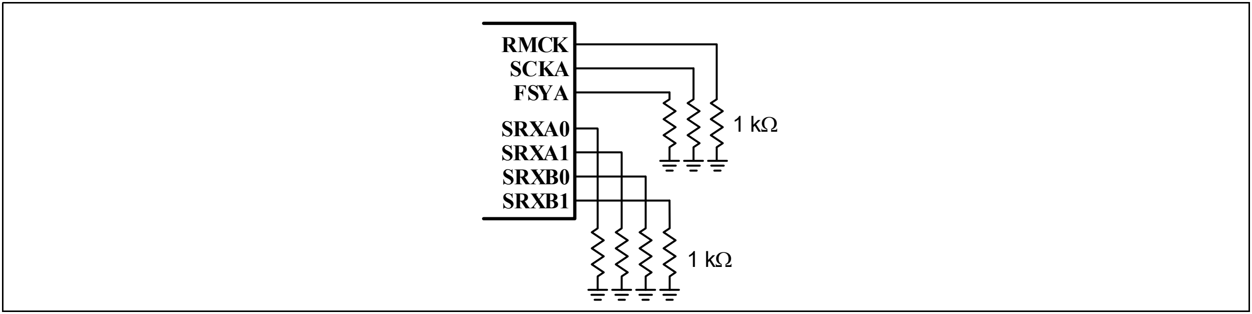
16.7 Termination

The OS81210 input pins should not be left floating in the application. Unused input pins can be connected directly to the power supply or to ground, unless otherwise noted. Alternatively, pull-ups or pull-downs can be used (in lieu of a direct connection to power or ground) for protection during erroneous configuration as an output as shown in the following figures.

16.7.1 STREAMING PORT

Figure 16-6 provides a recommendation for unused pin termination on the OS81210 Streaming Port.

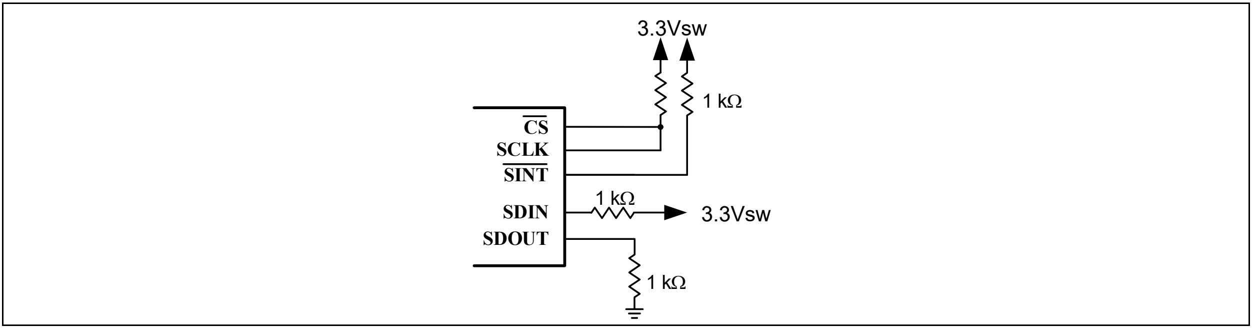
FIGURE 16-6: RECOMMENDED TERMINATION FOR STREAMING PORT



16.7.2 SPI PORT

Figure 16-7 provides a recommendation for unused pin termination on the OS81210 SPI Port.

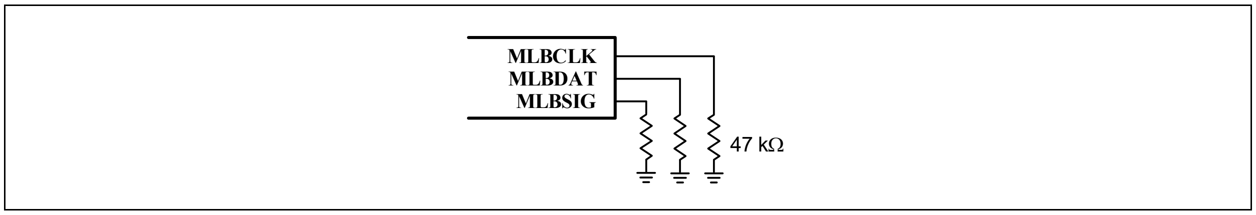
FIGURE 16-7: RECOMMENDED TERMINATION FOR SPI PORT



16.7.3 MEDIALB PORT

Figure 16-8 provides a recommendation for unused pin termination on the OS81210 MediaLB Port.

