PolarFire 10G SyncE Solution with ESMC and Jitter Attenuating PLL

AN5466

Introduction (Ask a Question)

This application note describes how the Synchronous Ethernet (SyncE) technology provides a solution for transporting synchronization over Ethernet. SyncE uses frequency synchronization from the physical layer together with a slow protocol and has been standardized by the ITU's Telecommunication Standardization Sector (ITU-T) for frequency distribution. SyncE leverages the PHY layer of Ethernet to transmit frequency to remote sites or other nodes.

SyncE is implemented according to the following ITU-T recommendations:

- G.8261: Defines the architecture of Synchronous Ethernet networks
- G.8262: Specifies the timing characteristics of synchronous Ethernet Equipment Clock (EEC)
- G.8264: Describes the Ethernet Synchronization Messaging Channel (ESMC) which is used to transfer the clock information in the form of Synchronous Status Message (SSM).

PolarFire® Gigabit Transceiver in conjunction with Jitter Attenuating Phase Lock Loop (JAPLL) provides Synchronous Ethernet (SyncE) solution for data rates up to 12.5 Gbps when the transmitter and the receiver are in the same clock domain.

Jitter Attenuating PLL

Jitter Attenuating PLL modulates the transmit clock inorder to lock to an input source. In the absence of any ports, the Jitter Attenuating PLL goes into hold over mode by locking to an on-board reference clock source.

Key Features (Ask a Question)

Following are the key features of JAPLL:

- ITU-T SyncE compliant PLL.
- Ability to hold outputs when no reference clock is available.
- Ability to detect loss of signal and perform a Hitless clock switching (Phase of the output clock does not change when PLL switches input clocks).
- Maintain frequency accuracy in the range of +/- 4.6 ppm.
- Limited phase wander as measured by TDEV and MTIE.
- Multi-rate protocol support such as:
  - Optical Transport Network (OTN)
  - Passive Optical Network (PON)
  - Common Public Radio Interface (CPRI) (Interface between Radio Equipment Control (REC) and Radio Equipment (RE))
  - Synchronous Ethernet (SyncE)
  - Video applications (High-Definition Serial Digital Interface (HD-SDI), 3 Gigabit Serial Digital Interface (3G-SDI), and so on)

Prerequisites (Ask a Question)

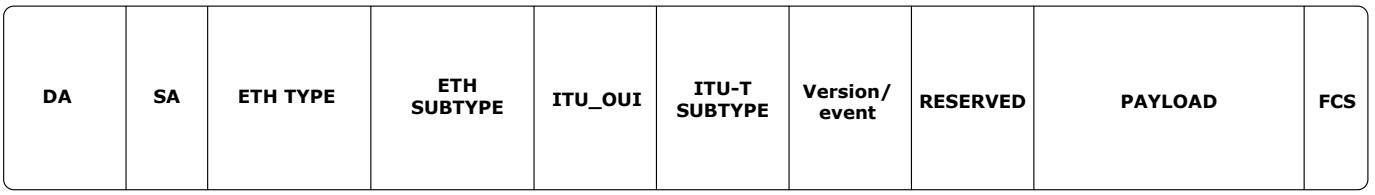
Before you begin, perform the following steps:

- Download the design files from: AN5466: PolarFire 10G SyncE Solution with ESMC and Jitter Attenuating PLL.
- Download and install Libero® SoC (as indicated in the website for this design) on the host PC from the following location: Libero SoC Software Downloads.

Ethernet Synchronization Messaging Channel (ESMC) (Ask a Question)

SyncE uses ESMC Protocol Data Unit's (PDU) to transfer the clock source information which allows for clock source traceability to correctly define the timing source to prevent timing loops. ESMC uses the Organization Specific Slow Protocol (OSSP) for transferring and receiving the QL value. ESMC QL values provide the information about the clock quality which helps the node to derive the timing from the most reliable source. The following figure shows the ESMC PDU format.

Figure 1. ESMC PDU Format



The following table lists the ESMC QL values.

Table 1. ESMC QL Values

Byte Number	Size	Field	Value	
1–6	6 bytes	Destination Address	48'h0180_C200_0002	
7–12	6 bytes	Source Address	48'h0000_0000_0002	
13–14	2 bytes	Ethernet type	16'h8809	
15	1 byte	Ethernet sub-type	8'h0A	
16–18	3 bytes	ITU-OUI	24'h0019A7	
19–20	2 bytes	ITU-T Subtype	16'h0001	
21	1 byte	Version (bits 7:4)	4'h1	
		Event flag (bit 3)	1 indicates event PDU, 0 indicates info PDU	
		Reserved (bits 2:0)	Reserved	
22–24	3 bytes	Reserved	Reserved	
25–1532	4 bytes	TLV_TYPE (1 byte)	8'h01	
		TLV_LENGTH (2 bytes)	16'h0004	
		SSM CODE (1 byte)	PRC	8'h02
			SSU-A	8'h04
			SSU-B	8'h08
			SEC	8'h0B
			DNU	8'h0F
—	32–1486 bytes	Future Enhancement	—	
Last 4	4 bytes	FCS	Core10GMAC computes the FCS and appends to the Frame before transmitting out.	

The following list provides the field description:

- Destination Address (DA): This is the IEEE®-defined slow protocol multicast address.
- Source Address (SA): The source address is the MAC address associated with the port through which the ESMC PDU is transmitted.
- Slow Protocol Ether type: ESMC PDU's must be type encoded and carry the slow protocol type field value.
- Slow Protocol subtype: Assigned by the IEEE and fixed with a value of 0x0A.
- ITU OUI: Organizational unique identifier assigned by the IEEE registration authority.
- ITU subtype: Assigned by ITU-T. The value of 00–01 applies to all usage defined in this recommendation.
- Version: The 4-bit field indicates the version of the ITU-T OSSP frame format. This field contains the value 0x1 to claim compliance with version 1 of this protocol.
- Event flag: This bit distinguishes the critical, time-sensitive behavior of the ESMC event PDU from the ESMC Information PDU. A value of 1 indicates an event PDU and a value of 0 indicates an information PDU.
- Data and Padding: The minimum frame size of 64 bytes and the maximum size for the ESMC PDU is 128 bytes.
- FCS: Four-byte frame check sequence as defined in clause 4 of IEEE 802.3.

The PolarFire FPGA 10G SyncE solution is compliant with the IEEE 802.3ae standard, which supports data transfer rates of up to 10.3125 Gbps. Advantages offered by using PolarFire FPGAs for building 10G Ethernet solutions include the use of low-power transceivers, low-power FPGA fabric, and in-built SyncE-compliant jitter attenuation.

The 10G Ethernet solution is implemented using the Core10GMAC soft IP Media Access Control (MAC) core, which can be configured either in 10GBASE-KR mode or 10GBASE-R mode. This demo design includes the following design, which can be used as reference designs for building a 10GBASE-R.

Ethernet loopback application: A 10GBASE-R Ethernet 64-bit loopback design that can be run on the PolarFire Evaluation Board using Paragon-X Network Tester.

Table of Contents

Introduction....1
Key Features....1
Prerequisites....2
Ethernet Synchronization Messaging Channel (ESMC)....2
1. PolarFire Jitter Attenuator....5
1.1. General Operation....5
2. Demo Design....9
2.1. Design Implementation....9
2.2. MAC and Transceiver Subsystem....10
2.3. ESMC Subsystem....11
2.4. Clock Reset Subsystem....15
2.5. JAPLL Controller Subsystem....17
3. Validation Setup....25
3.1. Running the Demo....26
3.2. Validating the G.8262 Short term Phase Transient Response....29
4. Appendix 1: Generation of Design....31
5. Appendix 2: Programming the Device Using FlashPro Express....33
6. References....35
7. Revision History....36
Microchip FPGA Support....37
Microchip Information....37
Trademarks....37
Legal Notice....38
Microchip Devices Code Protection Feature....38

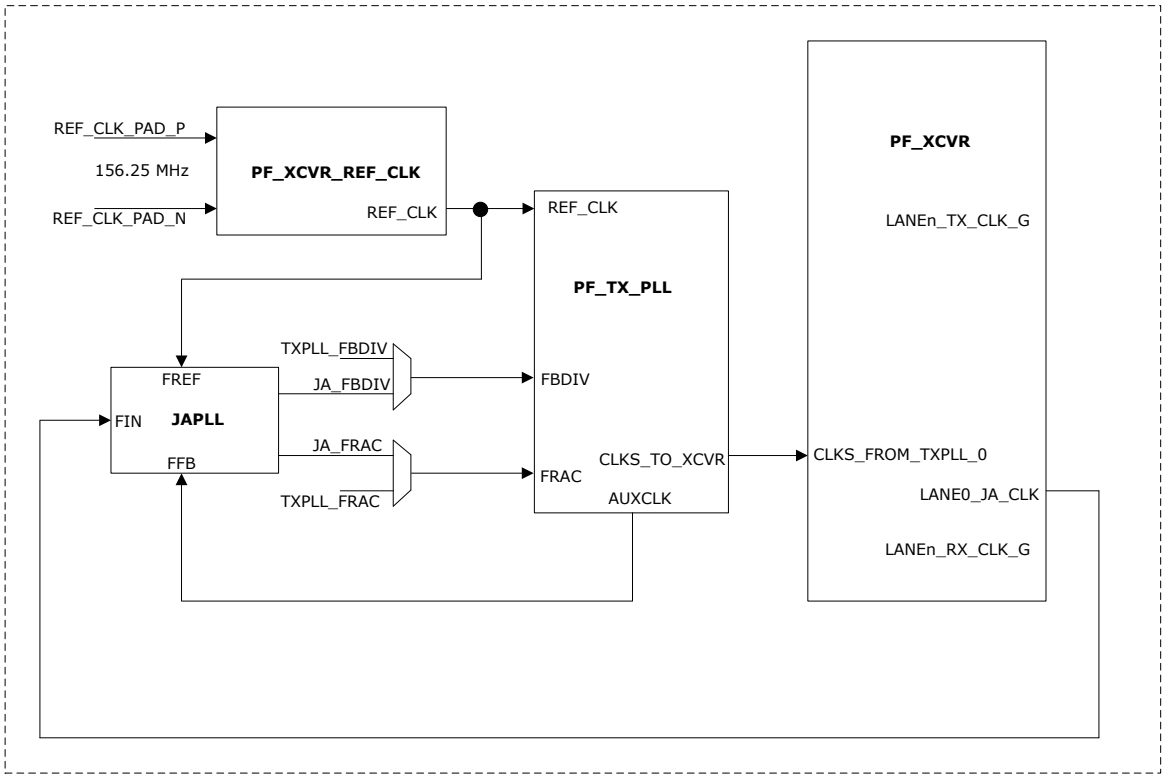
1. PolarFire Jitter Attenuator (Ask a Question)

This section describes the operation of the PolarFire Jitter Attenuator.

