Intel® IXDP425 Development Platform: Interfacing the Intel IXF3208D Octal T1/E1/J1 Framer/ LIU

Application Note

September 2004

Document Number: 254030-002

Contents

Figures

Tables

Revision History

Date	Revision	Description
September 2004	002	Updated Intel® product branding.
February 2004	001	Initial release.

This page is intentionally left blank.

1.0 Introduction

This application note demonstrates the interoperability of the Intel® IXP425 Network Processor, the Intel® IXF3208 Framer, and the Intel® Line Interface Unit (LIU) device.

The document describes general implementation and presents guidelines for connecting the Intel® IXDP425 Development Platform with the Intel® IXF3208D Octal T1/E1/J1 Framer Development Platform. The document is intended to provide general-connectivity information regarding T1 and E1 signals for these boards — many available features are not described in detail.

The following is a synopsis of the major steps involved in this document:

1. Configure the two platforms independently, test.
2. Connect the two platforms, test.
3. Discuss IXP425-IXF3208 framer connections.
4. Discuss an actual product design.

References to other documents are included (see "Reference Documents" on page 7); these documents may provide further insight into possible features and implementations.

Note: Intel is **discontinuing** its LXT3108 Line Interface Unit; the Exar* XRT83L38 octal LIU is the recommended replacement.

1.1 Scope

The purpose of this document is to provide hardware and software configuration guidelines on how to interface the IXDP425 development platform and the IXF3208D Octal T1/E1/J1 Framer, using an external reference clock rate of 8.192 MHz for both T1 and E1 modes. Main sections include:

- Section 4.0, "Interconnection of the Intel® IXF3208D Octal T1/E1/J1 Framer and the IXDP425 Development Platform" on page 14
- Section 5.0, "Intel® IXF3208D Octal T1/E1/J1 Framer Configuration – T1 Mode" on page 16; Section 6.0, "Intel® IXF3208D Octal T1/E1/J1 Framer Configuration — E1 Mode" on page 24
- Section 7.0, "Intel® IXDP425 / IXCDP1100 Development Platform Configuration" on page 32
- Section 8.0, "Interfacing Intel® IXP425 Network Processor to the Intel® IXF3208 Framer in a Product Design" on page 35

1.2 Reference Documents

It is strongly recommended that the user read and understand the following reference material:

Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet

http://developer.intel.com/design/network/datashts/252479.htm

Intel® IXP42X Product Line and IXC1100 Control Plane Network Processors Hardware Design Guidelines

http://developer.intel.com/design/network/desguides/252817.htm

Intel® IXDP425/IXCDP1100 Development Platform Documentation Kit

http://developer.intel.com/design/network/manuals/IXDP425_Documentation_Kit.htm

Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Specification Update

http://developer.intel.com/design/network/specupdt/253527.htm

Intel® IXP400 Software Releases (for Intel® IXP42X product line and IXC1100 control plane processors)

http://developer.intel.com/design/network/products/npfamily/ixp425swr1.htm

Intel® IXP400 Software Specification Update

http://developer.intel.com/design/network/specupdt/273795.htm

Intel® IXF3208 Octal T1/E1/J1 Framer with Intel® On-Chip PRM Datasheet

http://developer.intel.com/design/network/products/WAN/datashts/249544.htm

Intel® IXF3208D Evaluation Board Quick Start Guide Developer's Manual

http://developer.intel.com/design/network/products/wan/manuals/250455.htm

Intel® IXF3208D Evaluation Board Developer's Manual

http://developer.intel.com/design/network/products/wan/manuals/250272.htm

Intel® IXF3208 Octal T1/E1/J1 Framer with Intel® On-Chip PRM Memory Map Software Developer Manual

http://developer.intel.com/design/network/products/wan/manuals/251082.htm

1.3 Acronyms and Abbreviations

Table 1 describes the acronyms and abbreviations used in this application note.

Table 1. Acronyms and Abbreviations (Sheet 1 of 2)

Acronym/Abbr.	Description
AMI	Alternate Mark Inversion
API	Application Program Interface
B8ZS	Binary Eight Zero Substitution
BERT	Bit Error Rate Test
BPV	BiPolar Violation

Table 1. Acronyms and Abbreviations (Sheet 2 of 2)

Acronym/Abbr.	Description
CHI	Concentration Highway Interface
ChtoCh	Channel to Channel
CRC	Cyclic Redundancy Check
GUI	Graphical User Interface
H-MVIP	High-density Multi-Vendor Integration Protocol
HDB3	High Density Bipolar Three
HDLC	High level Data Link Control
HSS	High-Speed Serial [port]
I/O	In/Out
LH/SH	Long Haul/Short Haul
LIU	Line Interface Unit
LOS	Loss Of Signal
Mbps	Megabits per second
MVIP	Multi-Vendor Integration Protocol
NEG	Negative
NPE	Network Processor Engine
PLL	Phase Lock Loop
PCM	Pulse Code Modulation
POS	Positive
TDM	Time Division Multiplex
TX	Transmit (HSS is transmitting off-chip)
QRSS	Quasi-Random Sequence Signal
RX	Receive (HSS is receiving from off-chip)

2.0 Technical Overview

This section gives a brief overview of various technologies and protocols; the reader should be familiar with the specifications and standards involved. Only certain aspects and features of the IXF3208 framer and IXP425 are covered. It is possible to add other features and tune the configuration to be application-specific; however, for simplicity, this application note covers only basic configurations of the IXF3208D and IXDP425.

2.1 T1/E1

The E1 and T1 standards allow individual voice or data channels to be multiplexed together to form a high-speed connection that can be carried on a common transmission medium.

A T1 frame is constructed of 24 timeslots plus 1-framing bit. Each timeslot is regarded as a channel of 64 kbit/s bandwidth.

The frame length is 193 bits (24*8 + 1), and the frame rate is 8 kHz. The Framing bit creates a channel of 8 kbit/s and is used for messages, synchronization, and alarms.

A T1 line is mainly used in North America and Canada and has a line rate of 1.544 Mbits/s.

In Europe, the E1 standard is used, which bears many similarities to T1. In an E1 signal, there are 30 timeslots instead of 24, plus a further 2 used for alignment and signaling.

The line rate is determined by a total of 32 timeslots consisting of 8 bits at a rate of 8,000 times a second, giving a transmission rate of 2.048 Mbit/s.

Further information on T1 and E1 standards and different frame structures can be found in the *Intel® IXF3208D Octal T1/E1/J1 Framer Datasheet*.

Note: In this application note, the focus is on basic T1 and E1 frames.

2.1.1 Line Encoding

Detailed encoding information can be found in the *IXF3208D Octal T1/E1/J1 Framer Datasheet*; however, the basic line encoding is explained below. The line encoding can be configured in the LIU or the Framer; however, only one device would require the line encoding to be configured.

2.1.1.1 Alternate Mark Inversion (AMI)

Alternate mark inversion (AMI) is a form of bipolar signaling in which each successive mark is of the opposite polarity and spaces have zero amplitude. In AMI, the transmitted power is concentrated in the middle of the transmission bandwidth. This minimizes signal distortion while eliminating DC voltage build-up on the line.

2.1.1.2 High-Density Bipolar Three Coding (HDB3)

Advantages of bipolar signaling are the absence of a DC voltage component in the signal and the ability to recover clocking from an all-ones condition. A disadvantage, however, is that during a period of zero voltage (no signal), the E1 line may not be able to recover clocking and some type of coding may be required to maintain synchronization.

In high-density bipolar 3 (HDB3) coding, each string of four consecutive zeros in a byte are replaced by the HDB3 code. Two different HDB3 codes are used to ensure that the bipolar violations from adjacent four-zero groups are of the opposite polarity. The choice of which of the two codes to use depends on whether there was an odd or even number of ones since the previous bipolar violation occurred. If an odd number of ones occurred since the previous bipolar violation, the following coding method is used: 000V (where V is a bipolar violation).

If an even number of ones occurred since the previous bipolar violation, the following coding method is used: P00V (where P is a bit having the opposite polarity of the immediately preceding bit and V is a bipolar violation).

HDB3 encoding is mainly used in Europe over E1 lines.

2.1.1.3 Bipolar 8 Zero Substitution (B8ZS)

Used in North America, B8ZS is an updated version of AMI, which helps PLLs maintain synchronization even with long strings of zeros (a problem with AMI).

Sequential ones change polarity from + to -, as was the case in AMI, but now a string of eight zeros will transform to:

- 00PN0NP if the polarity of the preceding 1 is positive, or
- 000NP0PN if the polarity was negative.

(P stands for positive ‘1’, and N for negative ‘1’)

2.2 Intel® IXF3208D Octal T1/E1/J1 Framer Backplane Interface

Table 2 shows interfaces supported in the backplane of the IXF3208D. Specification and timing diagrams of each bus can be found in the *Intel® IXF3208D Octal T1/E1/J1 Framer Datasheet*. H-MVIP is the Bus Format discussed in this application note.

Table 2. Intel® IXF3208D Octal T1/E1/J1 Framer Backplane Interfaces

Backplane Bus Format	Complete Name	Source
MVIP H-MVIP	Multi-Vendor Integration Protocol High-Density MVIP	Global Organization for Multi-Vendor Integration Protocol
H.100 CT Bus	Computer Telephony Bus	Enterprise Computer Telephony Forum
ST-Bus	Serial Telecom Bus	Mitel Networks*
CHI	Concentration Highway Interface	AT&T Microelectronics*

2.3 Intel® IXF3208D Octal T1/E1/J1 Framer Backplane Concentration Mode

The IXF3208D supports a ‘backplane concentration mode’ in which eight ports can be output on a single ball. When the concentration mode is:

• 4x, the ball associated with:

— Port 0 outputs the data of Ports 0, 1, 2, and 3.

— Port 4 outputs the data of Ports 4, 5, 6, and 7.

• 8x, the ball associated with Port 0 outputs the data of all the eight ports, byte interleaved with port 0 byte output first, and then the byte of Port 1 output, and so on.

In each case, the IXF3208D internally interleaves data so they are output in a byte-interleaved way. For example, in 4x mode the byte of Port 0 is output first, and then the byte of Port 1, and so on.

Note: H-MVIP waveforms can be generated on both Ports 0 and 4.

This method of interlacing gives multiple E1 streams onto a single E1 line, and multiple T1 streams onto a single T1 line. This would require a clock rate four or eight times the base frequency in the backplane to be able to concentrate multiple T1/E1 lines.

Later sections discuss how this can be achieved in the IXF3208D. Increasing the clock frequency and alternating various timeslots helps provide this functionality. A single T1 line can also be mapped into an E1 line.

Note: When interfacing the IXF3208 framer with the IXP425, the T1 frame will need to be mapped into E1 structure, as the IXP425 can only be configured at a clock rate that is a multiple of E1 (for example, 8.192 MHz).

2.4 High-Speed Multi-Vendor Interface Protocol (H-MVIP)

HMVIP, a standard interface for makers of telephony devices, defines a synchronous TDM bus of N*64 kbps constant bit rate data streams. Each 64-kbps data stream carries an 8-bit data that operates at 8.192 MHz. There are 128 timeslots available to carry data on each stream. A framing pulse marks the beginning of timeslot 0, which occurs every 125 µs. For the Tcarrier environment, only 96 timeslots are used. Devices carrying E1 traffic use all 128 timeslots.

Both T-carrier and E-carrier HMVIP implementations carry a limit of 672 timeslots.

2.5 Test Patterns

In the configuration discussed in this application note, QRSS is the test pattern used to generate the traffic on a physical line. The T1/E1 test analyzer confirms synchronization of clock, and validation of the data sent and received after remote loopback is configured in the IXDP425.

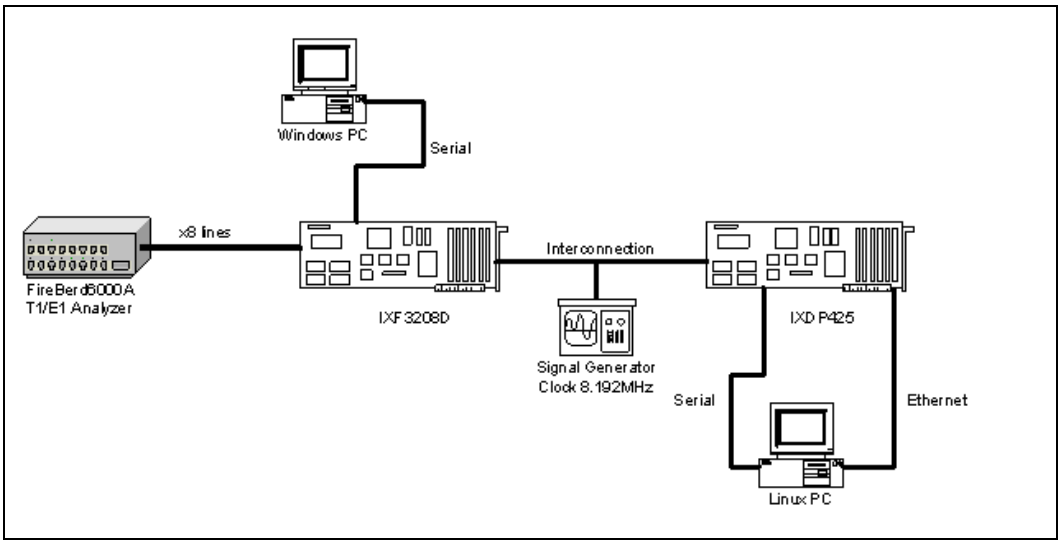
2.5.1 Quasi-Random Sequence Signal (QRSS)

This pseudo-random sequence is based on a 20-bit shift register and repeats every 1,048,575 bits. Consecutive zeros are suppressed to no more than fourteen in the pattern, which contains both high- and low-density sequences. QRSS is the most commonly used test pattern for T1 maintenance and installation. This pattern tests timing recovery, ALBO and equalizer circuits, but it also simulates customer traffic on the circuit. The QRSS pattern can be used framed or unframed and will force a B8ZS code in circuits.