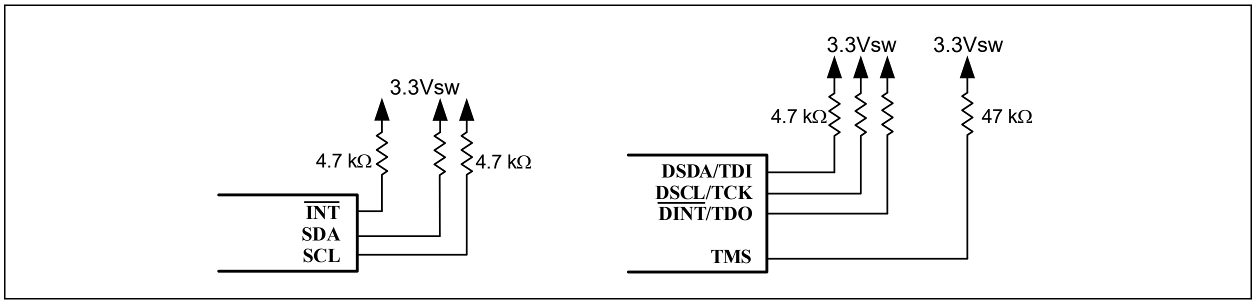
FIGURE 16-8: RECOMMENDED TERMINATION FOR MediaLB PORT



16.7.4 I^{2}C AND JTAG PORTS

Figure 16-10 provides a recommendation for unused pin termination on the OS81210 I^{2}C Port and JTAG Port.

FIGURE 16-9: RECOMMENDED TERMINATION FOR I^{2}C AND JTAG PORTS

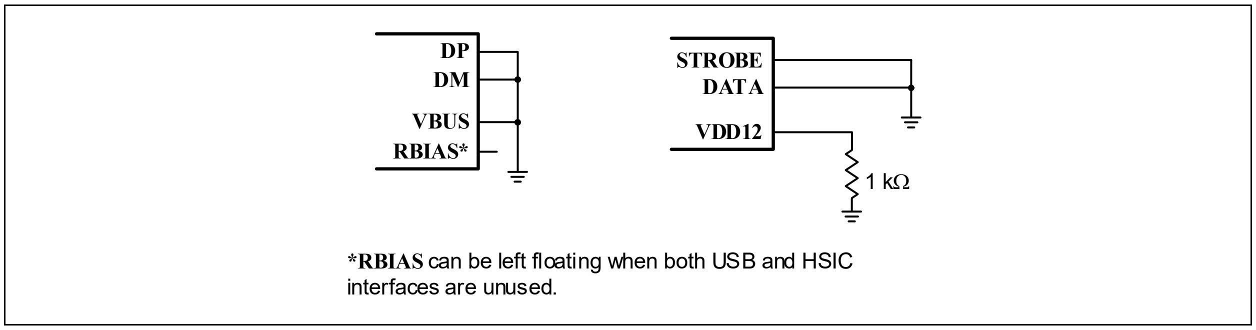


Note: Pull-ups are required on I^{2}C buses, regardless of whether the ports are used.

16.7.5 USB AND HSIC

Figure 16-10 provides a recommendation for unused pin termination on the OS81210 USB and HSIC Ports.

FIGURE 16-10: RECOMMENDED TERMINATION FOR USB AND HSIC PORTS


DP
DM
VBUS
RBIAS*
STROBE
DATA
VDD12
1 kΩ
*RBIAS can be left floating when both USB and HSIC interfaces are unused.

16.8 Crystal Oscillator Selection

Several factors must be considered when selecting a crystal. These include load capacitance, oscillator margin, cut, and operating temperature.

Oscillator margin is a measure of the stability of an oscillator circuit, and is defined as the ratio of the oscillator's negative resistance (R_{NEG}) to the crystal's ESR (R_{ESR}), or:

EQUATION 16-1: OSCILLATOR MARGIN MEASUREMENT

$$\text{Margin} = \frac{|R_{NEG}|}{R_{ESR}} = \frac{|R_{VAR}| + R_{ESR}}{R_{ESR}}$$

The negative resistance can be measured by placing a variable resistor (R_{VAR}) in series with the crystal and finding the largest resistor value where the crystal still starts up properly. This point would be just below where the oscillator does not start-up or where the start-up time is excessively long. Ideally, oscillator margin should be greater than 10, and should be at least 5. Smaller oscillator margin can affect the ability of the oscillator to start up.