1.1 General Operation (Ask a Question)

The following figure shows the general operation of PolarFire Jitter Attenuator.

Figure 1-1. General Operation of PolarFire Jitter Attenuator



PLL's integer and fractional feedback division bits control the JAPLL interfaces with Transmit PLL (PF_TXPLL). The PF_TXPLL automatically configures as a fractional PLL when Jitter cleaning mode is selected in the PF_TXPLL configuration tool in Libero. The output clock from the TXPLL is tied to the Feedback clock (FFB) input of the JAPLL to compare to the JAPLL's noisy reference input clock (FIN). The JAPLL has three phases of acquisition: frequency tracking, coarse phase tracking, and fine phase tracking. Each phase of acquisition utilizes programmable gains to tune the performance of respective stages.

Frequency gain control and frequency tolerance control modify the response time of PLL when in frequency tracking mode. This allows the tracking or filtering of a reference, tunable to the expected noise on the reference. Coarse phase gain control modifies the response while in phase tracking mode, ensuring the PLL can pull-in regardless of how large the frequency error may be when frequency lock is established. Fine phase gain control permits the PLL to have very low bandwidth, effectively averaging out higher frequency noise that may be due to spread spectrum or other noisy input references.

1.1.1 Frequency Tracking Mode (Ask a Question)

JAPLL enters the frequency tracking loop after reset and starts sampling inputs FIN and FFB. Once the frequency of FIN is within 5000 ppm of the frequency of FFB, FLOCK sets to 1 and the JAPLL transits to phase tracking mode and TRANSITION sets to "1". The frequency tracking loop continues to function while in phase tracking mode.

1.1.2 Phase Tracking Mode (Ask a Question)

If the frequency error between FIN and FFB increases beyond 5000 ppm, FLOCK is reset to "0" and the loop exits phase tracking mode and reenters frequency tracking mode. The loop remains in this stage till the frequency lock is achieved back.

Once the PLL is in phase tracking mode, it remains in this mode for DELAYK feedback clock cycles. It spends first DELAYK/2 cycles times in coarse phase tracking mode and the remaining half in fine phase tracking mode. At this point, PHASE_LOCK will be set to "1" and TRANSITION will be set to "0", indicating the PLL is locked to phase and frequency.

When PHASE_LOCK is "1", the PLL will not track high-frequency offset signals and its output frequency remains relatively stable. PLL only exits this state if the frequency lock is lost.

The following figures show the block diagram of Jitter Attenuating PLL and register information.

Figure 1-2. Jitter Attenuating PLL Block Diagram

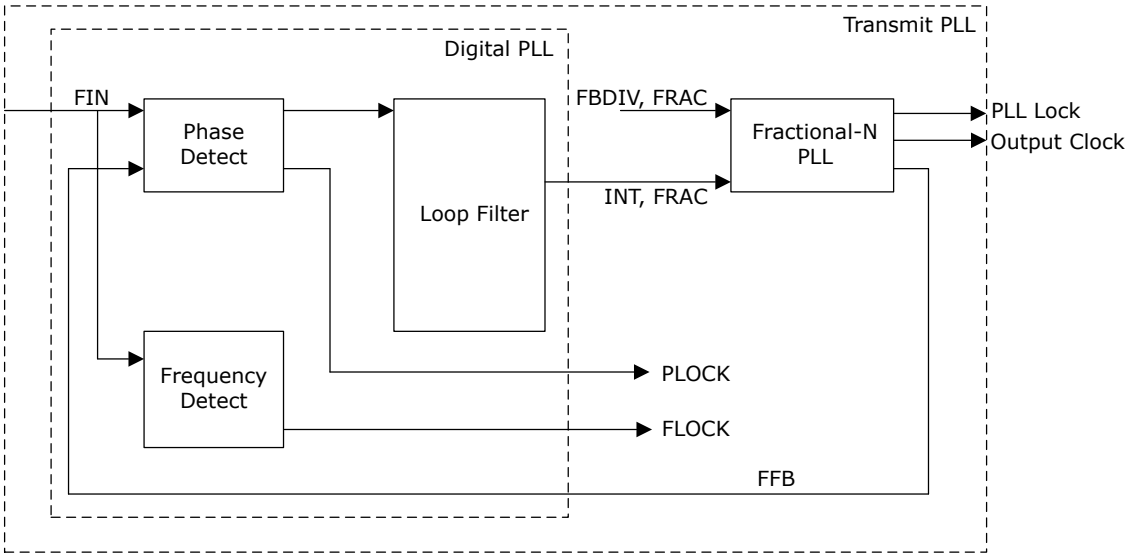
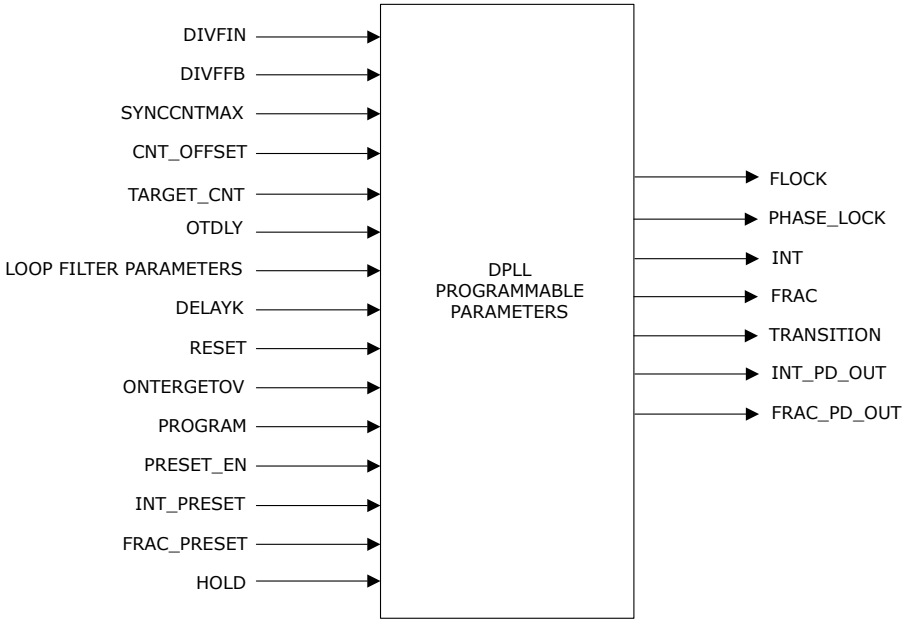


Figure 1-3. Register Information



The following table lists the register details.

Table 1-1. Register Details

SCB Register	DPLL Parameter	Description
TXPLL_JA_1	TXPLL_JA_DIVFIN	Integer divider for FIN
	TXPLL_JA_DIVFFB	Integer divider for FFB
TXPLL_JA_2	TXPLL_JA_SYNCCNTMAX	Maximum count values to determine frequency match
TXPLL_JA_3	TXPLL_JA_CNTOFFSET	OFFSET value used for frequency tracking
	TXPLL_JA_TARGETCNT	If pulse count between FIN and FFB are less than TARGETCNT then the frequency detector declares frequency match
TXPLL_JA_4	TXPLL_JA_OTDLY	Time duration to delay the FFB pulses when transitioning from frequency tracking to phase tracking stage to avoid glitches in FLOCK signal
	TXPLL_JA_FMI	M gain for frequency integral control
	TXPLL_JA_FKI	K gain for frequency integral control
TXPLL_JA_5	TXPLL_JA_PMP1	M gain for Proportional phase comparison loop course setting
	TXPLL_JA_PMP2	M gain for Proportional phase comparison loop fine setting
	TXPLL_JA_PMI1	M gain for Integral phase comparison loop course setting
	TXPLL_JA_PMI2	M gain for Integral phase comparison loop fine setting
TXPLL_JA_6	TXPLL_JA_PKP1	K gain for Proportional phase comparison loop course setting
	TXPLL_JA_PKP2	K gain for Proportional phase comparison loop fine setting
	TXPLL_JA_PKI1	K gain for Integral phase comparison loop course setting
	TXPLL_JA_PKI2	K gain for Integral phase comparison loop fine setting
TXPLL_JA_7	TXPLL_JA_DELAYK	Phase comparison loop initially settles using PKP1 and PKI1 (course settling) parameters, and switch over to PKP2 and PKI2 (fine settling) parameters. DELAYK FFB pulses after phase comparison loop takes over (OTDLY + DELAYK FFB pulses after initial frequency lock).
	TXPLL_JA_FDONLY	1'b1: Frequency + Phase control 1'b0: Frequency control only
	TXPLL_JA_ONTARGETOV	1'b1: Normal Operation 1'b0: Diagnostic mode
	TXPLL_JA_PROGRAM	1'b1: Normal Operation 1'b0: Diagnostic mode
TXPLL_JA_8	TXPLL_JA_FRAC_PRESET	When PRESET_EN: 1'b1, FRAC is equal to TXPLL_JA_FRAC_PRESET
	TXPLL_JA_PRESET_EN	Load preset INT and FRAC values into DPLL
	TXPLL_JA_HOLD	Hold output state of DPLL 1'b1: Hold, PLL will not phase lock in this state 1'b0: Normal operation
TXPLL_JA_9	TXPLL_JA_INT_PRESET	When PRESET_EN: 1'b1, INT is equal to TXPLL_JA_INT_PRESET
	TXPLL_JA_INT_PD_OUT	Integer bits of phase detector output
TXPLL_JA_10	TXPLL_JA_PHASE_LOCK	1'b1: DPLL is in fine tune phase tracking mode 1'b0: DPLL is in coarse tune phase tracking mode
	TXPLL_JA_FLOCK	1'b1: DPLL has achieved frequency lock 1'b0: DPLL has not achieved frequency lock
	TXPLL_JA_FRAC_PD_OUT	Fractional bits of phase detector output

.....continued

SCB Register	DPLL Parameter	Description
TXPLL_JA_RST	TXPLL_JA_RESET	1'b1: DPLL reset asserted 1'b0: DPLL reset deasserted
	TXPLL_JA_RESET_FFB_OVERRIDE	1'b0: Disables DPLL override signal for feedback clock domain reset signal. 1'b0: Enables DPLL override signal for feedback clock domain reset signal.
	TXPLL_JA_RESET_FFB_EXT	1'b0: DPLL reset for feedback clock domain is de-asserted 1'b1: DPLL reset for feedback clock domain is asserted
	TXPLL_JA_RESET_FIN_OVERRIDE	1'b0: Disables DPLL override signal for input clock domain signal 1'b1: Enables DPLL override signal for input clock domain reset signal
	TXPLL_JA_RESET_FIN_EXT	Reset for input clock domain when RESET_FIN_OVERRIDE is 1'b1
	TXPLL_JA_RESET_CLKS_OVERRIDE	1'b0: Disables DPLL override signal for PLL Sync clock domain reset signal 1'b1: Enables DPLL override signal for PLL Sync clock domain reset signal
	TXPLL_JA_RESET_CLKS_EXT	Reset for PLL Sync Clock Domain when RESET_CLKS_OVERRIDE is 1'b1

1.1.3 Hold Output State (Ask a Question)

When JA_HOLD is set to "1", JAPLL maintains its last value of INT and FRAC until HOLD is set to "0". This feature allows the Transmit PLL to generate a stable frequency in the event of a link loss or excessive drift between the transmit and receive clocks.