3.0 Execution Environment

This section describes the hardware and software used for the configuration discussed in this application note. Figure 1 shows the test setup.

Figure 1. IXDP425/IXF3208D Test Setup



3.1 Test Analyzer

The FIREBERD* 6000A* is used with an add-on T1 or E1 module, depending on the required line rate.

The FIREBERD 6000A is only able to generate clock frequency at 1.544 Mbps and below with T1 card, or 2.048 Mbps and below with E1 card. Hence, only one port is configured at a time.

However, other equipment is available that supports multiple T1/E1 lines for simultaneous 8-port testing.

QRSS is a data pattern used to generate traffic on the line. For T1, the port is configured at a 1.544-Mbps clock rate, B8ZS encoding and ESF framing type. Information regarding different frame structures can be found in the *Intel® IXF3208D Octal T1/E1/J1 Framer Datasheet.*

The clock is set to INTF (FIREBERD-specific), where no internal clocks are used to sync with Tx or Rx signals.

All other settings are left to default.

3.2 External Clock — Signal Generator

In this setup a signal generator is used as an external clock, output frequency is set to 8.192 MHz, 3.3V, square wave. The external clock provides the Rx and Tx clocks for both the IXF3208D and IXDP425. An oscillator can be used to generate the same Rx and Tx clock signals.

The IXF3208D uses the Cypress Semiconductor* CY2292F* PLL, a 20-MHz crystal, as a reference clock to generate all other required clocks on the board; refer to the clock-configuration section of the *Intel® IXF3208D Evaluation Board Developer's Manual*.

The PLLs in the IXF3208D are used in our setup to transfer a 8.192-MHz backplane clock rate (4 x E1) into 4 x T1.

3.3 IXF3208D and IXDP425 Setup

There has been no modification made to the IXF3208D, but one modification has been made to the IXDP425 HSS ports to enable access to necessary pins.

The IXF3208D is split into two sections: the LIU and the Framer. The configuration discussed below should only be used with a Revision B IXP425 network processor, Revision B3 LXT3108 LIU, and a Revision C0 IXF3208 framer; these revision numbers are visible on the board and chips. The GUI version 2.0 should be used to configure the IXF3208D.

The IXF3208D is connected to the Microsoft* Windows* XP PC via a GUI port. The user is expected to have read the *Intel® IXF3208D Evaluation Board Quick Start Guide Developer's Manual* for basic configuration information.

The IXF3208D and the Test Analyzer are connected via twisted pair.

The IXDP425 must be set up according to the *Intel® IXDP425 / IXCDP1100 Development Platform Quick Start Guide*. The IXDP425 (Revision B) needs to be configured with Intel® IXP400 Software v.1.3 in a Red Hat* Linux environment using MontaVista Pro 3.0.

4.0 Interconnection of the Intel® IXF3208D Octal T1/E1/J1 Framer and the IXDP425 Development Platform

This section outlines the wiring between the IXF3208D and the IXDP425. Pin connections should be established according to Figure 2.

Figure 2. IXF3208D-IXDP425 Connections

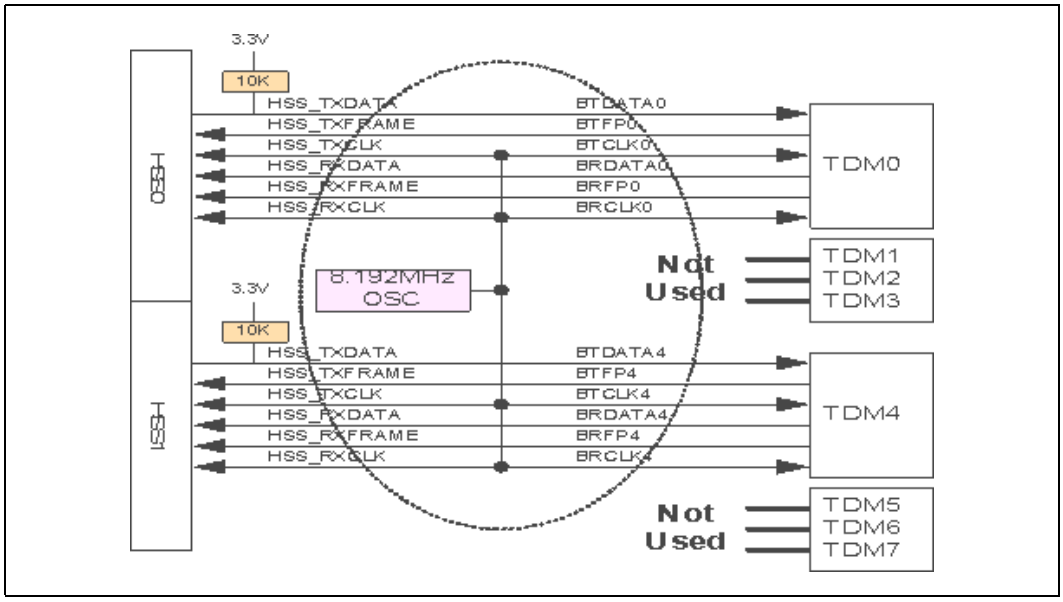
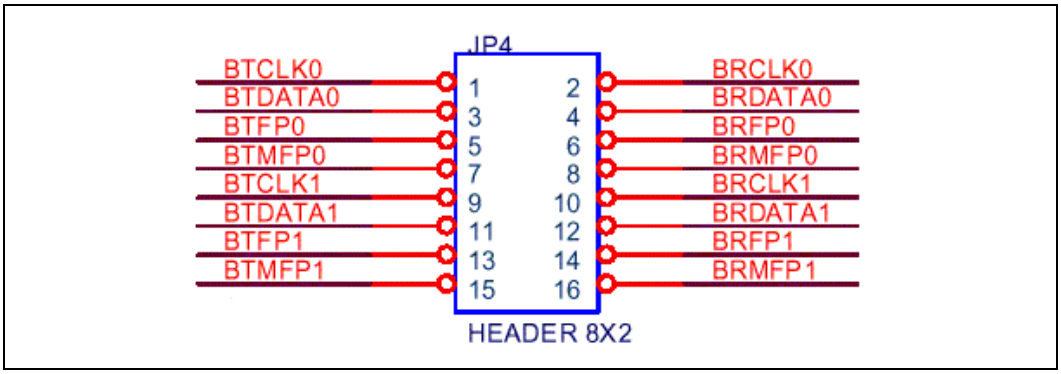


Figure 3 shows the pinout diagram of IXF3208D Jumper 4. Only Port 0 and Port 4 are used in this application; the Port 4 pinout diagram is identical and therefore not shown. Port 0-related pins from the IXF3208D should be connected to HSS0 pins of the IXDP425 and Port 4-related pins from the IXF3208D should be connected to HSS1 pins of the IXDP425. Section 4.0 shows how four channels are logically aggregated to one port.

Figure 3. IXF3208D Jumper 4 Pinout



The HSS0 and HSS1 pinout diagram can be found in the IXDP425 schematics.

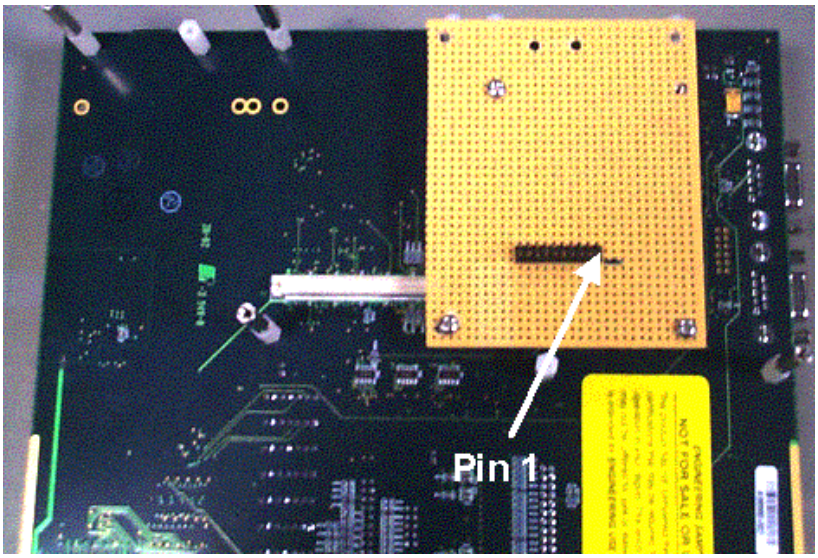
Each HSS connector (HSS-0 or HSS-1) is a 2 x 60-pin connector. On the IXDP425, HSS signals (HSS-0 or HSS-1) are routed to this connector.

To be able to access the required pins, a breadboard of approximately 110 mm x 70 mm should be used so that it can be mounted under the relevant HSS connector on the IXDP425. The IXDP425 should have pillars on it for mounting.

Next, solder the 10-way header onto the card; solder this header just above (and parallel to) the HSS0 connector and solder another 10-way header on the card to HSS1.

Figure 4 shows a connector soldered to HSS0. Ribbon cable is used to connect HSS0 pins to the connector. Ensure that all IXDP425 connection is on the signal side of the resistors.

Figure 4. HSS0 Modified Jumper Connector



Once this is done for both HSS0 and HSS1, connection between the IXF3208D and the IXDP425 can be established.

To create a common ground between the two platforms, connect one end of the test clip to TP11 on the IXF3208D and the other end to JP5 on the IXDP425.

Note: Shielded cable with test clips or crocodile clips are recommended when connecting the two boards.

Now complete the following:

1. Connect from BTCLK0 from JP4 to Card HSS0_TX_CLK.
2. Connect from BRCLK0 from JP4 to Card HSS0_RX_CLK.
3. Connect from BTDATA0 from JP4 to Card HSS0_TX_DATA.
4. Connect from BRDATA0 from JP4 to Card HSS0_RX_DATA.
5. Connect from BTFP0 from JP4 to Card HSS0_TX_FRAME.
6. Connect from BRFP0 from JP4 to Card HSS0_RX_FRAME.

Now establish the same connection for Port 4 to HSS1.

5.0 Intel® IXF3208D Octal T1/E1/J1 Framer Configuration – T1 Mode

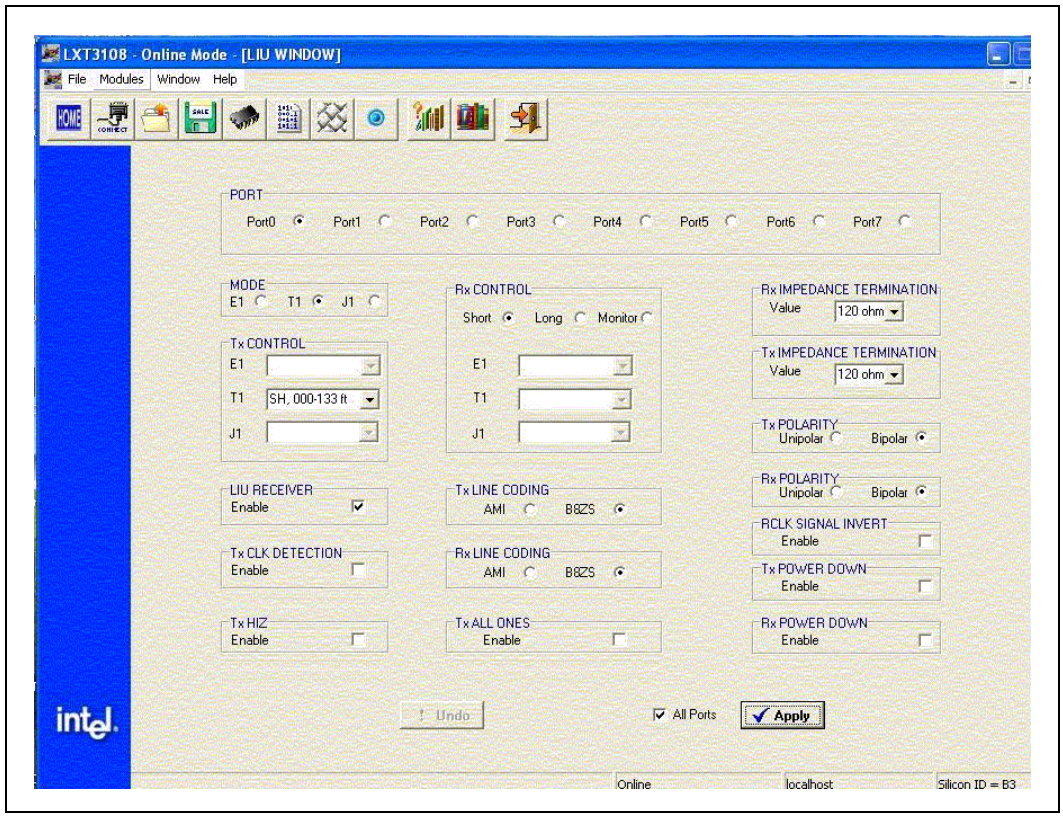
Using the GUI, configure the LXT3108 LIU in T1 mode, followed by IXF3208 framer configuration in T1 mode.