The load capacitance, specified when ordering the crystal, is the series combination of the capacitance on each leg of the crystal. This capacitance includes not only the added capacitors, but also PCB trace (shunt) capacitance and chip pin capacitance. Larger capacitors also have a negative effect on oscillator margin. External capacitors on each leg (C1 and C2 in Figure 11-1) are typically in the 10 to 22 pF range. The transconductance (g_m) of the internal INIC inverting amplifier is nominally 4.56 mS.

TABLE 16-1: CRYSTAL OSCILLATOR SPECIFICATIONS

Name	Value	Description
Frequency	384×Fs	Frequency relative to operating network frame rate (Fs)
Correlation	Parallel Resonant	Mode of oscillation
Osc. Mode	Fundamental	Oscillation mode or operation mode
C_L	10-22 pF	Load capacitance (typical)
C_O	7 pF	Recommended maximum shunt (parallel) capacitance
ESR	100 Ω	Recommended maximum equivalent series resistance
Drive Level	50 µW	Drive level (typical)
Cut	AT	AT cut produces the best temperature stability
Tolerance	±200 ppm	Frequency tolerance over all conditions

The crystal cut and tolerance value listed in Table 16-1 are *typical* values and may be changed to suit differing system requirements. Higher ESR values (than those listed in the Table 16-1) run the risk of having start-up problems and should be thoroughly tested before being used. Contact the crystal manufacturer for more information.

16.9 Layout Guidelines

INIC board designs should follow the basic guidelines outlined throughout this chapter, with special consideration given to component selection and placement.

16.9.1 POWER AND DECOUPLING

A dedicated decoupling capacitor should be used at each OS81210 power pin for localized decoupling. Typical values are shown in Figure 16-1. For EMI-sensitive applications C0G capacitors are strongly recommended for their superior performance at high frequencies.

To reduce trace impedance and keep the loop areas small, decoupling capacitors should be positioned as close as possible to their respective OS81210 power pins. The recommended placement of decoupling capacitors is on the same side of the PCB as the OS81210. However, decoupling capacitors may be mounted on the opposite side of the PCB to reduce the distance between the capacitor and the pin, since inductance from vias is less harmful than long traces that may allow external interference. Two vias to the ground plane should be used for each decoupling capacitor to minimize inductance.

In situations where both large (e.g. electrolytic or tantalum) and small (e.g. ceramic) capacitors are used, the smaller capacitor should take priority for placement and layout optimization.

As in all low-noise circuits, two vias should be used from power and ground traces to the respective power and ground planes to reduce parasitic inductance, whenever possible. Additionally, power traces should be as wide as practical. The exposed paddle of the QFN package is the primary ground connection for the OS81210 and must be adequately connected to the PCB ground plane.

Ground-plane fill is used under the part to minimize the impedance between ground connections, which also helps minimize EMI. If a ground-plane fill must be broken by a trace, multiple stitching vias should connect each section of the fill to a reference ground plane on another layer. Vias that connect high frequency decoupling capacitors to the ground plane should be as close as possible to the OS81210 to minimize loop area back to the package exposed paddle ground.

16.9.2 MediaLB PORT

The OS81210 *Media Local Bus* (MediaLB) 3-pin interface is provided to facilitate high-speed data exchange between INIC and the EHC. MediaLB is intended for use on a single PCB. For effective operation, especially at higher bus speeds, signal routing and component placement is extremely important. Refer to the MediaLB Specification [5] for application and implementation guidelines including trace length requirements and layout examples.

16.9.3 PHYSICAL LAYER

Proper hardware implementation (e.g. component placement, signal routing and shield/chassis/digital grounding) is essential for achieving stringent automotive performance. For optimum performance, attention should be given to the routing and placement of physical layer components. When possible, the signals should be routed on a single layer and the use of vias should be minimized. Additionally, traces should be routed over a continuous reference plane, preferably a ground plane. To minimize reflections, two 45 degree turns should be used rather than a single 90 degree turn. Crosstalk should be avoided by not running other signals in parallel to the transmit and receive lines for long distances.

Note: For Physical Layer layout recommendations, please refer to the latest OS8121x Physical Layer Reference Design Application Note [13] available from Microchip Support upon request.

16.9.4 USB PORT

The OS81210 USB Port is used for high-speed data exchange between INIC and the EHC. On the application board, USB communication is implemented using either the USB 2.0 physical interface (**DP**/**DM**) or the HSIC physical interface (**STROBE**/**DATA**).