1.1.4 Locking from INT and FRAC PRESET Settings (Ask a Question)

The JAPLL supports locking from preset INT and FRAC values to speed up lock time. The feature is activated by providing values to INT PRESET and FRAC PRESET and setting PRESET EN = 1'b1. When the loop detects a PRESET EN transition from 1'b0 to 1'b1, it assigns the values of INT PRESET and FRAC PRESET to the frequency detector and reenter frequency tracking mode and then proceed to lock as normal.

PRESET EN must maintain a value of 1'b1 for at least one FFB clock cycle to guarantee that the preset mode takes effect, however, PRESET EN may be left at a value of 1'b1 due to the fact that the preset logic only activates on a positive edge transition of PRESET EN.

In addition, PRESET EN must be set to 1'b1 only after the values of INT PRESET and FRAC PRESET have been written to ensure no glitch occurs. Specifying a new INT PRESET and FRAC PRESET requires that PRESET EN transition to 1'b0 for at least one FFB clock cycle and then to transition to 1'b1 again.

1.1.5 Exiting Hold State using PRESET Values (Ask a Question)

PRESET EN is set to 1'b1 while the PLL is in a hold state, and once HOLD = 1'b0, the PLL will resume locking from the INT PRESET and FRAC PRESET values.

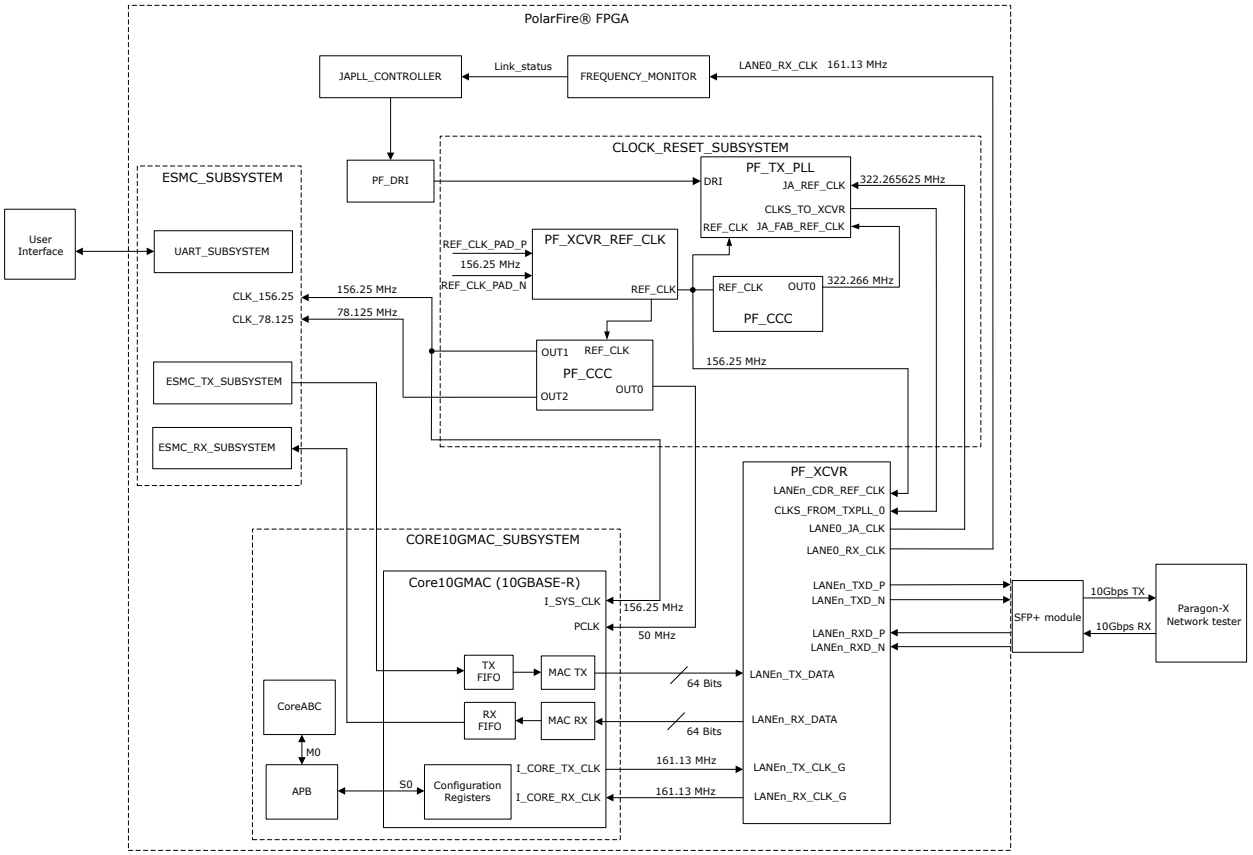
2. Demo Design (Ask a Question)

The PolarFire 10G SyncE solution receives the Ethernet traffic generated by the Paragon tester and checks for the destination address field in the Ethernet packet header.

When the destination address matches the IEEE 802.3 specified OSSP slow protocol destination address field, the packet is processed, and appropriate response is generated in the form of an ESMC PDU.

The following shows the top-level block diagram of the PolarFire 10G Synchronous Ethernet solution featuring ESMC.

Figure 2-1. Top-Level Block Diagram of the PolarFire 10G Synchronous Ethernet Solution Featuring ESMC



2.1 Design Implementation (Ask a Question)

The 10G SyncE solution includes the following components:

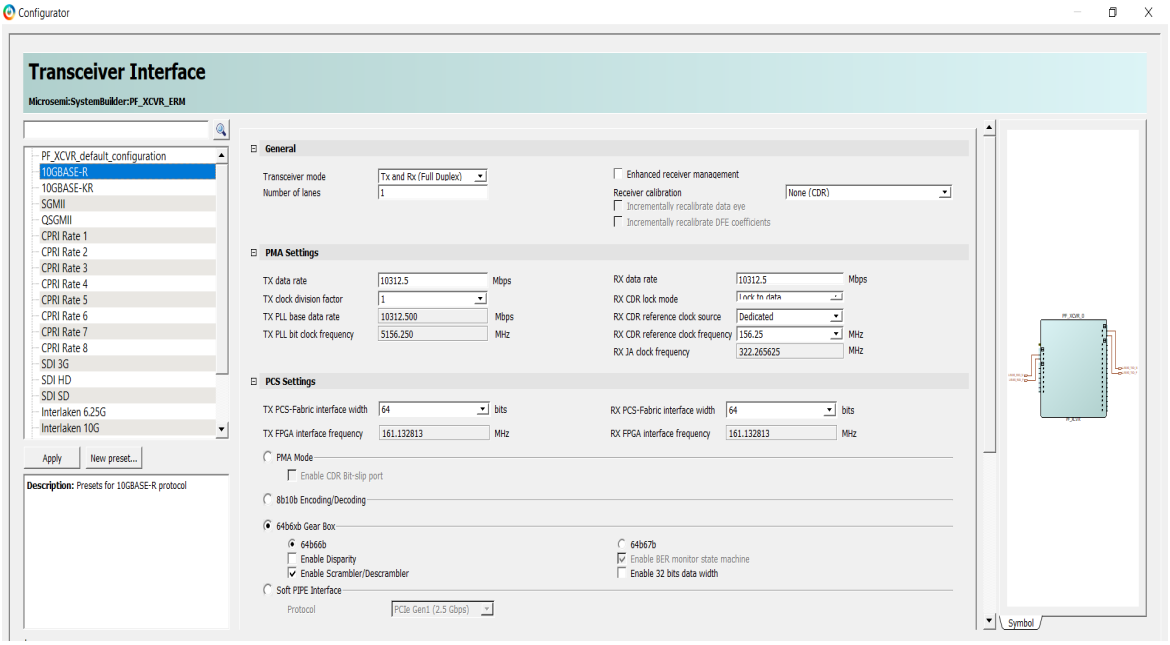
- MAC and Transceiver Subsystem
- ESMC Subsystem
- UART Subsystem
- Clock Reset Subsystem
- JAPLL Controller Subsystem

2.1.1 PF_XCVR (Ask a Question)

The PolarFire high-speed transceiver (PF_XCVR) is a hard IP block and supports data rates from 250 Mbps to 12.5 Gbps. In this demo, PF_XCVR is configured for the data rate of 10312.5 Mbps. It is configured with a CDR reference clock of 156.25 MHz with Lock to data selected as the CDR Lock

mode. The PCS of the transceiver is interfaced with Core10GMAC. It is configured for the 64/66b mode with scrambler/descrambler enabled. The self-synchronizing scrambler, generates sufficient transitions to aid data and clock recovery at the CDR. The following figure shows the transceiver interface configuration.

Figure 2-2. Transceiver Interface



2.2 MAC and Transceiver Subsystem (Ask a Question)

MAC and Transceiver subsystem consists of the following components.

2.2.1 Core10GMAC (Ask a Question)

Core10GMAC is configured for 10GBASE-R mode with a core data width of 32 bits. Core data width is the width of the data path connected to the transceiver interface. The system data width is the width of the interface to the user logic, and is configured as 64 bits. (In this demo, the FiFo_wrapper_top module provides this interface).

The Tx and Rx Pause features are disabled, and both the MAC TX FIFO depth and MAC RX FIFO depth are set to 256.

The Core10GMAC IP is configured using the CoreABC soft processor. The following table lists Core10GMAC configuration for the demo design.

Table 2-1. Core10GMAC Configuration

Register	Address	Offset	Bit	Binary Value
MAC Tx Config Register	(0xA)	0x3	cfg_sys_mac_tx_en	1
		0x4	sys_mac_tx_fcs_ins	1
MAC Rx Config Register	(0xB)	0x0	mac_rx_fcs_remove	1
		0x3	cfg_sys_mac_rx_en	1

For more information about the features and registers of Core10GMAC, see Core10GMAC User Guide.

2.2.2 CoreABC (Ask a Question)

CoreABC is a configurable, low-gate count controller intended for Advanced Microcontroller Bus Architecture Advanced Peripheral Bus (AMBA APB) based designs. The CoreABC processor is used in this design because this demo design requires only a few registers to configure and no dynamic changes are required in the configuration. Depending on the application requirements, RISC-V, Cortex- M1, or any other soft processor is used for configuring the registers.

2.2.3 CoreAPB3 (Ask a Question)

CoreAPB3 is a bus component that provides an AMBA AHB fabric for interconnection between an APB initiator and up to 16 APB targets. CoreAPB3 supports a single APB3 initiator. In this design, CoreAPB3 is used to connect the CoreABC APB initiator interface to the Core10GMAC APB target interface.