5.1 Intel® LXT3108 Line Interface Unit Configuration

1. Start the 'IXF3208D Application'.
   - Keep the baud rate settings as is, to 38400 on Serial Port COM1, unless using another COM port, and **START** the communication module with the Serial port.
2. Launch the LXT3108 GUI and establish connection by clicking on **LXT3108 connection**. See **Online** and **Silicon ID = B3** at the bottom of the window; this confirms correct silicon stepping and communication with the silicon.

   Note: The user should be familiar with the icons and features of the LXT3108 GUI.
   - Next, for simplicity, we focus on the setup.
3. Select **LIU** from the **Modules** drop-down menu and configure **All ports** according to Figure 5.
   - Ensure **LIU RECEIVER** is enabled.
   - All other modules should be left to default.

Figure 5. Intel® LXT3108 T1 Mode Configuration – LIU Module



4. For testing purposes, open the **DIAGNOSTIC** from the **Module** drop-down menu. Multiple **loopback modes** can be enabled for testing; enable **Remote loopback** on **All ports**, and you should be able to see receiving data being transmitted back into the test analyzer, with the clock synchronized.

Once this test is successfully completed, disable all Loopback modes and continue with the configuration.

5.2 IXF3208D Configuration

1. Launch the IXF3208D GUI and establish connection by clicking on the IXF3208D connection.

   Note: See **Online** and **Device ID = C0** at the bottom of the window; ensure the FW ID 0A. This confirms correct silicon stepping, firmware and communication with the silicon.

Again, there are a lot of possible variables; for simplicity, the following sections highlight the basic setup. Required module configuration are covered; modules not mentioned should retain their default settings.

5.2.1 Alarms

Configure Alarm Module by clicking on **Alarm** from the **Modules** drop-down menu. Ensure all alarms are **Disabled for all ports**, especially **Alarms to Backplane**, as this is not disabled by default.

5.2.2 Backplane

Open the **Backplane** module from the **Modules** drop-down menu and configure both **Rx Backplane** and **Tx Backplane** according to the upcoming figures, which show the configuration for Port 0. Note that configurations for Ports 1 to 3 are NOT identical to Port 0. Their differences are explained in their respective sections. Port 4 should be configured to match Port 0 and Port 5-7 should be configured to match Port 1-3.

Note: "Transmit" and "Receive" refer to the device connected to the backplane (that is, the IXDP425); the **Transmit** configuration on the backplane of the IXF3208D (that is, the Framer) should match the **Transmit** configuration of the HSS port on the IXDP425.

The following contains details common to both Transmit and Receive backplane configuration.

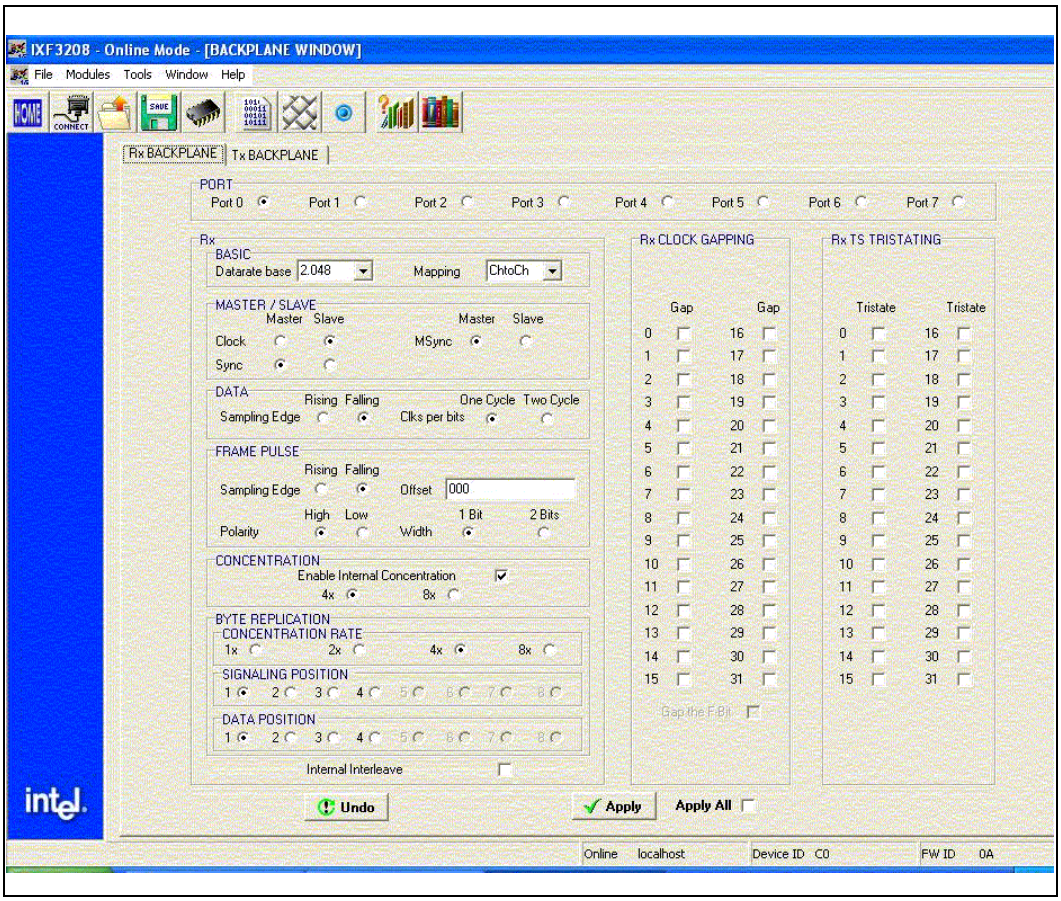
It can be seen in the port settings in the following figures that **Basic data rate** is set to **2.048 MHz** even though the Framer is being configured for T1. The reason for this is that Backplane clock and data rate uses the reference clock, which is a multiple of E1 (8.192 MHz), hence the backplane is configured in E1 and the PLLs are used to convert the E1 rate to T1; this is further described in Section 5.2.3. Ensure all ports are set to **2.048 MHz** base data rate on the Transmit and Receive side.

Also, since the backplane is configured at E1 rate, the T1 timeslots would need to be mapped into the E1 frame. To achieve this, Channel Mapping is used — a 1.544-Mbps T1 signal can be accommodated within the E1 structure in a channel-per-time-slot basis. The T1 channel TS0 carries the F-bit information. This is done by selecting **ChtoCh** mapping in **all ports** in the Transmit and Receive side. Refer to Backplane T1 to E1 Mapping section of the *Intel® IXF3208D Octal T1/E1/J1 Framer Datasheet* for further information.

FRAME PULSE, and configuration of the **Sampling Edges** should be matched with the configuration of the IXDP425 codelet. Signal generated on a falling edge on one device should be sampled on the rising edge of the other. If the IXF3208D is generating the signal (data or frame pulse), then the edge should be set to whatever it is set to on the IXDP425 (for example, falling-falling); if the IXF3208D is sampling the signal (**data or frame pulse**), the edge should be set to the opposite of the IXDP425 (for example, rising-falling). In both Transmit and Receive configurations, **Frame pulse widths** should be set to 1 bit, and the offset should be set to 000. **Data** should be set to **one** clock cycle per bit.

5.2.2.1 Rx Backplane

Figure 6. IXF3208D T1 Mode Configuration – Rx Backplane Module



Above is the configuration for Port 0. Port 1 to 3 should have **Sync** and **MSync** set to **Slave**, only **Master** for Port 0. Although **MSync** (multiframe sync) is not used it is important to have it set to **Master** for Port 0 so that the Framer does not expect an incoming signal.

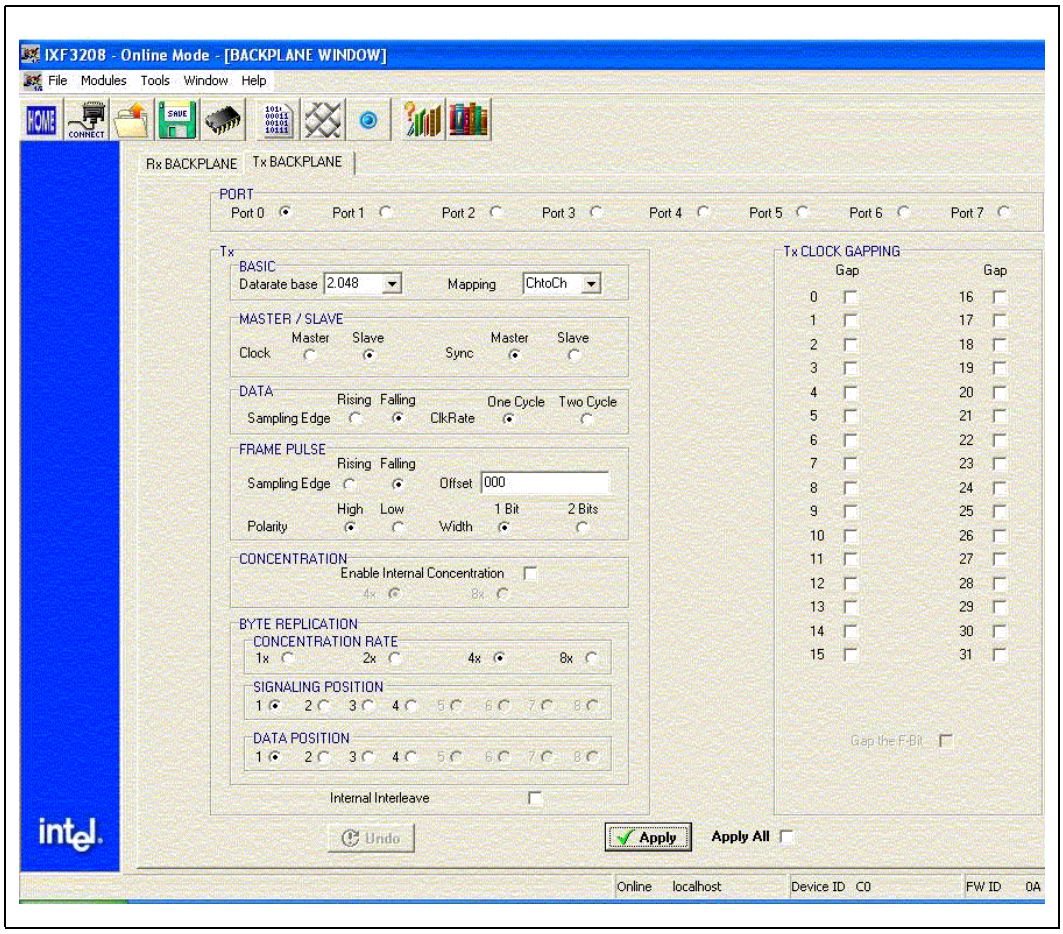
Port 1 should have its signaling and data positions set to 2, Port 2 should have its signaling and data positions set to 3, and Port 3 should have its signaling and data positions set to 4.

Use the same configuration as Port 0 to set up Port 4 and set Port 5-7 to the same signaling/data settings as Port 1-3.

Data and signaling positions are derived from the mechanics of the MVIP — on a port configured for MVIP there are effectively 128 (0 to 127) timeslots instead of 32. Setting Port 1 to a signaling and data position of 1 means that timeslots 32 to 63 are used for Port 1. In short, the position is the equivalent of an offset.

5.2.2.2 Tx Backplane

Figure 7. IXF3208D T1 Mode Configuration – Tx Backplane Module



There are few important points to note here too. Figure 7 shows the configuration for Port 0. Port 1 to 3 should have **Sync** set to **Slave**; only Port 0 should be set to **Master**.

Port 1 should have its signaling and data positions set to 2, Port 2 should have its signaling and data positions set to 3, and Port 3 should have its signaling and data positions set to 4.

5.2.3 Clock Selector Module

Clock selection should be identical for all eight ports configured. All timing information is concentrated in one selector module, where it is distributed to all other regions.