As mentioned in Chapter 8.0 “USB Port” on page 41, the OS81210 USB Port is only intended for implementations contained on a single application board. The PCB traces between the OS81210 and the EHC (USB host controller) are considered to be a USB captive cable. Therefore, any discussion of detachable USB cables or off-board USB connectors falls outside the scope of OS81210 USB Port layout recommendations.

For optimal operation, the USB signals should be treated as high-priority traces that are optimized for signal integrity with adequate countermeasures against external interference (e.g. crosstalk). When possible, the traces should be routed on a single layer and the number of vias should be minimized. Additionally, traces should be routed over a continuous reference plane, preferably a ground plane. To minimize reflections, two 45 degree turns should be used rather than a single 90 degree turn.

When the USB 2.0 physical interface is used, the **DP** and **DM** signals must be routed as an impedance controlled (e.g 90 Ω) differential pair and should be tightly coupled and matched in length with constant spacing maintained between them over their entire length. When the HSIC physical interface is used, the **STROBE** and **DATA** signals must be routed as impedance controlled (e.g. 50 Ω) singled-ended transmission lines. Refer to the Universal Serial Bus Specification 2.0 [15] for additional application and implementation guidelines, including trace length requirements.

16.9.5 CRYSTAL OSCILLATOR

When using an external crystal, the distance between the INIC **XTI**/**XTO** pins and the external crystal should be minimized. As mentioned in Chapter 11.0 “Clock Manager” on page 63, jitter on **XTI** must also be minimized. When an externally generated clock signal is used in lieu of a crystal oscillator, the clock signal must be routed as a high-priority trace with adequate countermeasures against jitter and crosstalk.

16.9.6 THERMAL CONSIDERATIONS

The exposed paddle of the QFN package provides the ground connection for the OS81210 and must be adequately connected to the PCB ground plane. The paddle of the package must be soldered to the thermal land (thermal pad) for efficient heat dissipation and proper grounding. The thermal pad should be at least as large as the exposed portion of the package paddle and be connected to the ground plane of the board using heat tubes (plated hole vias without thermal reliefs). At least nine heat tubes are recommended.

The OS81210 64-pin QFN land pattern is shown in Figure 17-4.

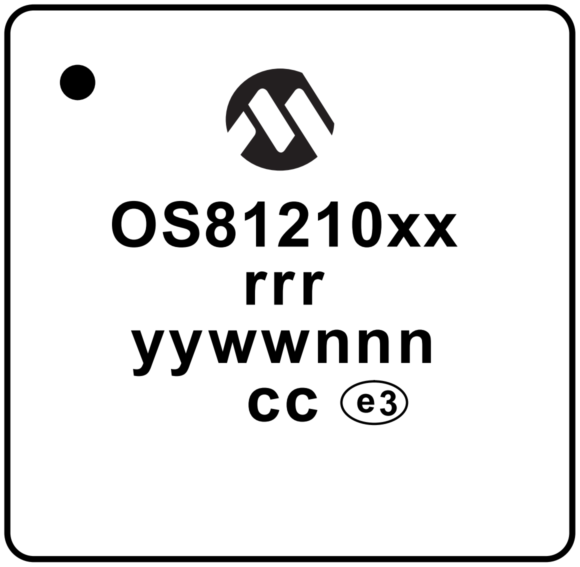
16.9.7 MISCELLANEOUS

Floating areas of copper fill near the INIC must be avoided due to the risk of noise coupling into the system. If needed, a keep-out can be used around the OS81210 and its immediate peripheral circuitry (e.g. decoupling capacitors, crystal, termination components, etc.) to ensure isolated regions of floating copper are not inadvertently created when copper fill is poured. If a keep-out is not used, multiple stitching vias should connect any floating copper fills to an appropriate reference on other power or ground layers.

17.0 PACKAGING INFORMATION

17.1 Package Marking

FIGURE 17-1: OS81210 TOP MARKING



The package designators are:

- xx - Device Identifier
  - **OS81210AF** - Solderable Flank QFN
- rrr - Product Revision Code
- yy - last two digits of Assembly Year
- ww - Assembly Work Week
- nnn - Tracking Number
- cc - Country of Origin Abbreviation (optional - up to 2 characters)
- e3 - Pb Free Symbol