2.3 ESMC Subsystem (Ask a Question)

ESMC subsystem processes data received from the Core10GMAC and uses the information to control the JAPLL to enable/disable SyncE.

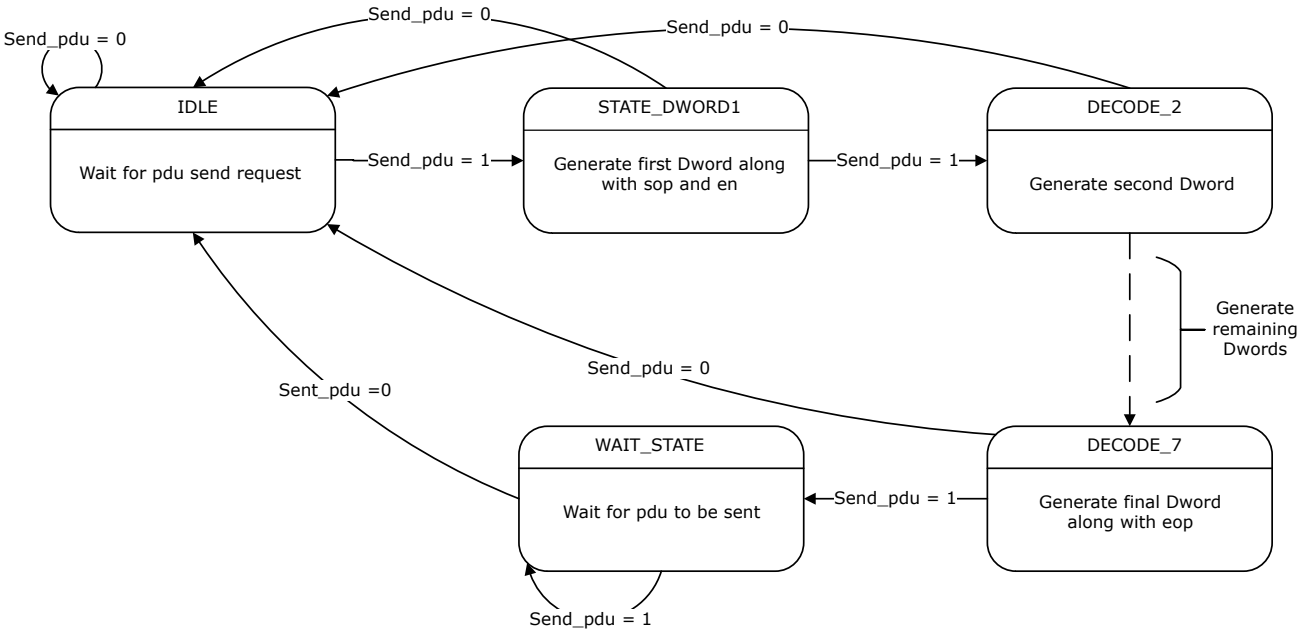
2.3.1 ESMC TX Subsystem (Ask a Question)

ESMC TX subsystem consists of the following modules:

- pdu_generator: This module handles the following features:
  - pdu_generate_request_servicing: This module services the request received from the pdu_checker and also generates an acknowledgment signal to indicate the successful transmission of PDU.
  - pdu_packet_framer: This module frames 58-byte packet based on a different type of PDU request received. Currently, the framer accepts a request to generate 10 different PDU's.
  - pdu_counter_logic: Counter logic to keep a log of how many PDU's have been sent out. Current logic keeps track of all the different types of PDU's sent.
  - pdu_sent_max_threshold: Logic to limit the number of PDU's that can be sent using a 1 second timer.
- fifo_mgmt: FIFO management module is used to efficiently manage the clock domain crossing between a PDU generator (slow clock domain) and Core10GMAC (fast clock domain). The module also consists of a controller that manages the writes and reads for a group of 10 FIFO's.
  - FIFO controller reads from FIFO when i_sys_mac_tx_fifo_af signal is "0".
  - The writes to the FIFO's are performed by pdu_generator when a "PDU send" request is generated from the Rx subsystem.

The following figure shows the PDU generator finite state machine.

Figure 2-3. PDU Generator Finite State Machine



PDU generator finite state machine traverses through the following states when any one of the send PDU requests are received:

- IDLE: Check for any of the PDU send requests for "1" and then traverse to STATE_DWORD1.
- STATE_DWORD1 to STATE_DWORD8: The framer accesses the predefined data word from the defined file based on the request sent from pdu_checker and also traverses to the next state.
- WAIT_STATE: Logic checks for any of the PDU send requests to be "1" and then traverse to the IDLE state.

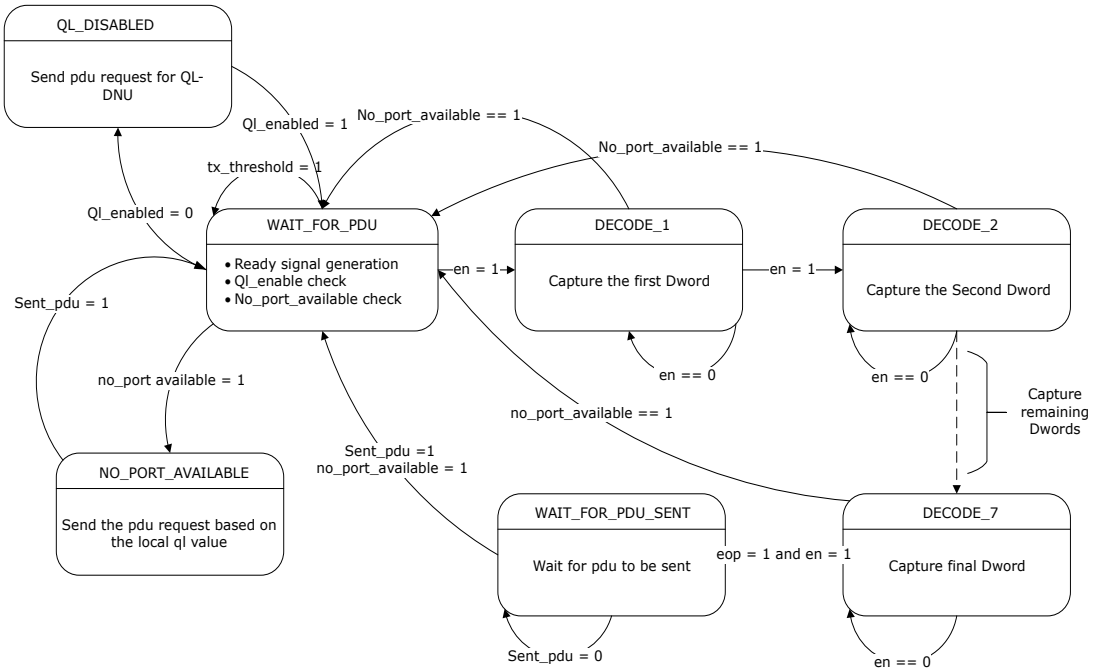
2.3.2 ESMC RX Subsystem (Ask a Question)

ESMC RX subsystem consists of the following modules:

- pdu_parser: It implements a FIFO management system to detect and store the ESMC PDU by checking the Destination Address (48'h0180_C200_0002).
  - Header decoder
  - Storing pdu
- pdu_checker: This module handles the following features:
  - pdu_info_extraction: A finite state machine decodes and stores the data fields from the received PDU.
  - ql_priority_selection_algo: Logic to compares the local QL value with the decoded QL value and selects the QL value which has higher priority (better clock quality).
  - pdu_generation_request: When ql_enabled is "0", a DNU info pdu send request is sent to the pdu_generator, whereas when ql_enabled is "1", pdu_checker module sends a pdu generation request to pdu_generator based on the result of QL selection logic.
  - Port availability information

The following figure shows the PDU checker finite state machine.

Figure 2-4. PDU Checker Finite State Machine



PDU checker finite state machine traverses through the following states when any of the send PDU requests are received:

- The Enabling Jitter Cleaner mode has the following scenarios:
  - After reset is de-asserted enable_japll signal is "1", when pdu_data_ready signal is "1", pdu_checker will deassert the JAPLL by setting enable_japll to "0".
  - This sequence is performed only once when ql_enable is "1" to show the actual working of the jitter cleaner PLL.
  - Enable_japll will remain "0", until there is any change in the input QL value.
  - If the QL selection logic detects a change in the input QL value, enable_japll becomes "1" for the transmitter to lock to the new clock reference which has a better clock quality than the local reference clock.

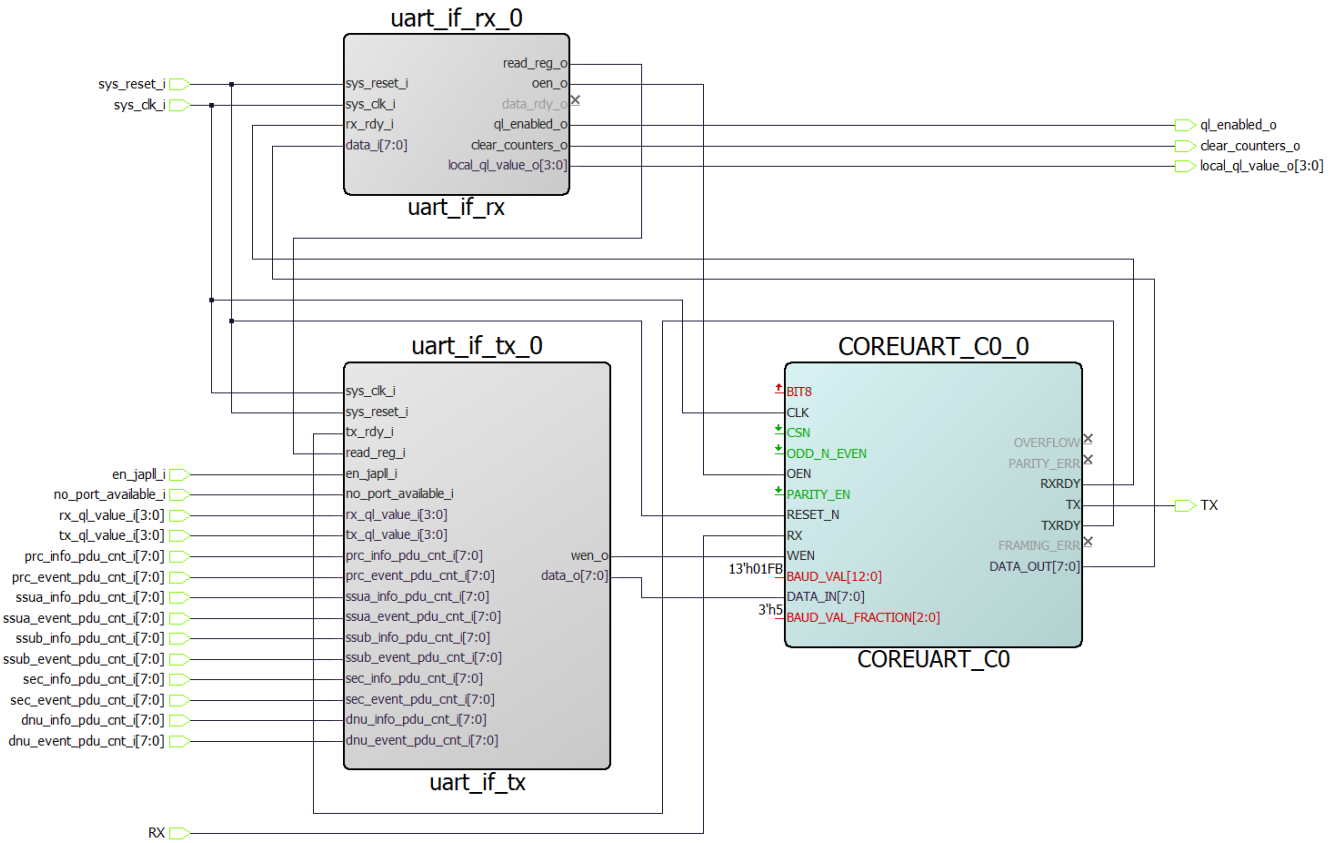
2.3.3 UART Subsystem (Ask a Question)

The Universal Asynchronous Receiver/Transmitter (UART) subsystem implements the fabric UART logic to interface with GUI, which is used as a user interface to control and display status of the running design. The UART subsystem has three
sub-modules:

- UART RX Interface
- UART TX Interface
- CORE UART

The following figure shows the UART subsystem.