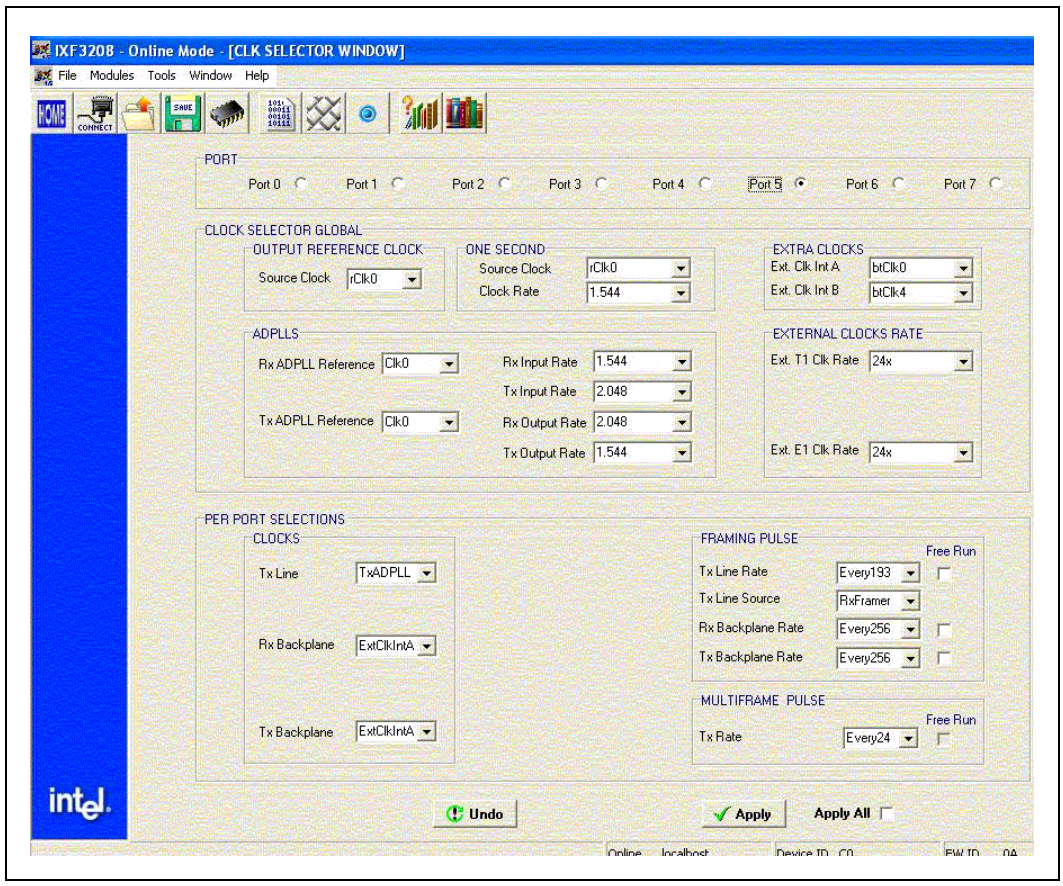
In this module, the Rx Backplane and Tx Backplane are configured to use the external clock, 8.192 MHz. The PLL is then used to convert the 4 x E1 clock rate into 4 x T1 for the line side.

The frame pulse configuration should match the configuration shown in Figure 8. As can be seen, the same principle applies here as well when converting E1 frames in to T1 on the Transmit and Receive side. In this instance, Multiframe Pulse is not used; therefore, its configuration is irrelevant.

Figure 8 shows the clock selector module; **all** port clock selectors should be configured identically.

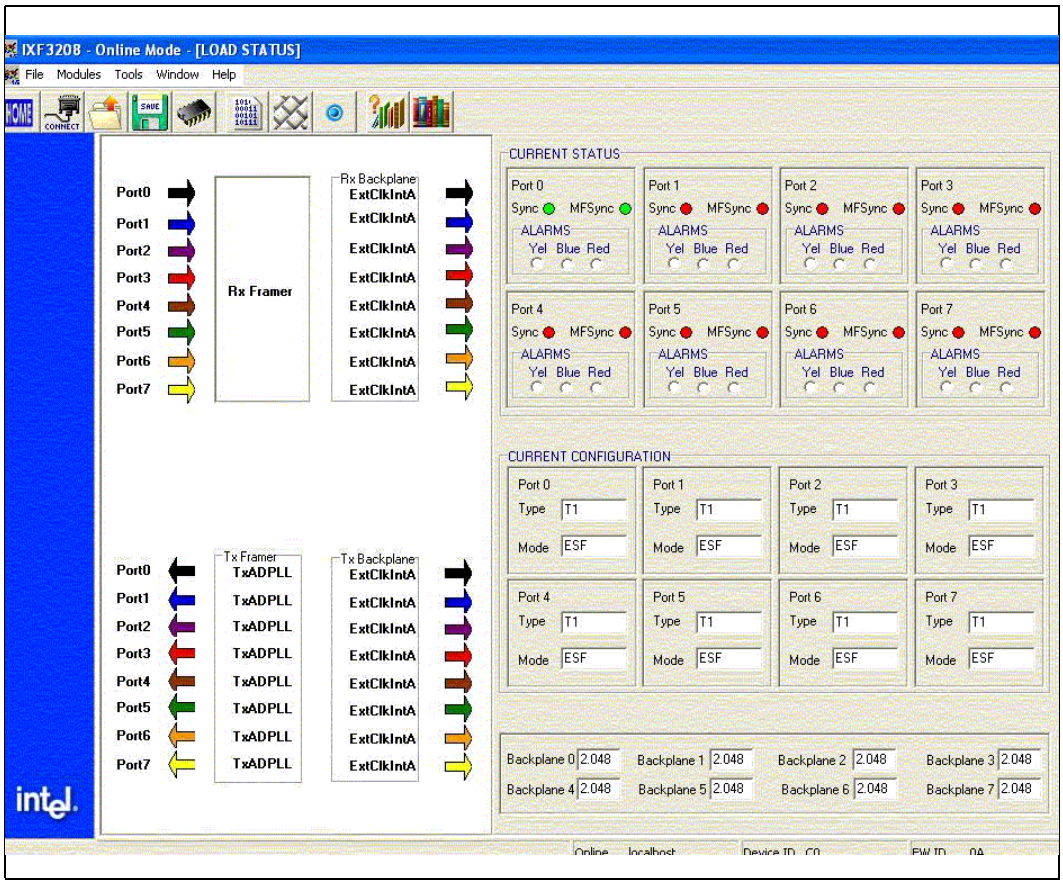
Refer to Timing Configurations section of the *Intel® IXF3208D Octal T1/E1/J1 Framer Datasheet* for further information regarding all the options available on this window.

Figure 8. IXF3208D T1 Mode Configuration – Clock Selector Module



Once configuration is complete, check the **Status** by clicking on the 'chip' icon. It should display the clock configuration shown in Figure 9.

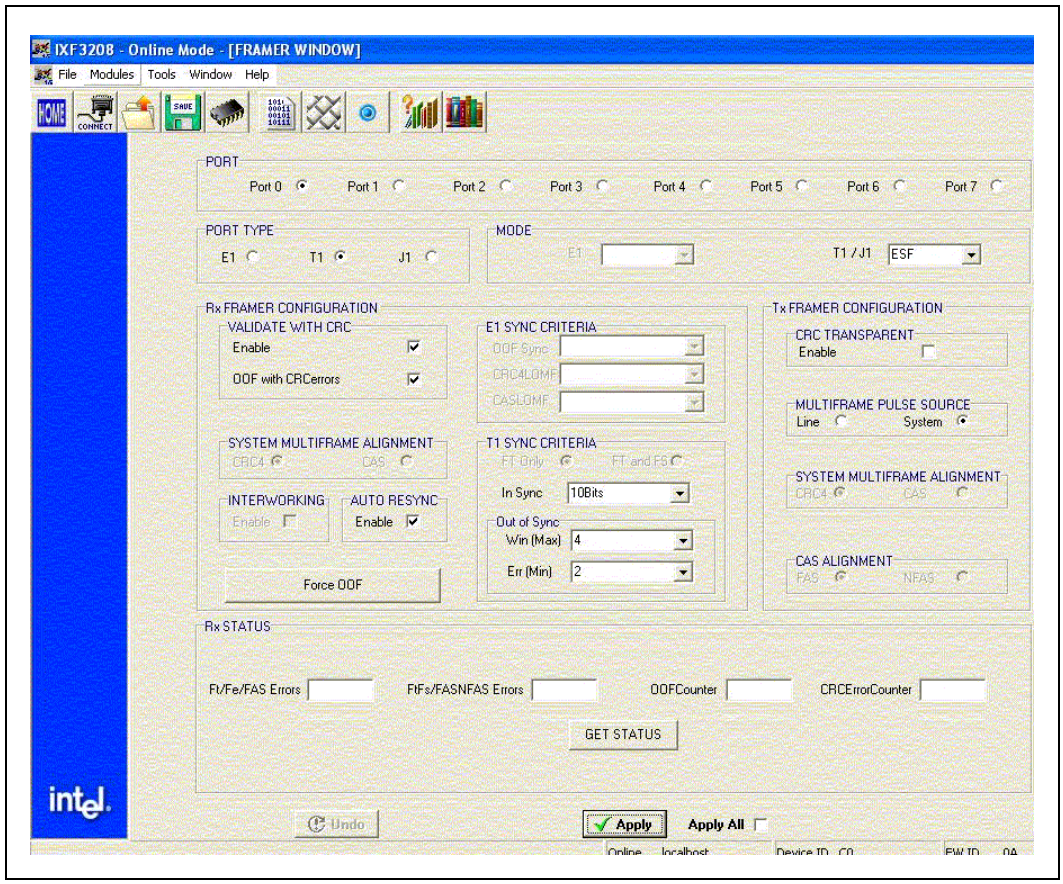
Figure 9. IXF3208D T1 Mode Configuration – Status Window



5.2.4 Framer Module

The Framer module should be configured according to Figure 10 for all ports; all options are self-explanatory.

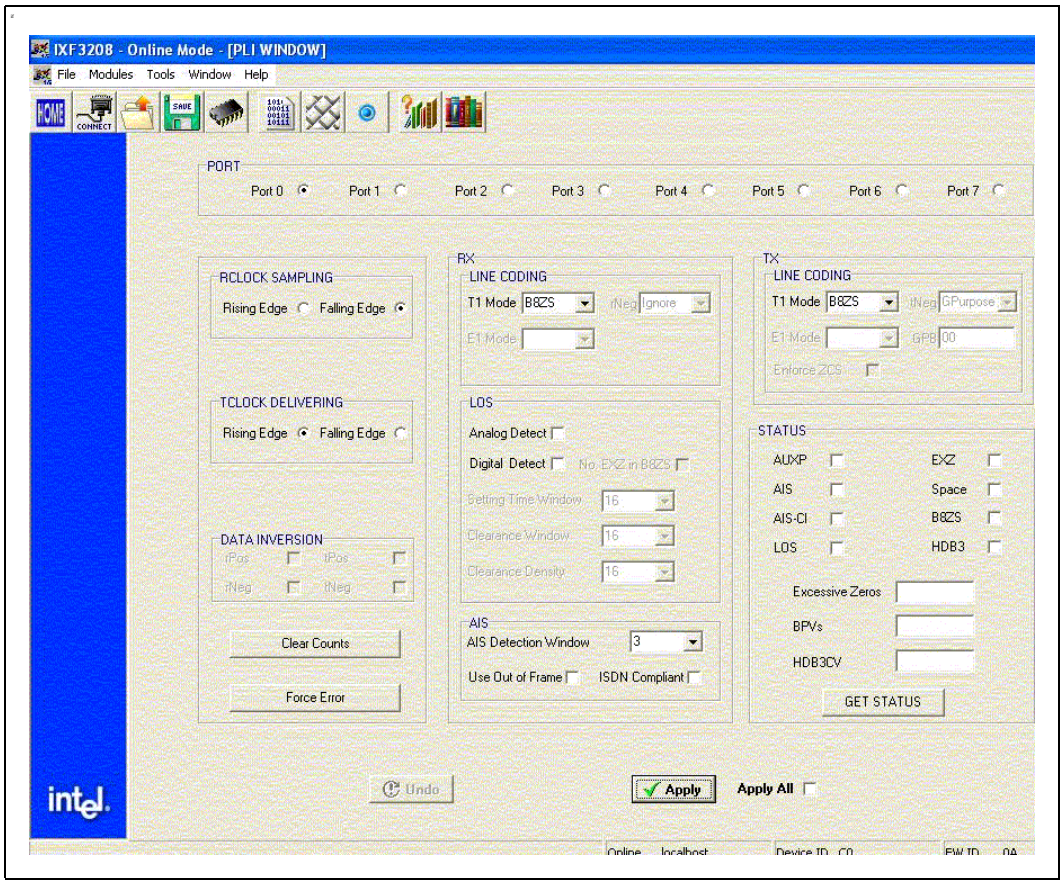
Figure 10. IXF3208D T1 Mode Configuration – Framer Module



5.2.5 PLI

Ensure **Rclock Sampling and Tclock Sampling** are set correctly. There are other options available for line encoding; however, B8ZS is chosen in this instance.

Figure 11. IXF3208D T1 Mode Configuration – PLI Module



6.0 Intel® IXF3208D Octal T1/E1/J1 Framer Configuration — E1 Mode

Using the GUI, first look at the configuration of LXT3108 LIU in E1 mode, then the configuration of the IXF3208D in E1 mode.