17.2 Package Thermal Characteristics

TABLE 17-1: OS81210 64-PIN SOLDERABLE FLANK QFN THERMAL CHARACTERISTICS

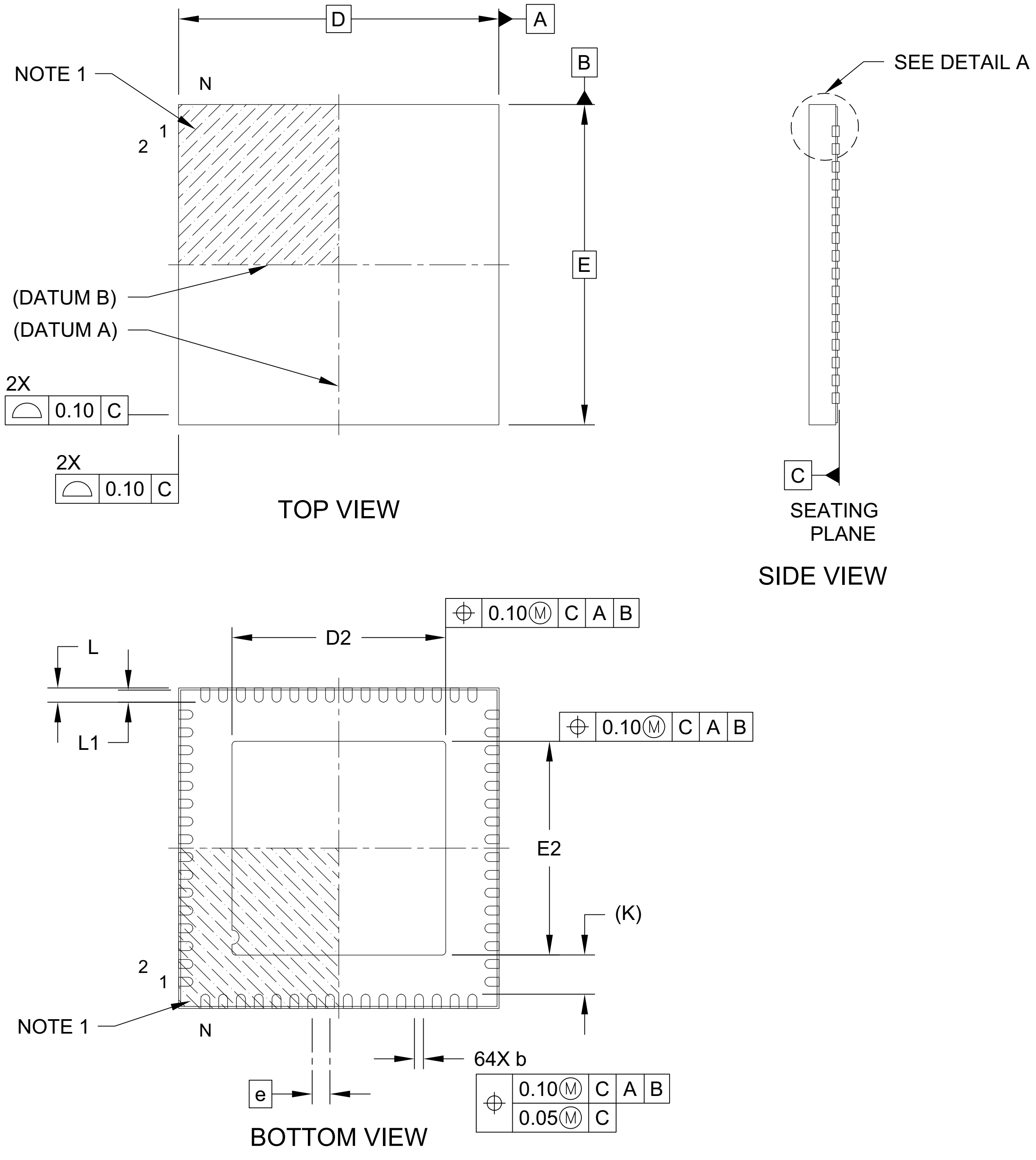
Parameter	Symbol	Value	Unit
Typical Junction to Package	Ψ_{JT}	0.1	ºC/W
Typical Junction to Ambient	θ_{JA}	22	ºC/W

17.3 Package Outline Drawings

FIGURE 17-2: OS81210 64-PIN SOLDERABLE FLANK QFN PACKAGE DRAWING (1 OF 3)

64-Lead Very Thin Plastic Quad Flat, No Lead Package (KJX) - 9x9x0.9 mm Body [VQFN] Wettable Flanks (Stepped)