Figure 2-5. UART Subsystem



2.3.3.1 UART RX Interface (Ask a Question)

The UART RX interface finite state machine receives the write request and write data from CORE_UART and provides the following data to the esmc_tx and esmc_rx subsystems:

- QL enable/disable: Determines whether ESMC IP operates in synchronous mode or non-synchronous mode.
  - When ql_enabled is "1", design is in synchronous mode and processes the ESMC PDU's.
  - When ql_enabled is "0", design is in non-synchronous mode and does not process ESMC PDU's.
- Local QL value: Local QL value of the EEC equipment is used after reset.
- Clear counter: Clear counter signal clears all the PDU counters.

2.3.3.2 UART TX Interface (Ask a Question)

The UART TX interface finite state machine receives the following design status and provides the status to CORE_UART upon receiving read request:

- QL enable/disable
- Current local QL value
- PRC info pdu count
- PRC event pdu count
- SSUA info pdu count
- SSUA event pdu count
- SSUB info pdu count

- SSUB event pdu count
- SEC info pdu count
- SEC event pdu count
- DNU info pdu count
- DNU event pdu count

2.3.3.3 Core UART (Ask a Question)

CORE UART configures at baud value 91600 bps and a 78.125 MHz clock is provided as reference.

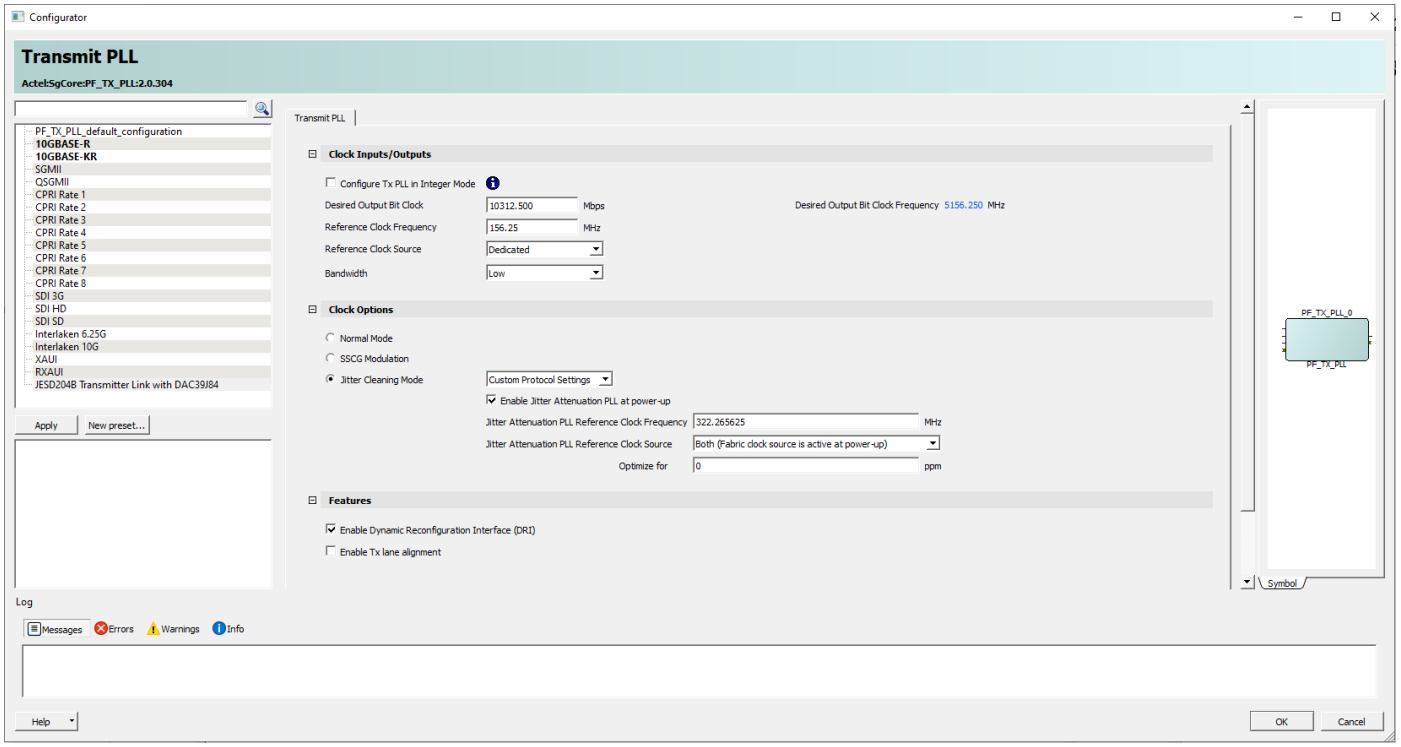
2.4 Clock Reset Subsystem (Ask a Question)

Clock Reset Subsystem generates the necessary clocks and resets for PF_XCVR, Core10GMAC, and ESMC_TOP.

2.4.1 Transmit PLL (Ask a Question)

The PolarFire Transmit PLL (PF_TX_PLL) is a hard IP block that provides a bit clock and a reference clock to the transceiver block. The transmit PLL is configured with a reference clock of 156.25 MHz and generates an output clock of 10312.5 Mbps. The following figure shows the Transmit PLL configurator.

Figure 2-6. Transmit PLL



2.4.2 Transceiver Reference Clock (Ask a Question)

The Transceiver Reference Clock (PF_XCVR_REF_CLK) is a hard IP block that provides a reference clock (REF_CLK) of 156.25 MHz to the transmit PLL and a fabric reference clock (FAB_REF_CLK) which is provided as input to the Clock Conditioning Circuit (CCC) to generate the pclk (for configuration) and I_SYS_CLK of the Core10GMAC.

2.4.3 PF_POWER_INIT (Ask a Question)

The PF_POWER_INIT block ensures the device is turned on systematically. The process of turning on the device includes following three steps:

1. Power-on reset
2. Programmed device boot
3. Design initialization

During design initialization, the transceiver configuration is initialized using the data stored in the non-volatile memory. The output of the PF_POWER_INIT block is ANDed with the resets used in the design to reset entire logic. The following table lists the reset signals.

Table 2-2. Reset Signals

Reset Signal Name	Driven System
common_reset_o	• ESMC_SUBSYSTEM • JAPLL_CONTROLLER_SUBSYSTEM
core10gmac_tx_reset_o	XCVR_CORE10GMAC_SUBSYSTEM
core10gmac_rx_reset_o	XCVR_CORE10GMAC_SUBSYSTEM
coreabc_reset_o	CoreABC within XCVR_CORE10GMAC_SUBSYSTEM

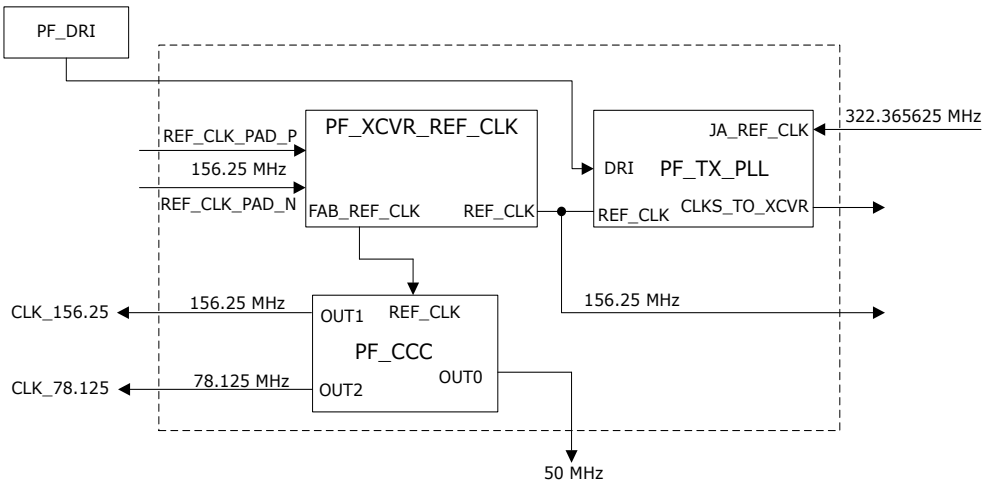
2.4.4 PF_CCC (Ask a Question)

The PolarFire CCC (PF_CCC) block takes an input clock of 156.25 MHz from the FAB_REF_CLK signal (output of PF_XCVR_REF_CLK) and generates a 50 MHz clock at OUT0, 156.25 MHz at OUT1 and 78.125 MHz at OUT2. The OUT0 port of the CCC is used for the configuration and, OUT1 and OUT2 port of the CCC are used for the user logic in the design.

2.4.5 Clocking Structure (Ask a Question)

There are three clock domains in the reference designs; RX_CLK (161.13 MHz), TX_CLK (161.13 MHz), and CLK_156pt25 (156.25 MHz). The following figure shows the clocking structure.

Figure 2-7. Clocking Structure



The following table lists the clock descriptions.