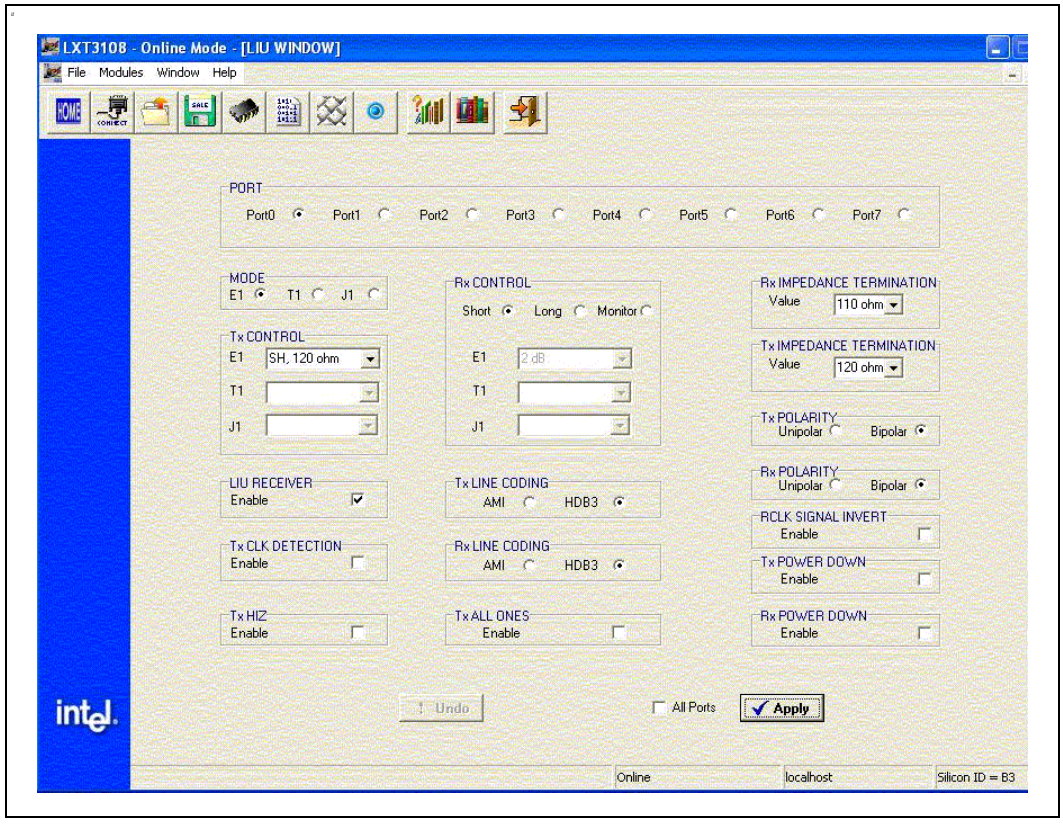
6.1 Intel® LXT3108 Line Interface Unit Configuration

1. Start ‘IXF3208D Application’.
   - Keep the Baud Rate settings as is, to 38400 on Serial Port COM1, unless using another COM port, and **START** the communication module with the serial port.
2. Launch the LXT3108 GUI and establish connection by clicking on **LXT3108 connection**. See **Online** and **Silicon ID = B3** at the bottom of the window; this confirms correct silicon stepping and communication with the silicon.

   Note: The user should be familiar with all the icons and features of the LXT3108 GUI.

3. Select **LIU** from Modules drop-down menu and configure **All ports** according to Figure 12.
   - Ensure **LIU RECEIVER** is enabled. Note that HDB3 encoding is used here.

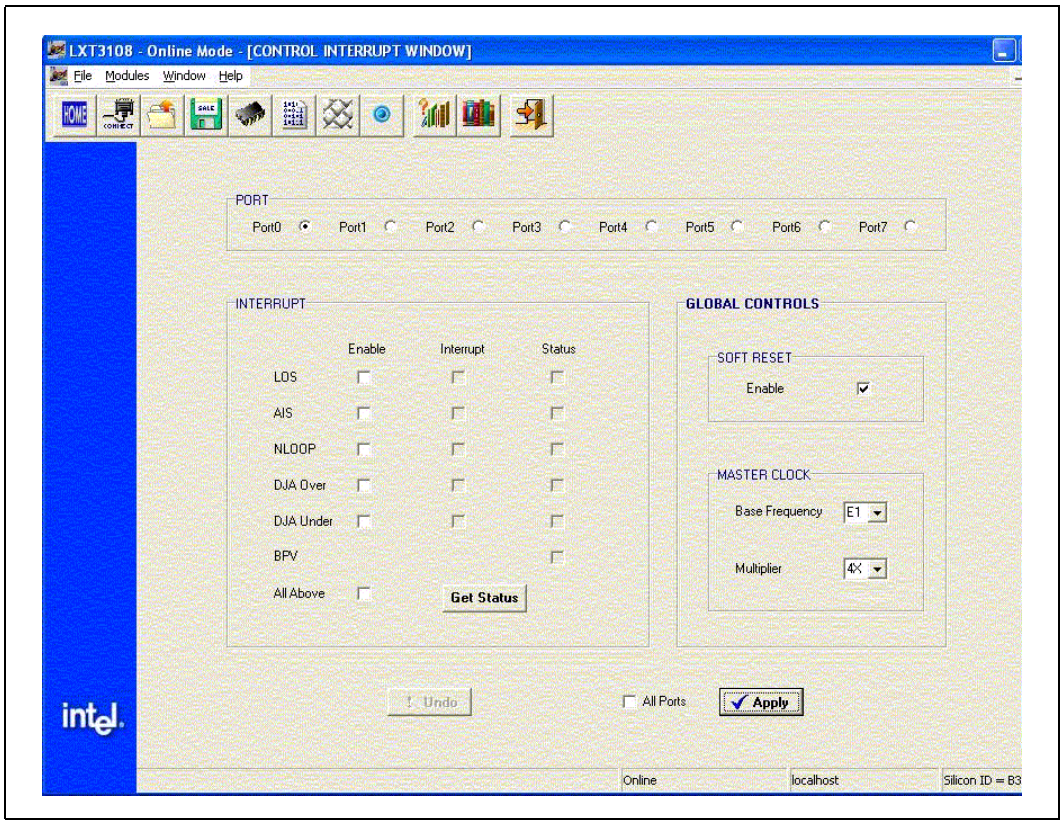
Figure 12. Intel® LXT3108 E1 Mode Configuration – LIU Module



6.1.1 Control Interrupt Module

In this module, ensure that the master clock base frequency is set to E1 and the multiplier is set to 4x.

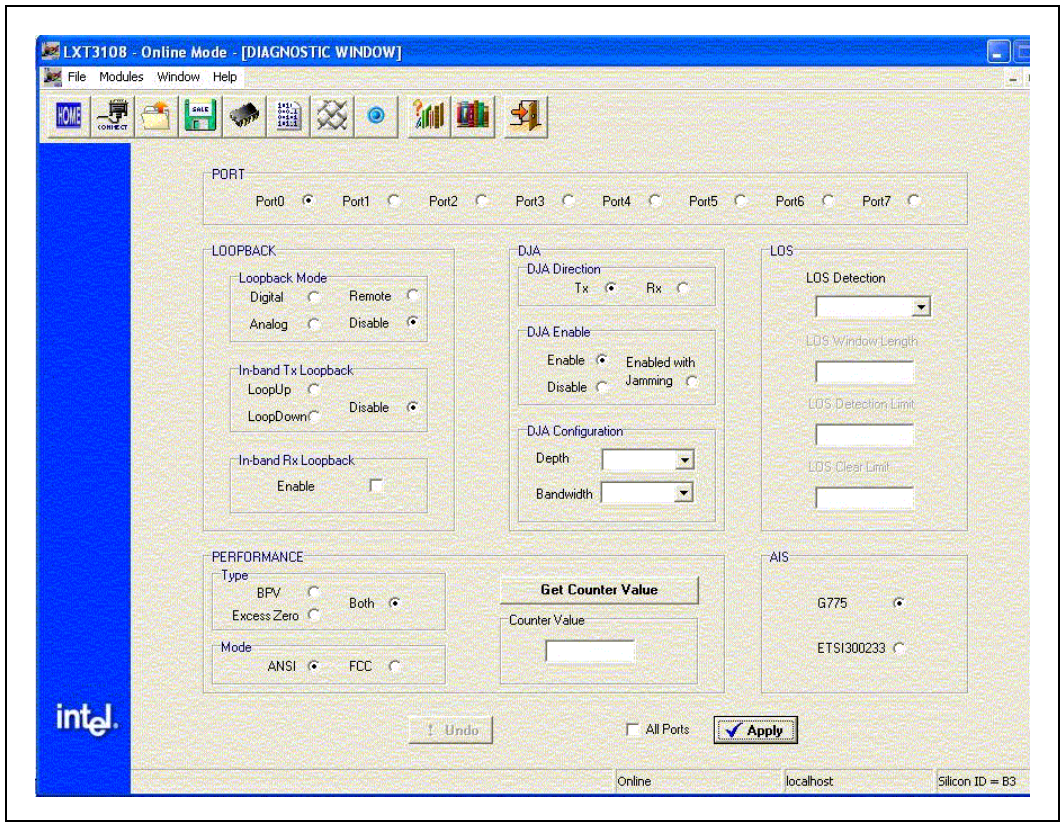
Figure 13. Intel® LXT3108 E1 Mode Configuration – Control Interrupt Module



6.1.2 Diagnostics Module

In this module, any kind of **Loopback** should be switched off. However, for testing purposes the Remote loopback can be enabled to check the line sync.

Figure 14. Intel® LXT3108 E1 Mode Configuration – Diagnostic Module



6.2 IXF3208D Configuration

In this section, most of the configuration is identical to the T1 setup. Framer setup for E1 is much simpler because the reference clock is a multiple of E1 line rate. Hence, no frame mapping, or use of PLL is required.

6.2.1 Alarms

Configure Alarm Module by clicking on **Alarm** from the **Modules** drop-down menu. Ensure all alarms are **Disabled for all ports**, especially **Alarms to Backplane**, as this is not be disabled by default.

6.2.2 Backplane

The Backplane Module is split into Receive and Transmit.

Note: "Transmit" and "Receive" refer to the device connected to the backplane (that is, the IXDP425); the **Transmit** configuration on the backplane of the IXF3208D (that is, the Framer) should match the **Transmit** configuration of the HSS port on the IXDP425.

The upcoming figures show the configuration for Port 0. Note that configurations for Ports 1 to 3 are **NOT** identical to Port 0; differences are explained in each section. Port 4 should be configured identical to Port 0 and Port 5-7 should be configured as Port 1-3.

Configuration of the edges should be matched with the configuration of the IXDP425; signal generated on a falling edge on one device should be sampled on the rising edge of the other. The configuration of the edges for the IXDP425 is straightforward (as can be seen in Section 7.1, "HSS Port Configuration" on page 32); however, the configuration of the IXF3208D a bit more complex. If the IXF3208D is generating the signal (data or frame pulse), then the edge should be set to whatever it is set to on the IXDP425 (for example, falling-falling); if the IXF3208D is sampling the signal (data or frame pulse), the edge should be set to the opposite of the IXDP425 (for example, rising-falling).

The **basic data rate** is set to **2.048 MHz** as the framer is being configured in E1. Since the backplane is configured at E1 rate, there is no mapping required.

In both Transmit and Receive configurations, frame pulse widths should be set to 1 bit, and data should be set to one clock cycle per bit.

6.2.2.1 Rx Backplane

Figure 15. IXF3208D E1 Mode Configuration – Rx Backplane Module

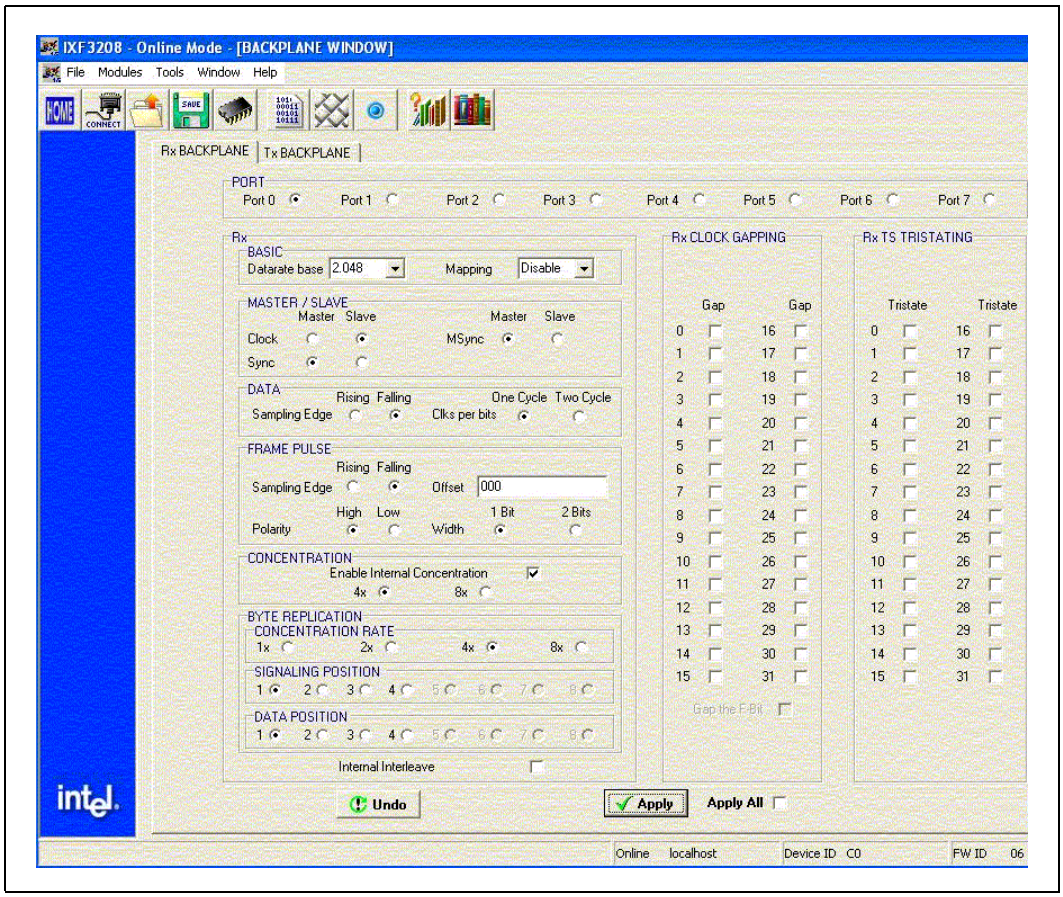


Figure 15 shows the configuration for Port 0. Port 1 to 3 should have **Sync and MSync** set to **Slave**, and only **Master** for Port 0. Although **MSync** (multiframe sync) is not used, it is important to have it set to **Master** for Port 0 so that the IXF3208D does not expect an incoming signal.