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

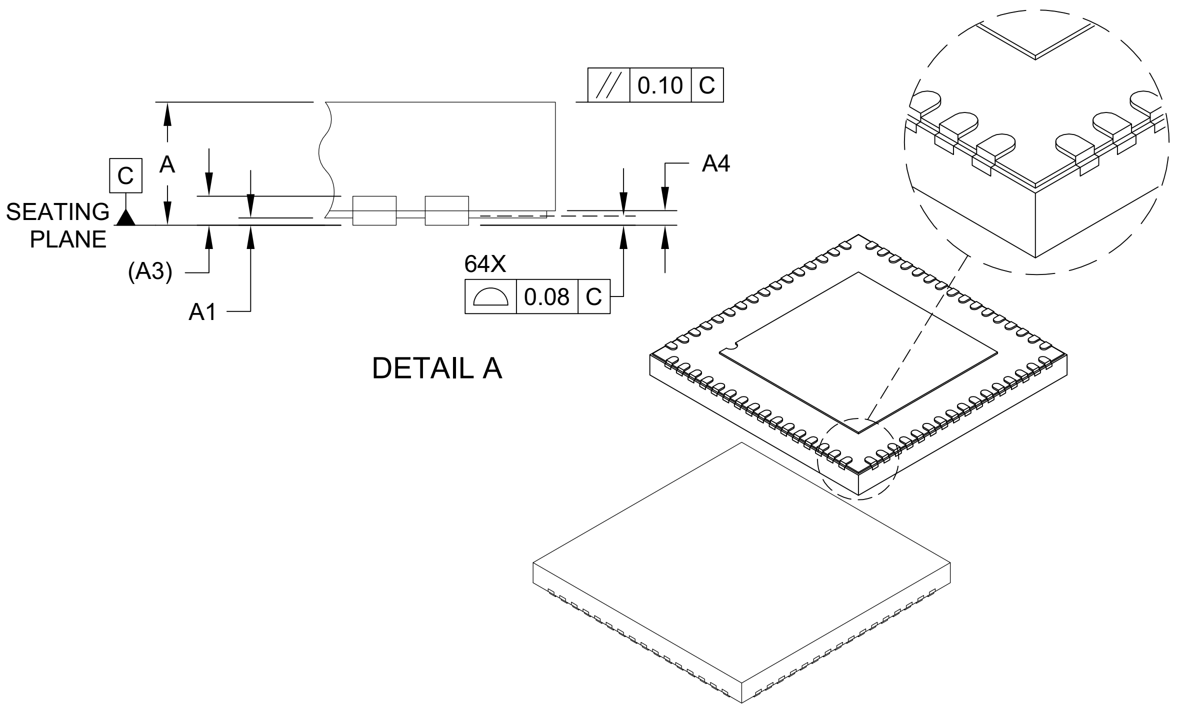


Microchip Technology Drawing C04-450A Sheet 1 of 2

FIGURE 17-3: OS81210 64-PIN SOLDERABLE FLANK QFN PACKAGE DRAWING (2 OF 3)

64-Lead Very Thin Plastic Quad Flat, No Lead Package (KJX) - 9x9x0.9 mm Body [VQFN] Wettable Flanks (Stepped)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		64	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Step Height	A4	0.10	-	-
Overall Width	E		9.00 BSC	
Exposed Pad Width	E2	5.90	6.00	6.10
Overall Length	D		9.00 BSC	
Exposed Pad Length	D2	5.90	6.00	6.10
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.35	0.40	0.45
Step Length	L1	0.28	0.34	0.40
Terminal-to-Exposed Pad	K		1.10 REF	

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

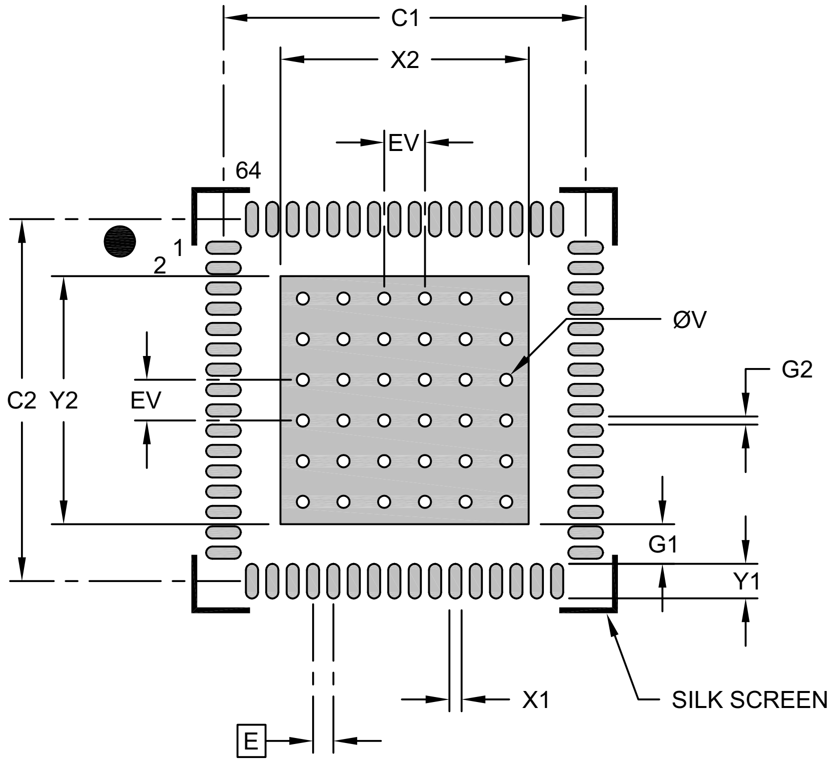
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-450A Sheet 2 of 2