Table 2-3. Clocks

Clock Name	Frequency	Instance Name
clk_156pt25 MHz	156.25	esmc_tx_subsystem_0
		esmc_rx_subsystem_0

.....continued		
Clock Name	**Frequency**	**Instance Name**
clk_78pt125 MHz	78.125	esmc_tx_subsystem_0
		esmc_rx_subsystem_0
		UART_IF_0
TX_CLK_R	161.13	CORE10GMAC0_0
RX_CLK_R	161.13	CORE10GMAC0_0
		Frequency_change_monitor_0
clk_156pt25 MHz	156.25	CORE10GMAC0_0
clk_50 MHz	50	CORE10GMAC_SUBSYSTEM
		Japll_controller_0
		PF_DRI_0
		COREABC0_0

2.5 JAPLL Controller Subsystem (Ask a Question)

This section describes the JAPLL controller subsystem.

2.5.1 Frequency Change Monitor (Ask a Question)

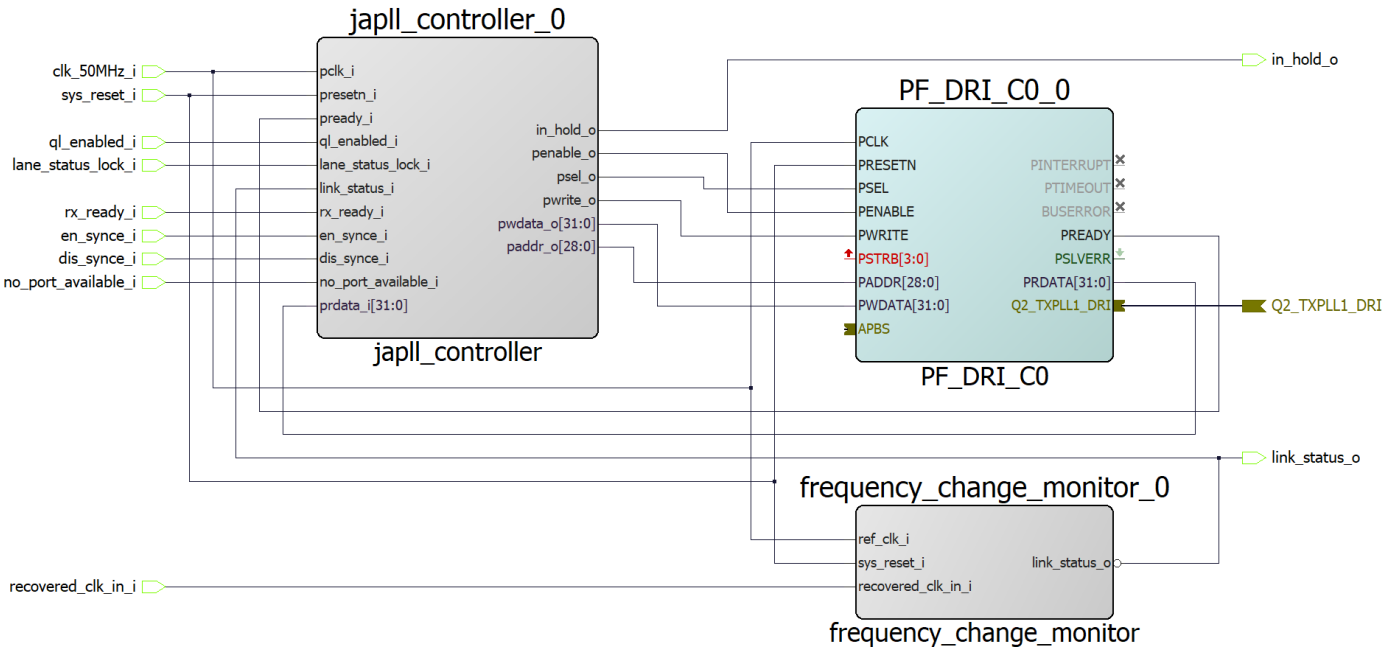
The frequency change monitor provides the following scenarios:

- The frequency change monitor takes the recovered clock as input and monitors the frequency change with respect to a stable clock.
- The logic checks the frequency of the clock at fixed intervals and determines the stability of the clock. If the deviation in the frequency exceeds the predetermined limits, a link loss signal is asserted, and the japll_controller is informed of a break link condition.

2.5.2 JAPLL Controller (Ask a Question)

The following figure shows the JAPLL controller subsystem interface.

Figure 2-8. JAPLL Controller



japll_controller implements a DRI initiator interface to interact with the JAPLL through a PF_DRI core.

Fabric logic and the PF_DRI core operates on the same frequency of 50 MHz. Enabling and disabling of syncE is done based on the requests received from the ESMC IP. After power-up, JAPLL will be locked to the external auxiliary clock. When the en_synce command is received from the ESMC IP, JAPLL controller executes a sequence of steps for the JAPLL to switch its input to the cdr recovered clock. JAPLL generates the INT and FRAC values which are derived based on the cdr recovered clock. TX clock synchronizes with the recovered clock to show that synchronous Ethernet is enabled. For more information, see Hitless Clock Switching: Enable SyncE.

When the dis_synce command is received from ESMC IP, JAPLL input is switched back to the local auxiliary clock. JAPLL generates the INT and FRAC values based on the local auxiliary clock. The TX clock synchronizes with the local auxiliary clock to prove that synchronous Ethernet is disabled. For more information, see Hitless Clock Switching: Disable SyncE.

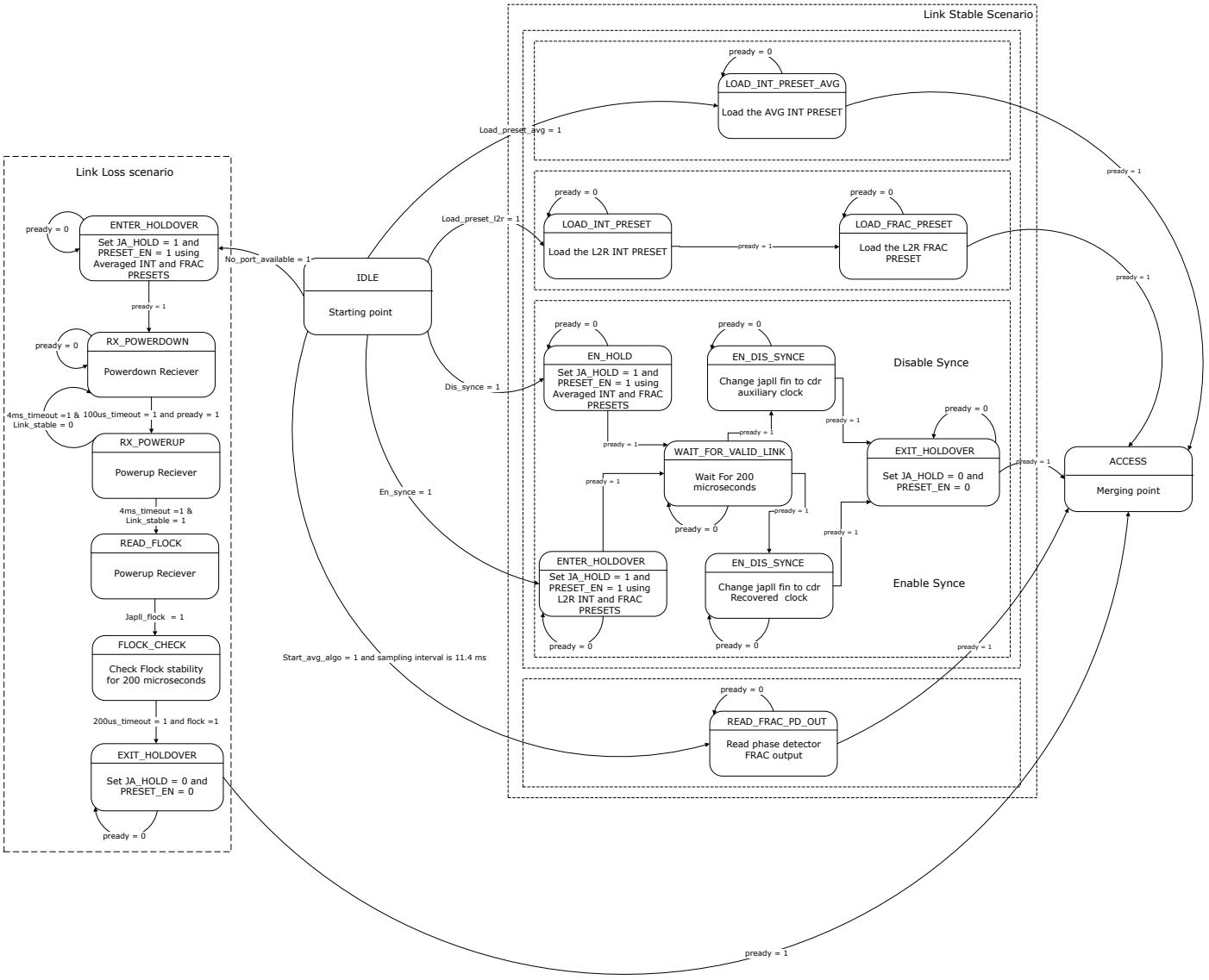
Japll_apb3_master performs the following register write operations.

Table 2-4. Register Write Operation

Function	Register	Description
Write	EXT_PLL_JA_8	Enabling/disabling JA_HOLD and PRESET_EN
Write	EXT_PLL_JA_8	Load FRAC PRESET value
Write	EXTPLL_JA_9	Load INT_PRESET value
Write	DES_RSTPD	For RX power-down and power-up
Read	EXTPLL_JA_10	Read JAPLL Flock and Plock values
Read	EXTPLL_JA_9	Read INT_PD_OUT value
Read	EXTPLL_JA_10	Read FRAC_PD_OUT value

JAPLL Controller handles two different scenarios in this reference design. The following figure shows the reference design.

Figure 2-9. Reference Design



JAPLL Controller handles the following functions:

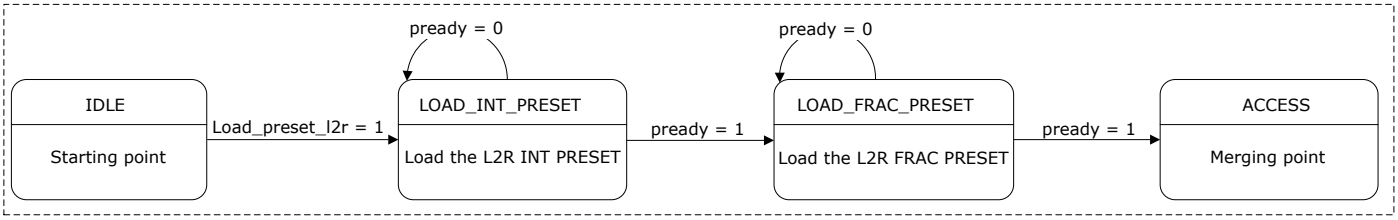
- Load the JAPLL with the default INT and FRAC preset values
- Hitless Clock Switching: Enable SyncE and Disable SyncE
- Read the JAPLL phase detector output every 11.4 ms
- Implementing the averaging algorithm
- Load the JAPLL with the computed averaged INT preset value
- Handling the break-link and make-link conditions

These scenarios are described in the following sections.