Port 1 should have its signaling and data positions set to 2, and Port 2 should have its signaling and data positions set to 3. Port 3 should have its signaling and data positions set to 4.

Use the same configuration as Port 0 to set up Port 4, and set Port 5-7 the same as Port 1-3.

Since HMVIP is multiplexed, Port 1 will place its timeslots in positions 1, 5, 9, 13, etc.

So, Port 0 will place its timeslots in positions 0, 4, 8, 12, 16, 20, 24, …, 124

Port 1 in positions 1, 5, 9, 13, 17, 21, 25, …125

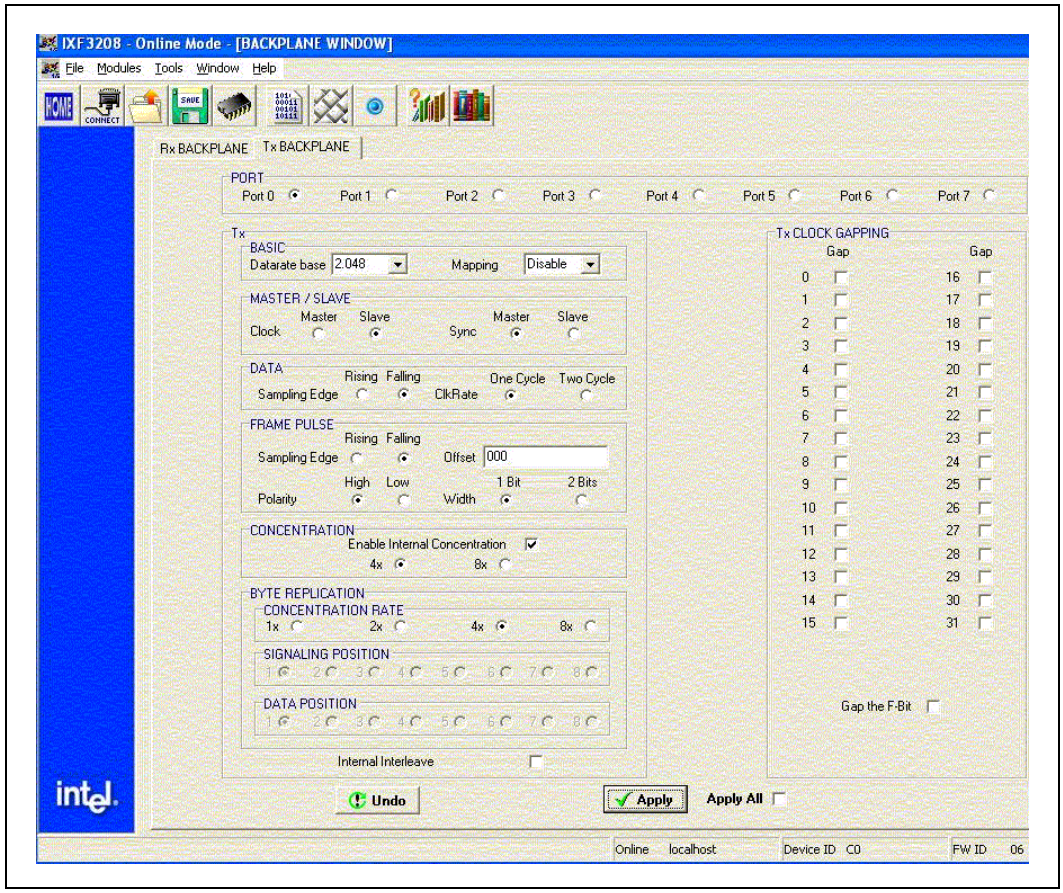
Port 2 in positions 2, 6, 10, 14, 18, 22, 26, ...126

Port 3 in positions 3, 7, 11, 15, 19, 23, 27, …127

Assuming we name the HMVIP timeslot numbers 0, 1, 2, 3, 4, 5, 6, 7, ..., 125, 126, 127

6.2.2.2 Tx Backplane

Figure 16. IXF3208D E1 Mode Configuration – Tx Backplane Module



There are few important points to note here. Figure 16 shows the configuration for Port 0. Port 1 to 3 should have Sync set to Slave, and only Port 0 should be set to master.

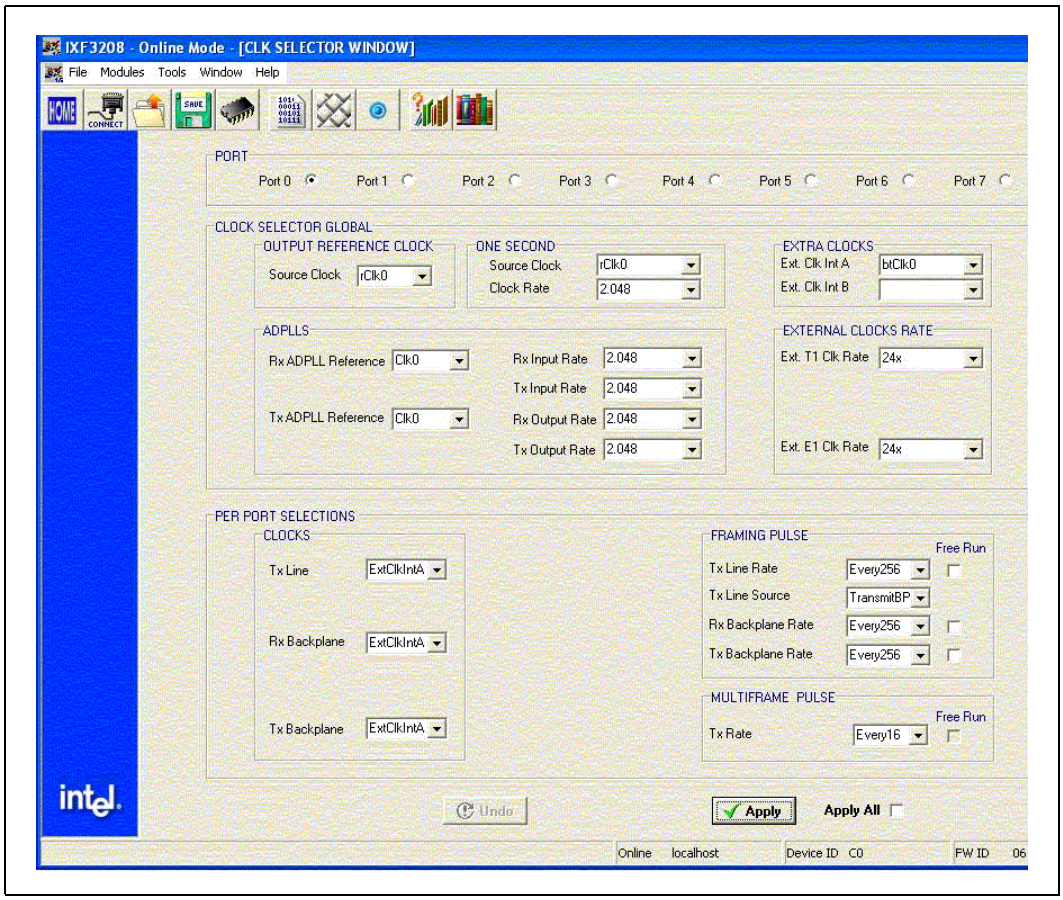
Port 1 should have its signaling and data positions set to 2, Port 2 should have its signaling and data positions set to 3, and Port 3 should have its signaling and data positions set to 4.

6.2.3 Clock Selection Module

Clock Selection should be identical for all eight ports configured. In this module, we define btClk0 (Clock on Port 0, transmit side) to be the reference clock ExtClkIntA. This reference clock is then used throughout the IXF3208D. The frame pulse configuration should match the configuration shown in Figure 17. In this instance, Multiframe Pulse is not used; therefore, its configuration is irrelevant.

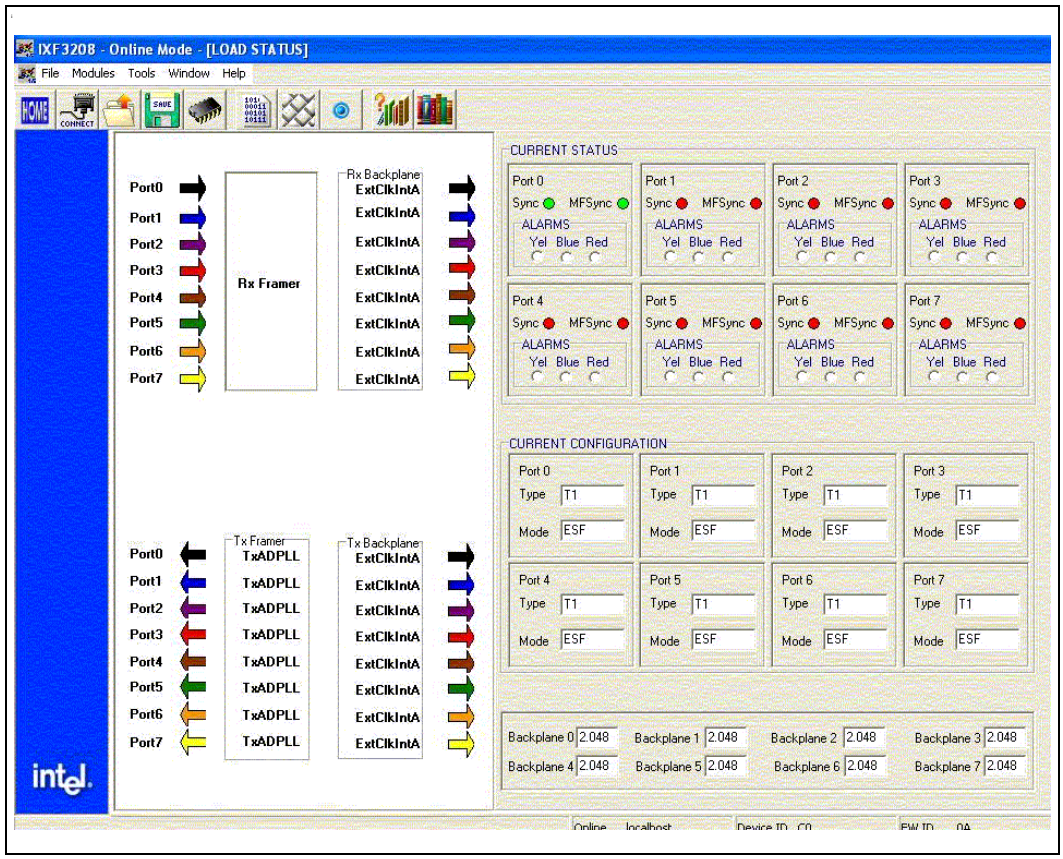
Also, as shown Figure 17, there are no PLL used in this configuration.

Figure 17. IXF3208D E1 Mode Configuration – Clock Selector Module



Once configuration is complete, check the status by clicking on the ‘chip’ icon. It should display the clock configuration shown in Figure 18.