FIGURE 17-4: OS81210 64-PIN SOLDERABLE FLANK QFN PACKAGE DRAWING (3 OF 3)

64-Lead Very Thin Plastic Quad Flat, No Lead Package (KJX) - 9x9x0.9 mm Body [VQFI Wettable Flanks (Stepped)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			6.10
Optional Center Pad Length	Y2			6.10
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			0.85
Contact Pad to Center Pad (X64)	G1	0.20		
Contact Pad to Contact Pad (X60)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2450A

APPENDIX A: REFERENCES

Documents listed below and referenced within this publication are current as of the release of this publication and may have been reissued with more current information. To obtain information on Microchip automotive documentation please submit a case through www.microchip.com/support.

All non-Microchip documentation should be retrieved from the applicable website locations listed below. Microchip is not responsible for the update, maintenance or distribution of non-Microchip documentation.

Because the Internet is a constantly changing environment, all Internet links mentioned below and throughout this document are subject to change without notice.I2C-Bus Specification [7]

[1] ISO 21806 (Parts 1-7): 2020 – Road Vehicles – Media Oriented Systems Transport (MOST) Specifications

International Standards Organization:

Part 1: General Information and Definitions: www.iso.org/standard/71833.html

Part 2: Application Layer: www.iso.org/standard/71834.html

Part 3: Application Layer Conformance Test Plan: www.iso.org/standard/71835.html

Part 4: Transport Layer and Network Layer: www.iso.org/standard/71836.html

Part 5: Transport Layer and Network Layer Test Plan: www.iso.org/standard/71837.html

Part 6: Data Link Layer: www.iso.org/standard/71838.html

Part 7: Data Link Layer Test Plan: www.iso.org/standard/71839.html

[2] ISO 21806 (Parts 14-15): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) Lean Application Layer Specification

International Standards Organization:

Part 14: Lean Application Layer: www.iso.org/standard/77930.html

Part 15: Lean Application Layer Conformance Test Plan: www.iso.org/standard/77931.html

[3] INICkit Tool

Microchip. www.k2l.de.

[4] INIC Interface Specification

DS60001464. Microchip. www.microchip.com/support.

[5] MediaLB Specification

TB0400AN4V2. Microchip. www.microchip.com/support.

[6] Port Message Protocol Version 2 User's Guide

Microchip. www.microchip.com/support.

[7] I^{2}C-Bus Specification

UM10204_3. NXP (formerly a division of Philips). www.nxp.com.

[8] MediaLB Monitor Adapter User's Manual

Microchip. www.k2l.de.

[9] MediaLB Analyzer User's Manual

Microchip. www.k2l.de.

[10] SPI Block Guide Version 4.01

S12SPIV4. NXP Semiconductors (formerly Freescale Semiconductor). www.nxp.com.

[11] IEEE Standard Test Access Port and Boundary-Scan Architecture

IEEE Std. 1149.1. Institute of Electrical and Electronics Engineers (IEEE). www.ieee.org.

[12] ISO 21806 (Parts 12-13): 2021 – Road Vehicles – Media Oriented Systems Transport (MOST) 50 Mbit/s Balanced Media Physical Layer Specification

International Standards Organization:

Part 12: 50 Mbit/s Balanced Media Physical Layer: www.iso.org/standard/77927.html

Part 13: 50 Mbit/s Balanced Media Physical Layer Conformance Test Plan: www.iso.org/standard/77928.html

[13] OS8121x Physical Layer Reference Design Application Note

DS60001452. Microchip. www.microchip.com/support.