2.5.2.1 Load the JAPLL with the Default INT and FRAC Preset Values (Ask a Question)

The following figure shows the state machine for JAPPL with default INT and FRAC Preset values.

Figure 2-10. Load the JAPLL with the Default INT and FRAC Preset Values

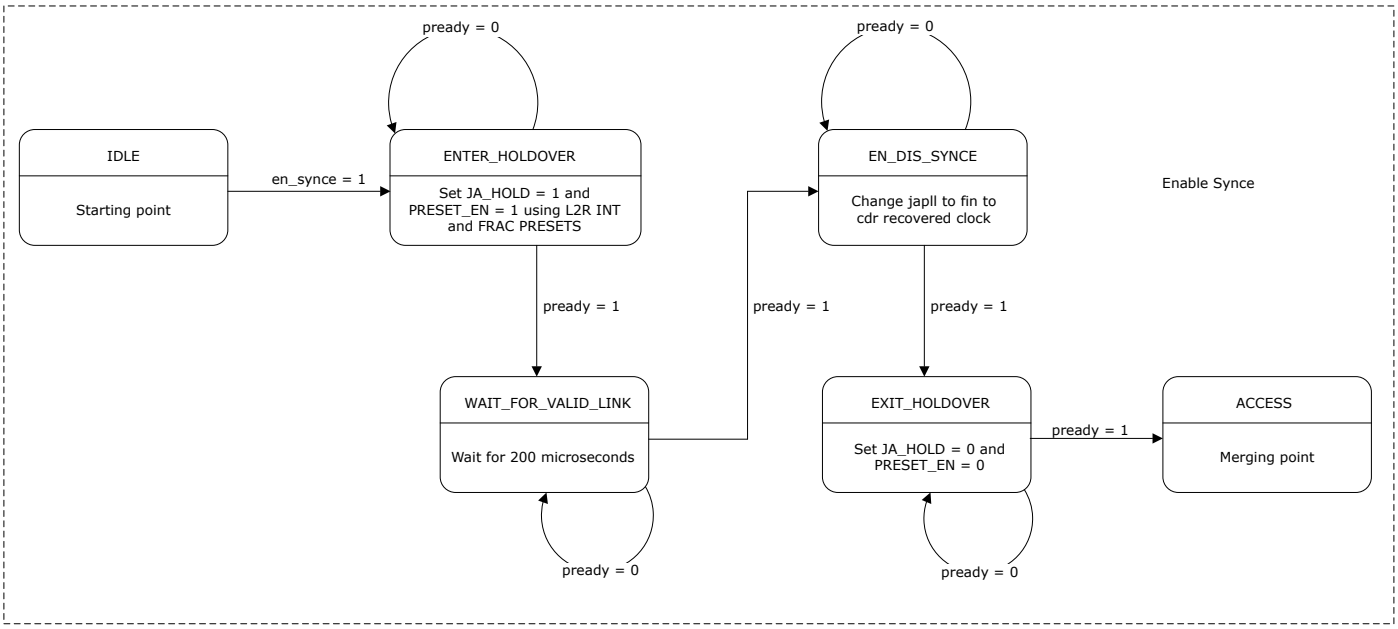


The first task carried out by the JAPLL Controller finite state machine after reset is to load the JAPLL INT and PRESET values with the default's integer and fractional PLL divider settings with respect to the local auxiliary clock frequency. Loading the JAPLL with new INT and FRAC preset values speeds up the JAPLL locking to the reference clock.

2.5.2.2 Hitless Clock Switching: Enable SyncE (Ask a Question)

The following figure shows the Hitless clock switching for Enable SyncE.

Figure 2-11. Hitless Clock Switching: Enable SyncE



Whenever the ESMC receiver subsystem receives a new QL-value, the selection algorithm compares the incoming QL-value with the already existing local QL-value and makes the appropriate decision.

If the incoming QL-value has higher priority than the existing local QL-value, then the selection algorithm notifies the JAPLL Controller to take necessary action such that the JAPLL locks to the better quality recovered clock and enable the SyncE.

The JAPLL Controller performs the following sequence of steps to enable the SyncE.

1. ENTER_HOLDOVER:
   - Enable JAPLL hold by setting JA_HOLD to "1" and PRESET_EN to "1" to use the default INT and FRAC preset values.
2. WAIT_FOR_VALID_LINK:
   - Wait for 200 µs to make sure the JAPLL enters into holdover.
3. EN_DIS_SYNCE:
   - Switch the FIN of JAPLL to the cdr recovered clock.

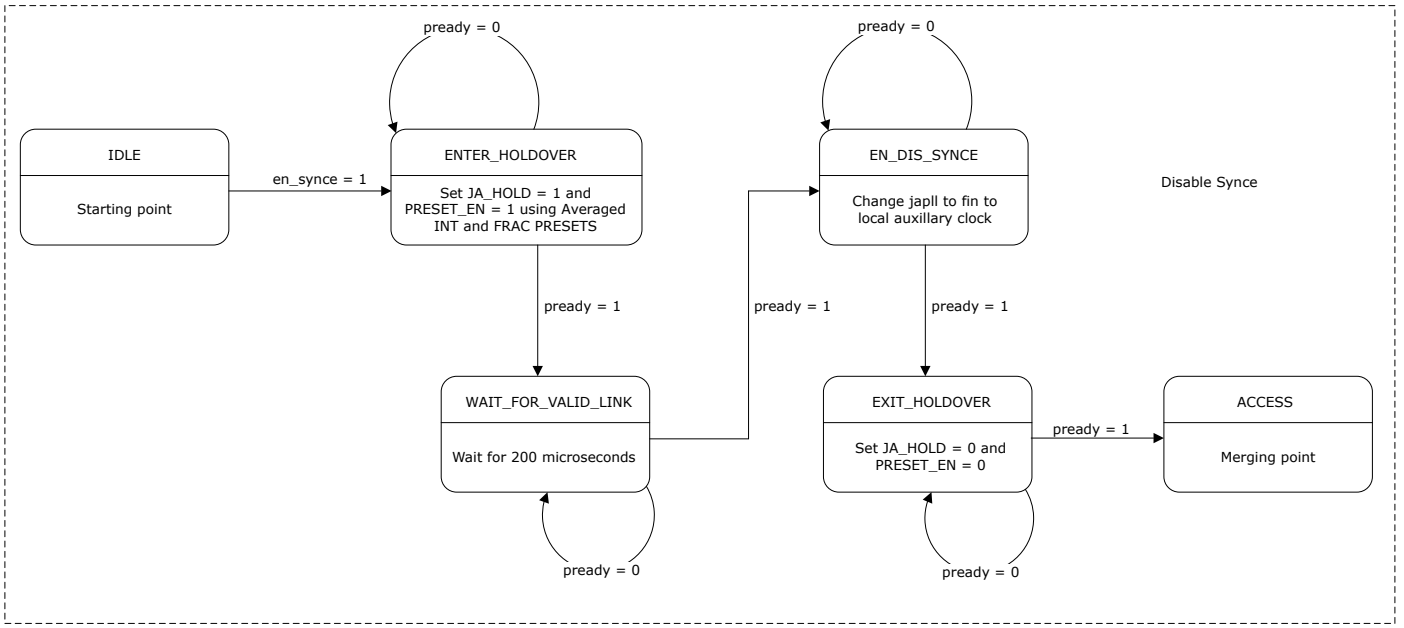
- Now JAPLL starts locking to the recovered clock.

4. EXIT_HOLDOVER:
   - Set JA_HOLD to "0" and PRESET_EN to "0" to disable JAPLL hold.
5. Start the averaging algorithm: For more information, see Averaging Algorithm Implementation.

2.5.2.3 Hitless Clock Switching: Disable SyncE (Ask a Question)

The following figure shows the Hitless clock switching for disable SyncE.

Figure 2-12. Hitless Clock Switching: Disable SyncE



Whenever the ESMC receiver subsystem receives a new QL-value, the selection algorithm compares the incoming QL-value with the already existing local QL-value and makes the appropriate decision.

If the incoming QL-value has a lower priority than the existing local QL-value, then the selection algorithm notifies the JAPLL Controller to take necessary action such that the JAPLL locks to the better-quality local auxiliary clock and disable the SyncE.

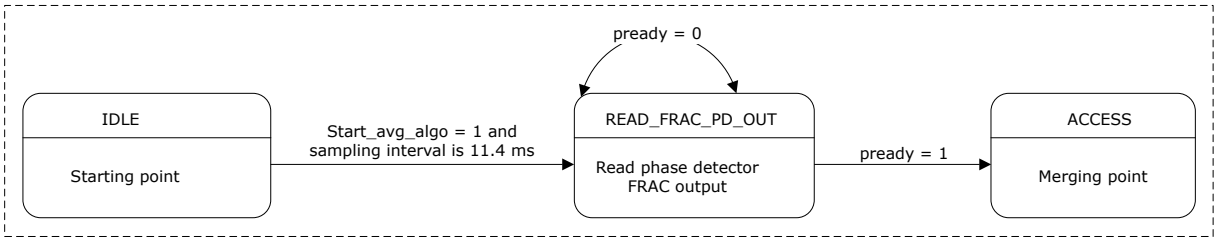
The JAPLL Controller performs the following sequence of steps to disable the SyncE:

- ENTER_HOLDOVER:
  - Enable JAPLL hold by setting JA_HOLD to "1" and PRESET_EN to "1" to use the Averaged INT and FRAC preset values.
- WAIT_FOR_VALID_LINK:
  - Wait for 200 µs to make sure the JAPLL enters holdover.
- EN_DIS_SYNCE:
  - Switch the FIN of JAPLL to the local auxiliary clock.
  - Now JAPLL starts locking to the local auxiliary clock.
- EXIT_HOLDOVER:
  - Set JA_HOLD to "0" and PRESET_EN to "0" to disable JAPLL hold.
- Stop the averaging algorithm. For more information, see Averaging Algorithm Implementation.

2.5.2.4 Monitoring JAPLL Phase Detector Output (Ask a Question)

The following figure shows the state machine which reads the JAPLL phase detector output.