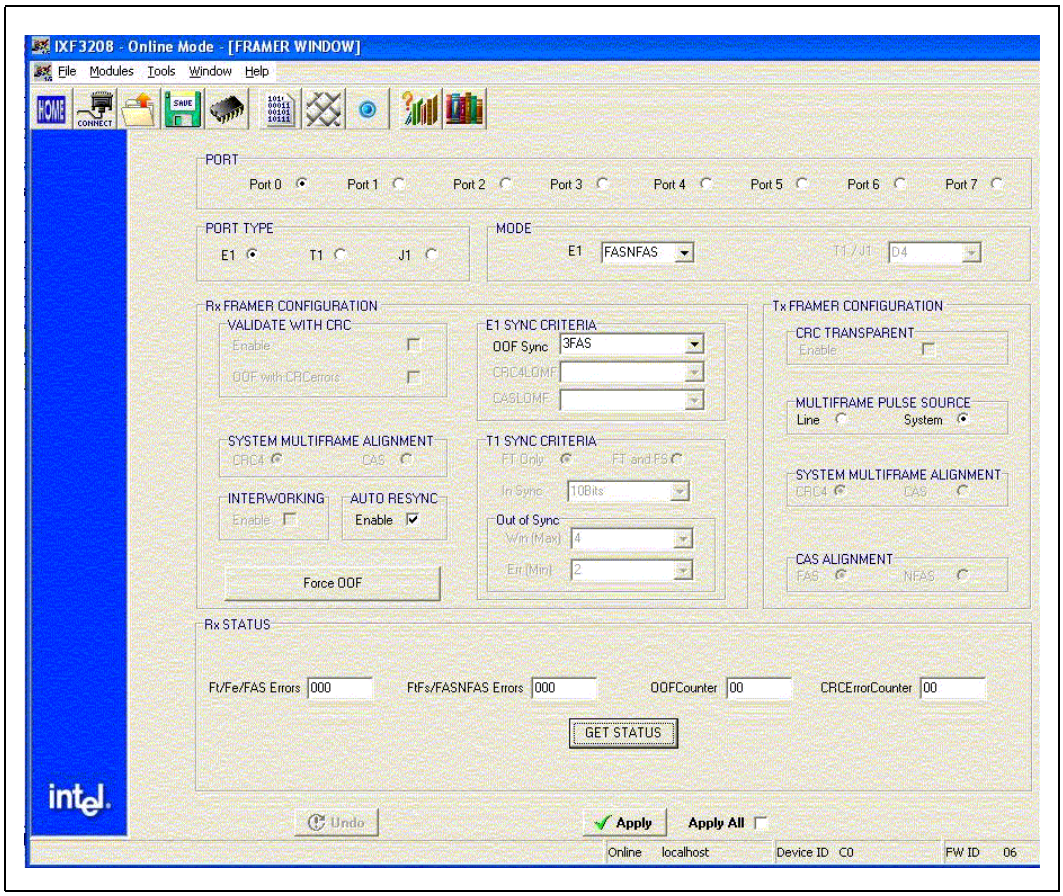
Figure 18. IXF3208D E1 Mode Configuration – Status Window



6.2.4 Framer Module

Port Type should be set to E1; framer mode should be set to FASNFAS, depending on the type of encoding required.

Figure 19. IXF3208D E1 Mode Configuration – Framer Module



7.0 Intel® IXDP425 / IXCDP1100 Development Platform Configuration

As mentioned in Section 3.3, the IXDP425 runs MontaVista Pro 3.0 on RedHat Linux 7.3. The software used for the reception and transmission of the traffic is based on the hssAcc codelet from the Intel® IXP400 Software v.1.3.

7.1 HSS Port Configuration

To establish communication with the IXF3208D, the HSS port has to be configured according to the framer settings. After unzipping the IXP400 software, the source code for the codelet can be found under *ixp425_xscale_sw\src\codelets\hssAcc*; to change the configuration, certain settings in the *IxHssAccCodeletConfig.c* file need to be modified.

The following listing shows the values used to match the framer configuration:

```
static IxHssAccPortConfig txHssPortConfig =
{
```

```
        /* Frame sync type = active low */
        IX_HSSACC_FRM_SYNC_ACTIVE_LOW,
        /* Frame sync output enable = input */
        IX_HSSACC_FRM_SYNC_INPUT ,
        /* Frame sync clock edge = falling */
        IX_HSSACC_CLK_EDGE_FALLING,
        /* Data clock edge = rising */
        IX_HSSACC_CLK_EDGE_RISING,
        /* Clock direction = input */
        IX_HSSACC_SYNC_CLK_DIR_INPUT,
        /* Frame usage = enabled */
        IX_HSSACC_FRM_PULSE_ENABLED,
        /* Data rate = clock rate */
        IX_HSSACC_CLK_RATE,
        /* Data polarity = same */
        IX_HSSACC_DATA_POLARITY_SAME,
        /* Data endianness = msb endian */
        IX_HSSACC_MSB_ENDIAN,
        /* Drain mode = normal */
        IX_HSSACC_TX_PINS_NORMAL,
        /* FBit usage = data */
        IX_HSSACC_SOF_FBIT,
        /* Data enable = data */
        IX_HSSACC_DE_DATA,
        /* 56K type = low */
        IX_HSSACC_TXSIG_LOW,
        /* Unassigned type = low */
        IX_HSSACC_TXSIG_LOW,
        /* FBit type = fifo */
        IX_HSSACC_FB_FIFO,
        /* 56K endianness = bit 7 unused */
        IX_HSSACC_56KE_BIT_7_UNUSED,
        /* 56K selection = 32/8 data */
        IX_HSSACC_56KS_32_8_DATA,
        /* Frame offset = 0 */
        0,
        /* Frame size = 1024 (4 trunks * 32 timeslots * 8 bits) */
        1024,
    };
```

```
static IxHssAccPortConfig rxHssPortConfig =
{
    /* Frame sync type = active low */
    IX_HSSACC_FRM_SYNC_ACTIVE_HIGH,
    /* Frame sync output enable = input */
    IX_HSSACC_FRM_SYNC_INPUT ,
    /* Frame sync clock edge = falling */
    IX_HSSACC_CLK_EDGE_FALLING,
    /* Data clock edge = falling */
    IX_HSSACC_CLK_EDGE_FALLING,
    /* Clock direction = input */
    IX_HSSACC_SYNC_CLK_DIR_INPUT,
    /* Frame usage = enabled */
    IX_HSSACC_FRM_PULSE_ENABLED,
    /* Data rate = clock rate */
    IX_HSSACC_CLK_RATE,
    /* Data polarity = same */
    IX_HSSACC_DATA_POLARITY_SAME,
    /* Data endianness = msb endian */
    IX_HSSACC_MSB_ENDIAN,
    /* Drain mode = normal */
    IX_HSSACC_TX_PINS_NORMAL,
    /* FBit usage = data */
    IX_HSSACC_SOF_FBIT,
    /* Data enable = data */
    IX_HSSACC_DE_DATA,
    /* 56K type = low */
    IX_HSSACC_TXSIG_LOW,
    /* Unassigned type = low */
    IX_HSSACC_TXSIG_LOW,
    /* FBit type = fifo */
    IX_HSSACC_FB_FIFO,
    /* 56K endianness = bit 7 unused */
    IX_HSSACC_56KE_BIT_7_UNUSED,
    /* 56K selection = 32/8 data */
    IX_HSSACC_56KS_32_8_DATA,
    /* Frame offset = 0 */
    0,
    /* Frame size = 1024 (4 trunks * 32 timeslots * 8 bits) */
    1024,
```

```
};
```

As we are using an external 8.192-MHz clock, we need to set the receive and transmit clock for the HSS port to input. The same is true for the frame sync, which the IXDP425 receives from the receive and transmit side of the IXF3208D backplane.

For the transmission, data should be generated on the rising edge, the frame pulse should be sampled on the falling edge and set to high polarity to match the IXF3208D configuration.

On the receive side, data should be sampled on the falling edge and the frame pulse should be high polarity and sampled on the falling edge.

For the TDM configuration, we have unassigned timeslot 0 and assigned timeslots 1-31 to TS_VOICE56k. This setup corresponds to four T1 clients per interface, as required. Accordingly, the number of channels for the channelized operation should be set to 32 by using the pre-defined constant, which can be modified in the file *IxHssAccCodeletChan.h*:

configParams.numChannelised = IX_HSSACC_CODELET_CHAN_NUM_CHANS;

These settings will activate a PHY loopback for both HSS ports. To use HDLC data, the packetized service needs to be configured and the TDM map modified.

Once the codelet has been modified and compiled on the host system, it needs to be downloaded to the target. At the Linux command prompt, it needs to be started with the following parameters:

Insmod ixp400_codelets_hssAcc.o operationType=1 portMode=3 verifyMode=1

This will start the channelized service for both HSS ports and doesn't let the code verify the outgoing data. If the run time of 10 seconds is not long enough, it can be changed in the file *IxHssAccCodelet.h* by modifying the value for `IX_HSSACC_CODELET_DURATION_IN_MS`.

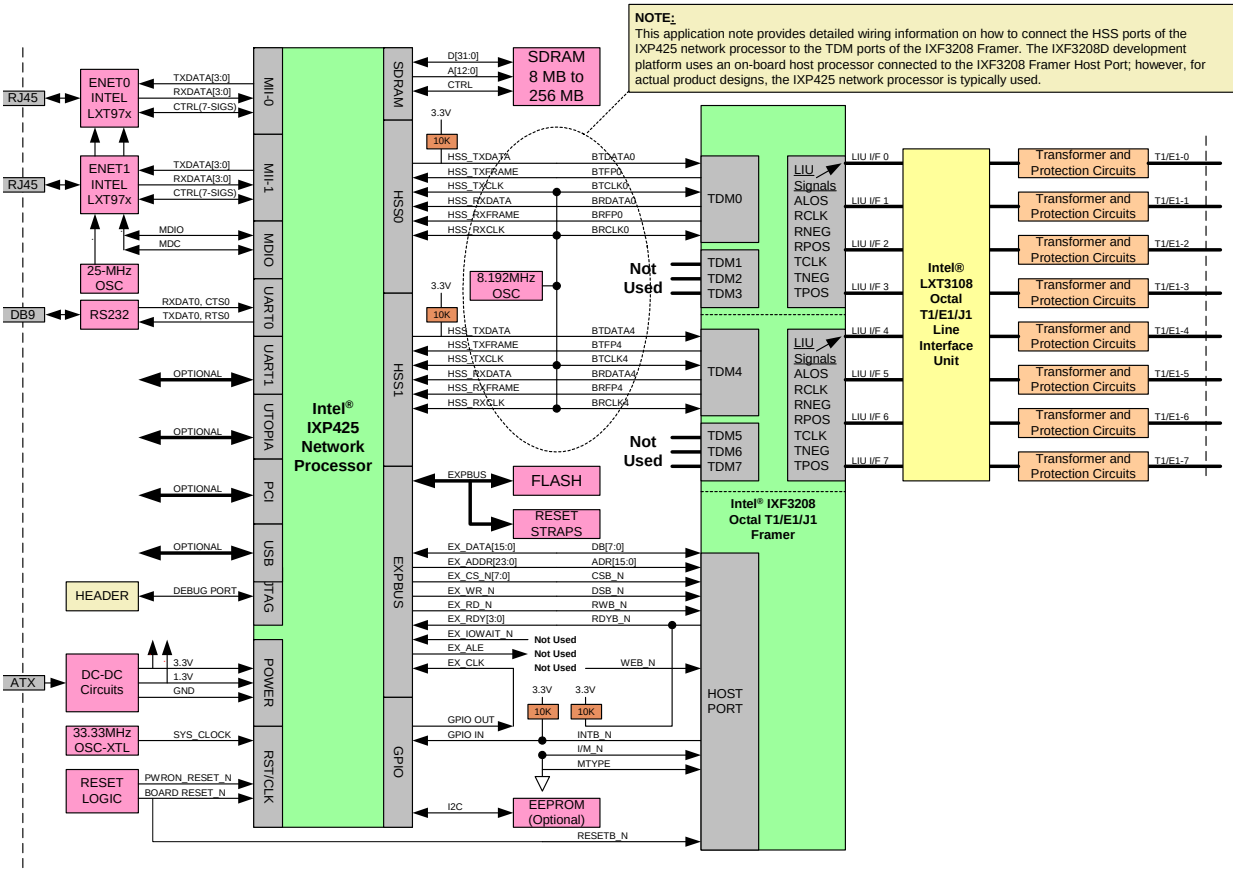
After the pre-defined running time, the screen should show statistics of how many packets/bytes have been received and set by the two HSS interfaces.

8.0 Interfacing Intel® IXP425 Network Processor to the Intel® IXF3208 Framer in a Product Design

Until now, this application note has focused on the electrical and software connections of the IXDP425 to the IXF3208D platform. Although this setup is ideal for enabling early software development and validating product designs with different T1/E1 backplane protocols, there are many components across both platforms that are not necessary for a final design. This section covers additional design considerations for interfacing the IXP425 network processor to the IXF3208 framer in an actual product design.

Figure 20 shows all 'key' signal interfaces typically required between the IXP425 network processor and the IXF3208 framer. Further detail on these interfaces is provided in following sections.

Figure 20. Key Signal Interfaces



NOTES: Figure 20 only shows detailed signal connections between the IXP425 network processor and IXF3208 framer. Depending on end-product requirements, optional IXP425 network processor interfaces may by used.

Each IXP425 Network Processor HSS Port connects to one IXF3208 framer TDM port, transporting quad T1/E1 payloads per HSS/TDM port. Using two HSS and two TDM ports yields a total of octal T1/E1 payloads. The Rx and Tx clocks for both HSS and TDM ports must be generated using an external 8.192-MHz clock source.

All IXF3208 framer internal registers and RAM are accessed using the IXP425 Network Processor Expansion Bus connected to the IXF3208 Framer Host Port Interface; a software API for the IXF3208 framer is available. Configuration of the IXF3208 framer registers requires set up of many parameters, such as enabling of both TDM0 and TDM4 interfaces to run MVIP protocol to support transport of quad T1/E1 per TDM (for further details, see related application notes at http://www.intel.com/design/network/products/wan/docs/ixf3208_docs.htm#applnot).

The IXP425 Network Processor Expansion Bus is configured as Motorola* Mode, non-multiplexed, lower eight bits used for the IXF3208 framer, using one CS and one RDY signal. Expansion bus cycles are configured to accommodate timing requirements of the IXF3208 Framer Host Port.

Although the IXF3208 framer is shown, the pin-compatible Intel® IXF3204 T1/E1/J1 Quad Framer can also be used for quad T1/E1 applications; when using the IXF3204, the TDM4 port and the IXP425 Network Processor HSS1 port are **not** used.