[14] Device Update Process User's Guide

DS60001494. Microchip. www.microchip.com/support.

[15] Universal Serial Bus Specification 2.0

Universal Serial Bus Organization. www.usb.org.

[16] High-Speed Inter-Chip USB Electrical Specification 1.0

Universal Serial Bus Organization. www.usb.org.

APPENDIX B: REVISION HISTORY

Revision E (DS60001493E March 2023)

- Removal of “Confidential” status

Revision D (DS60001493D July 2021)

- Updated BRXN, BRXP Pin Voltage Reference
- Updated Receive Eye Mask Voltage Reference
- Updated Reference to ISO MOST Specifications

Revision C (DS60001493C, March 2020)

- Updated for Release to Production
- Electrical Characteristics
  - Updated max current, power, and network characteristics for B2B hardware

Revision B (DS60001493B, January 2019)

- General
  - Updated pin names to reference the B1A hardware update
- Overview
  - Introduced Centralized Network Diagnostics and Fallback Operation
- MediaLB Port
  - Updated MediaLB 3-pin diagrams to better demonstrate bus topology
- Streaming Port
  - Introduced Streaming Port pulse density modulation (PDM) format
  - Added TDM support for 512×Fs
- Electrical Characteristics
  - Introduced maximum MUTE pin delay from event (t_{muteon})
  - Updated max current, power, and network characteristics for B1A hardware

Revision A (DS60001493A, July 2017)

- Initial document release.

APPENDIX C: LIST OF ACRONYMS

The following is a list of commonly used acronyms found throughout this document.

ADC – Analog to Digital Converter

AFE – Analog Front-End

AKE – Authentication and Key Exchange

API – Application Programming Interface

AVP – Audio / Video Packetized

CKE – Content Key Exchange

DAC – Digital to Analog Converter

DCI – Driver Control Interface

DFI – Discrete Frame Isochronous

DTCP – Digital Transmission Content Protection

ECU – Electronic Control Unit

EHC – External Host Controller

FPH – FIFO Protocol Header

GPIO – General Purpose I/O

ICM – INIC Control Message

INIC – Intelligent Network Interface Controller

IOC – I/O Companion

IPC – Inter-Processor Communication

JTAG – Joint Test Action Group

MCM – Meta Control Message

MEP – Meta Ethernet Packet

MDP – Meta Data Packet

MHP – MOST High Protocol

MOST – Media Oriented Systems Transport

PDM – Pulse Density Modulation

PMHL – Port Message Header Length

PML – Port Message Length

PMP – Port Message Protocol

QFN – Quad Flat No-leads package

QoS – Quality of Service

RCM – Remote Control Message

RHC – Remote Host Controller

RMCK – Remote Master Clock

SPI – Serial Peripheral Interface

STP – Switch To Power

STP – Shielded Twisted Pair

TDM – Time Division Multiplexing

UNICENS – Unified Centralized Network Stack

UTP – Unshielded Twisted Pair

VIOC – Video I/O Companion

VQFN – Very thin Quad Flat No-leads package

APPENDIX D: LIST OF TABLES

APPENDIX E: LIST OF FIGURES

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under “Support”, click on “Customer Change Notification” and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://microchip.com/support

AUTOMOTIVE PRODUCT SUPPORT

Customers using Microchip automotive products can receive additional help through the Automotive Support Contacts page of the Microchip web site.

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	X	X	[X]	-	rrr	-	vvvvvv	[ss]	-	[xxx]
Device	Grade	Package Type	Tape and Reel Flag		Product Revision Code		Firmware Revision Code	Firmware Service Release		Special Feature Code

Device	OS81210	=	50 Mbit/s Automotive Intelligent Network Interface Controller with USB
Grade	A	=	All Features
Package Type	F	=	QFN with solderable flanks
Tape and Reel Flag (optional)	Blank	=	Standard Packaging (Tube/Tray)
	R	=	Tape and Reel
Product Revision Code	rrr	=	3 character code specifying product revision
Firmware Revision Code	vvvvvv	=	6 character code specifying firmware revision
Firmware Service Release (optional)	ss	=	2 character code specifying service release
Special Feature Code (optional)	xxx	=	3 character code for special requirements

Examples:

a) OS81210AF-rrr-vvvvvv-xxx
64-pin solderable flank QFN package

b) OS81210AFR-rrr-vvvvvv-xxx
64-pin solderable flank QFN package,
Tape and Reel

NOTES:

ISBN: 978-1-5224-8443-1

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820