8.1 IXF3208 Framer Host Port Interface Summary

An on-board CPU, the Intel® i386™ EX microprocessor, is used for hosting the IXF3208 framer. For actual product designs, the function of this CPU is typically replaced by the IXP425 network processor, which acts as the host.

The IXF3208 Framer Host Port Interface allows a host processor; in this case, the IXP425 network processor, to access the IXF3208 framer internal memory map. The IXF3208 framer memory map is organized into four main sections:

1. Global registers; these contain the information of the whole eight ports regarding reset, interrupts, masks, and status.
2. Per-port access; this is where each port is mapped such that the user adds an offset to a base address to access similar information on each port.
3. Functional modules; this is where the HDLC and BERT are mapped.
4. Internal processor instruction and data memory; these are combined into one section.

Access to the IXF3208 framer memory map allows host applications to perform functions such as IXF3208 framer reset, configuration, modification, monitoring of T1/E1 characteristics and performance measurements. Further details of the IXF3208 framer internal memory map, including configuration registers and software drivers are contained in a separate document titled the *Intel® IXF3208 Octal T1/E1/J1 Framer with Intel® On-Chip PRM Memory Map Software Developer Manual*. This information is not repeated in this application note, whereas the details of connecting the IXP425 Network Processor Expansion Bus to the IXF3208 Framer Host Port are provided in the following sections.

8.2 IXP425 Network Processor Expansion Bus Connection to the IXF3208 framer Host Port

The IXP425 Network Processor Expansion Bus is used as the electrical interface to the IXF3208 Framer Host Port as shown in Figure 20. To keep this interface simple and glueless, both the IXP425 Network Processor Expansion Bus and IXF3208 Framer Host Port are configured to operate in Motorola* Mode, using non-multiplexed address and data format.

For the IXP425 Network Processor Expansion Bus, only the lower eight bits of the Data bus are used, and only the lower 16 bits of the Address Bus are used. The IXF3208 framer also has provisions for a Ready signal, which directly interfaces to the IXP425 network processor Ready signal, providing automatic cycle stretching (or pacing), depending on what internal IXF3208 framer memory map accesses are initiated.

After the IXP425 Network Processor Expansion Bus is configured, software developers can begin implementing and integrating the available IXF3208 framer API for an end product. This IXF3208 framer–based API allows software developers to write software applications that interact with the IXF3208 framer without having to learn all internal details of the IXF3208 framer control registers. Interfacing with the IXF3208 framer through the API expedites development and can reduce time to market.

Table 3 provides a summary of all IXP425 Expansion Bus signals and IXF3208 framer Host Port signals.

Table 3. IXP425 Network Processor Expansion Bus and IXF3208 Framer Host Port Signal Descriptions (Sheet 1 of 3)

IXP425 Signal	IXP425 Description	IXF3208 Signal	IXF3208 Description	Notes
EX_DATA[15:0]	Expansion-bus, bidirectional data	DB[7:0]	Data Bus. The 8-bit DB signals are used to transfer data through read or write transactions. **NOTE:** A DB signal enters a tristate either when there is no data transfer or when the data transfer is a write process.	Only the lower eight bits of the data bus are used. The IXP425 network processor can be configured to interface with the IXF3208 in byte mode.
EX_ADDR[23:0]	Expansion-bus address used as an output for data accesses over the expansion bus.	ADR[15:0]	Address. The 16-bit ADR address lines are used to read or write any internal register or RAM region. For details, see the memory map document for the IIXF3208.	Only the lower 16 bits of the address bus are used by the IXF3208. The upper address bits are decoded internal to the IXP425 network processor to generate the proper address decoding for the CS_N signal.
EX_CS_N[7:0]	External chip selects for expansion bus. • Chip selects 0 through 7 can be configured to support Intel or Motorola* bus cycles. • Chip selects 4 through 7 can be configured to support TI HPI bus cycles.	CSB_N	Chip Select. (Active Low.) When CSB is low, it selects the IXF3208 framer, allowing it to perform read/write operations.	Typically tied to any one of the eight IXP425 network processor EX_CS_N output signals.
EX_WR_N	Intel-mode write strobe OR Motorola-mode data strobe (EXP_MOT_DS_N) OR TI*-mode data strobe (TI_HDS1_N).	DSB_N	Data Strobe. (Active Low.) Depending on the processor being used, DSB operates in a different way in the modes supported by the three processors.	Direct connection because both devices are configured in Motorola mode
EX_RD_N	Intel-mode read strobe OR Motorola-mode read-not-write (EXPB_MOT_RNW) OR TI mode read-not-write (TI_HR_W_N).	RWB_N	Read / Write. (Active Low.) RWB distinguishes between read and write operations. • 0 = Write • 1 = Read	Direct connection because both devices are configured in Motorola mode

Table 3. IXP425 Network Processor Expansion Bus and IXF3208 Framer Host Port Signal Descriptions (Sheet 2 of 3)

IXP425 Signal	IXP425 Description	IXF3208 Signal	IXF3208 Description	Notes
EX_RDY[3:0]	HPI interface ready signals. Can be configured to be active high or active low. These signals are used to halt accesses using Chip Selects 7 through 4 when the chip selects are configured to operate in HPI mode. There is one RDY signal per chip select. This signal only affects accesses that use EX_CS_N[7:4].	RDYB_N	Ready Signal. (Active Low, Open Drain.) • When a data transfer to the IXF3208 is selected, RDYB is set to ‘1’. • When the data transfer is complete, RDYB is set to ‘0’. • When the host processor removes CSB (CSB = ’1’), then RDYB goes to a tristate.	Tied to 1 of the 4 IXP425 network processor EX_RDY input signals. This signal is pulled high because it is an open drain.
GPIO_IN	General Purpose Input/ Output pins. May be configured as an input or an output. As an input, each signal may be configured a processor interrupt. Default after reset is to be configured as inputs.	INTB_N	Hardware Interrupt Output, Open Drain. When INTB is not asserted, it is in a tristate. When the I/M# ball sets the processor mode for a: • Motorola processor, INTB is active in the ‘0’ state. • Intel® processor, INTB is active in the ‘1’ state.	Typically tied to 1 of the 16 IXP425 network processor GPIO pins, configured as an interrupt input. This signal is pulled high because it is an open drain.
EX_IOWAIT_N	Data ready/acknowledge from expansion-bus devices. Expansion-bus access is halted when an external device sets EX_IOWAIT_N to logic 0 and resume from the halted location once the external device sets EX_IOWAIT_N to logic 1. This signal affects accesses that use EX_CS_N[7:0] when the chip select is configured in Intel- or Motorola-mode of operation.	N/A	N/A	Not used because the IXP425 network processor is using the EX_RDY signal instead.
EX_ALE	Address-latch enable used for multiplexed address/data bus accesses. Used in Intel and Motorola multiplexed modes of operation.	N/A	N/A	Not Used by the IXF3208 because IXP425 network processor is set-up to be non-multiplexed mode
EX_CLK	Input clock signal used to sample all expansion interface inputs and clock all expansion interface outputs.	N/A	N/A	This IXP425 network processor clock input is driven by a GPIO configured as an output clock

Table 3. IXP425 Network Processor Expansion Bus and IXF3208 Framer Host Port Signal Descriptions (Sheet 3 of 3)

IXP425 Signal	IXP425 Description	IXF3208 Signal	IXF3208 Description	Notes
N/A	N/A	WEB_N	Write Enable (active low for Motorola MPC860* processor.) Depending on the processor being used, WEB indicates a write operation when CSB is active, and the IXF3208 loads the internal register with data provided on the data bus.	Not used because the IXF3208 is configured in Motorola 68K mode, for which this signal becomes a don't care.
N/A	N/A	I/M_N	Intel® processor / Motorola processor. I/M# selects a processor interface mode as follows: • I/M# = 0 selects a Motorola processor. • I/M# = 1 selects an Intel processor.	Pulled low to select Motorola Processor mode
N/A	N/A	MTYPE	Motorola processor types. MTYPE selects a Motorola processor type as follows: • MTYPE = 0 selects from the family of Motorola PowerPC* processors (such as the Motorola MPC860 processor) • MTYPE = 1 selects from the family of Motorola 68K processors (such as the Motorola 68302 processor)	Pulled low to select Motorola 68K Processor mode
N/A	N/A	RESETB_N	Master Hardware Reset. (Active Low.) When low, RESETB resets the IXF3208 hardware. RESETB has an internal pull-up resistor.	Typically tied to system- or board-reset signal.

8.3 IXP425 Network Processor Expansion Bus Configuration

Since the IXF3208 framer resides on the Expansion Bus, the IXP425 network processor simply views this device as a memory-mapped device. There are no IXP425 network processor-related APIs required to provide this access-layer support. To define the expansion bus parameters for accessing this device, only one IXP425 network processor register needs to be configured after power-on-reset.

The IXP425 network processor supports a total of eight chip select signals: EX_CS[7:0]. The Expansion Bus device (in this case, the IXF3208 framer) is typically assigned to only one chip select signal. The IXP425 network processor allows the user to configure each of the eight chip select signals differently, depending on the devices to which they are attached. Therefore, each chip select has its own configuration register, as defined in Table 4.

Table 4. IXP425 Network Processor Chip Select Signal Configuration

Address	R/W	Name	Description
0xC4000000	R/W	EXP_TIMING_CSO	Timing and Control Register for Chip Select 0
0xC4000004	R/W	EXP_TIMING_CS1	Timing and Control Register for Chip Select 1
0xC4000008	R/W	EXP_TIMING_CS2	Timing and Control Register for Chip Select 2
0xC400000C	R/W	EXP_TIMING_CS3	Timing and Control Register for Chip Select 3
0xC4000010	R/W	EXP_TIMING_CS4	Timing and Control Register for Chip Select 4
0xC4000014	R/W	EXP_TIMING_CS5	Timing and Control Register for Chip Select 5
0xC4000018	R/W	EXP_TIMING_CS6	Timing and Control Register for Chip Select 6
0xC400001C	R/W	EXP_TIMING_CS7	Timing and Control Register for Chip Select 7
0xC4000020	R/W	EXP_CNFG0	General-purpose Configuration Register 0
0xC400024	R/W	EXP_CNFG1	General-purpose Configuration Register 1
0xC400028	–	–	Reserved

The details of each register bit are defined in the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual* and are repeated in Table 5. These registers provide configuration fields/bits for Expansion Bus parameters such as chip select enable, address space, cycle timing, cycle types, byte or word mode, etc.

EX_CS[4] is assigned for chip-selecting the IXF3208 framer. This chip select is chosen to correspond with use of the EX_RDY[0], respectively. Therefore, only the EXP_TIMING_CS4 register needs to configured after power-on-reset per the recommendations (highlighted in **bold**) in Table 5.

Table 5. IXP425 Network Processor Chip Select Register Bit Description (Sheet 1 of 2)

Bits	Name	Description
31	CSx_EN	0 = Chip Select x disabled **1 = Chip Select x enabled**
30	RESERVED	
29:28	T1 – Address timing	**00 = Generate normal address phase timing** 01 - 11 = Extend address phase by 1 - 3 clocks
27:25	T2 – Setup / Chip Select Timing	**00 = Generate normal setup phase timing** 01 - 11 = Extend setup phase by 1 - 3 clocks
25:22	T3 – Strobe Timing	0000 = Generate normal strobe phase timing 0001-1111 = Extend strobe phase by 1 - 15 clocks **0110 = Recommend 6 clock extension = 7 x 15 ns = 105 ns**
21:20	T4 – Hold Timing	**00 = Generate normal hold phase timing** 01 - 11 = Extend hold phase by 1 - 3 clocks

Table 5. IXP425 Network Processor Chip Select Register Bit Description (Sheet 2 of 2)

Bits	Name	Description
19:16	T5 – Recovery Timing	**0000 = Generate normal recovery phase timing** 0001-1111 = Extend recovery phase by 1 - 15 clocks
15:14	CYC_TYPE	00 = Configures expansion bus for Intel cycles. **01 = Configures expansion bus for Motorola cycles.** 10 = Configures expansion bus for HPI cycles. 11 = Reserved
13:10	CNFG[3:0]	Device Configuration Size. Calculated using the formula: SIZE OF ADDR SPACE = 2(9+CNFG[3:0]) For Example: 0000 = Address space of 2(9) = 512 bytes **0111 = Address space of 2(16) = 64 Kbytes** 1000 = Address space of 2(17) = 128 Kbytes 1111 = Address space of 2(24) = 16 Mbytes
9:7	RESERVED	
6	BYTE_RD16	Byte read access to half-word device **0 = Byte access disabled.** 1 = Byte access enabled.
5	HRDY_POL	HPI HRDY polarity (reserved for exp_cs_n[7:4] only) **0 = Polarity low true.** 1 = Polarity high true.
4	MUX_EN	**0 = Separate address and data buses.** 1 = Multiplexed address / data on data bus.
3	SPLT_EN	**0 = AHB split transfers disabled.** 1 = AHB split transfers enabled.
2	RESERVED	
1	WR_EN	**0 = Writes to CS region are disabled.** 1 = Writes to CS region are enabled.
0	BYTE_EN	0 = Expansion bus uses 16-bit-wide data bus. **1 = Expansion bus uses only 8-bit data bus.**