Figure 2-13. JAPLL Phase Detector Output Read



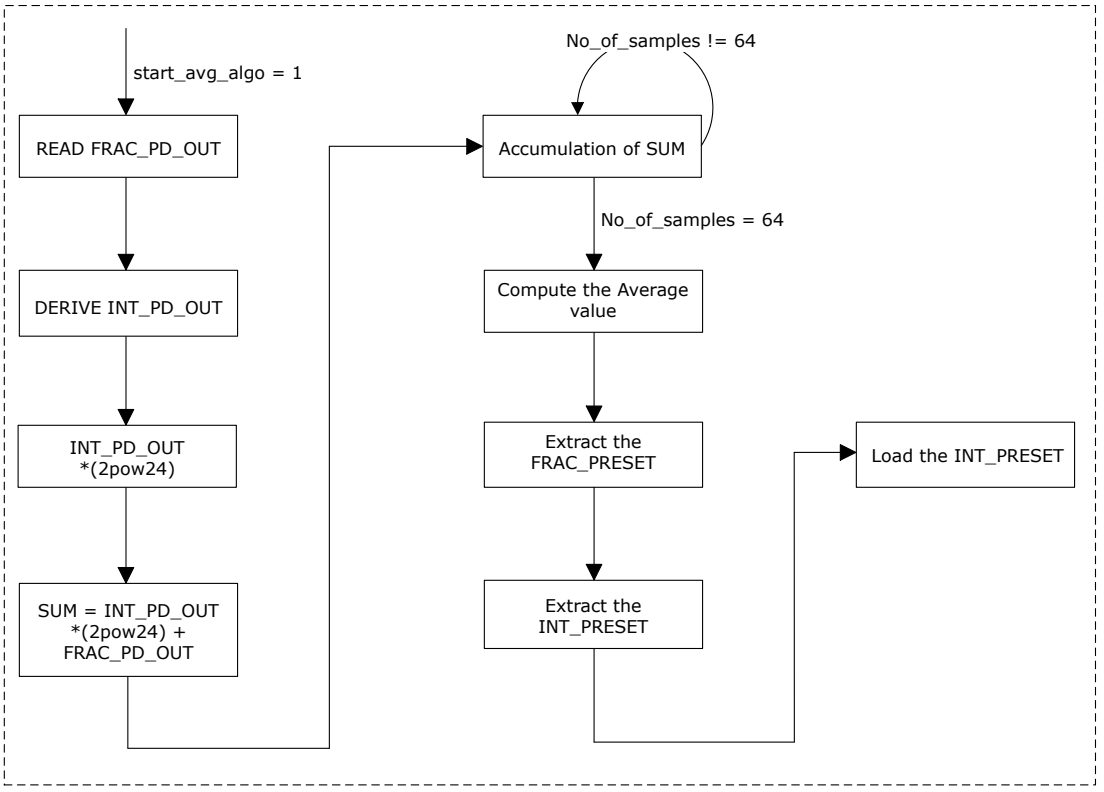
For the JAPLL to be G.8262 compliant, the loop bandwidth has to be the minimum so that the averaging frequency can be within the Wander cut-off frequency limits, which is 1–10 Hz.

The sampling rate is set to 11.4 ms, and the total number of samples to 64, the averaging frequency is 1.36 Hz which falls in the range of 1–10 Hz.

2.5.2.5 Averaging Algorithm Implementation (Ask a Question)

The following figure shows the sampling algorithm block diagram.

Figure 2-14. Averaging Algorithm



When the start_avg_algo_i signal is asserted, the japll_controller finite state machine executes the averaging algorithm by continuously reading the INT_PD_OUT and FRAC_PD_OUT values of the Phase detector and computes the average values till it reaches the user-specified limit.

The following procedure computes the average value:

1. Read INT_PD_OUT and FRAC_PD_OUT.
2. Accumulate the INT_PD_OUT and FRAC_PD_OUT.
3. Repeat the preceding steps till the count reaches AVG_COUNT specified by the user.

4. After AVG_COUNT iterations, compute the average value.
5. Load the JAPLL INT preset using the new computed INT_PRESET.
6. Extract the INT_PRESET and FRAC_PRESET from the computed average value.

In the event of a clock switching or link loss, JAPLL is programmed to use the computed INT_PRESET and FRAC_PRESET when entering in to hold state using the PRESERT_EN function. During Hold state, the phase detector is shut down, and the outputs of JAPLL are held to their last values.

Averaging algorithm forces the JAPLL outputs to be loaded with the computed INT_PRESET and FRAC_PRESET values in order to make sure the Transmit PLL receives the INT and FRAC values which are closer to the actual and do not exceed the ±7.5 ppm requirement of syncE.

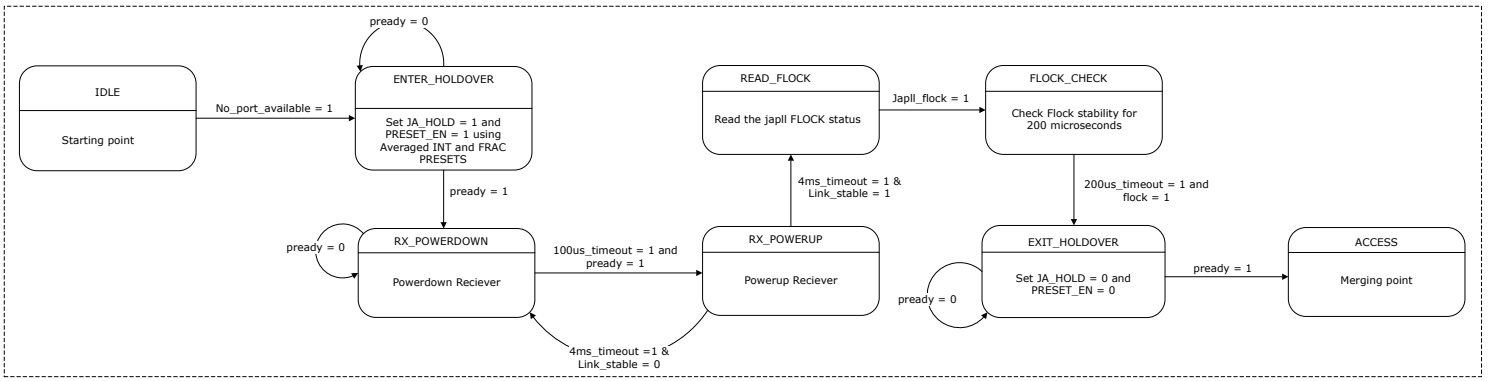
Averaging Algorithm starts when start_avg_algo signal becomes "1" and stops for the following cases:

- When start_avg_algo signal becomes "0" (occurs during the disable syncE process)
- When there is a break-link situation

2.5.2.6 Break-link and Make-link Conditions Handling (Ask a Question)

The following figure shows the break-link and make-link conditions.

Figure 2-15. Break-link and Make-link Conditions



The following are the steps to handle the break-link and make-link condition:

1. When the frequency monitor logic detects the deviation in the recovered clock frequency, it asserts the link status signal indicating the break-link condition.
2. ESMC RX SUBSYSTEM generates a no_port_available signal to the JAPLL Controller.
3. ENTER_HOLDOVER: When no_port_available = 1, Set JA_HOLD to "1" and PRESET_EN to "1" to enable JAPLL hold, to use the previously loaded preset values and proceed to RX_POWERDOWN state.
4. RX_POWERDOWN: To check the make link condition, FSM periodically checks the cable reinsert sequence by powering up and shutting down the receiver multiple times. Wait for 100 µs in RX_POWERDOWN state and proceed to RX_POWERUP state.
5. RX_POWERUP: Wait for 4 ms to check the stability of the following signals:
   - rx_ready from transceiver
   - lane_status_lock from transceiver
   - link status from frequency change monitor

   If all the preceding signals are stable at the end of 4 ms time, then FSM proceeds to READ_FLOCK state else return to the RX_POWERDOWN state.

6. READ_FLOCK: After the make-link condition is successfully verified, the JAPLL FLOCK signal is verified for its stability.
7. FLOCK_CHECK: Check the FLOCK stability for 200 ms and if the signal is stable, then proceed to EXIT_HOLDOVER state else return to the READ_FLOCK state.
8. EXIT_HOLDOVER: Set JA_HOLD to "0" and PRESET_EN to "0" to disable the JAPLL hold.

The following table lists the ports and their description.

Table 2-5. Port Description

Port Name	Direction	Description
LANE0_RXD_N	IN	ESMC Receiver inverted input
LANE0_RXD_P	IN	ESMC Receiver non-inverted input
REF_CLK_PAD_N	IN	Inverted reference clock obtained from on-board 156.25 MHz oscillator
REF_CLK_PAD_P	IN	Non-Inverted reference clock obtained from on-board 156.25 MHz oscillator
UART_RX_IF_I	IN	UART receiver interface
reset_i	IN	Asynchronous reset
LANE0_TXD_N	OUT	ESMC Transmitter inverted input
LANE0_TXD_P	OUT	ESMC Transmitter non-inverted input
UART_TX_IF_I	OUT	UART Transmitter interface

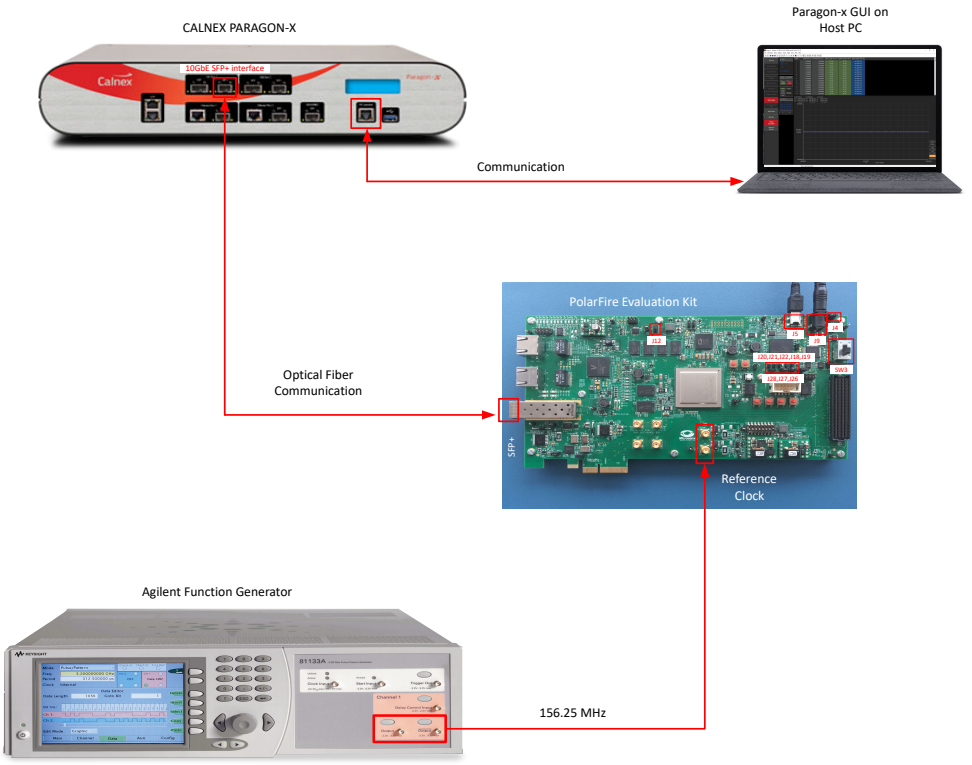
3. Validation Setup (Ask a Question)

This chapter describes how to install and use the GUI to run the ESMC demo.

The following are the prerequisites:

- Make sure that the PolarFire Evaluation board is connected.
- Ensure that the PolarFire FPGA is programmed with the ESMC design.

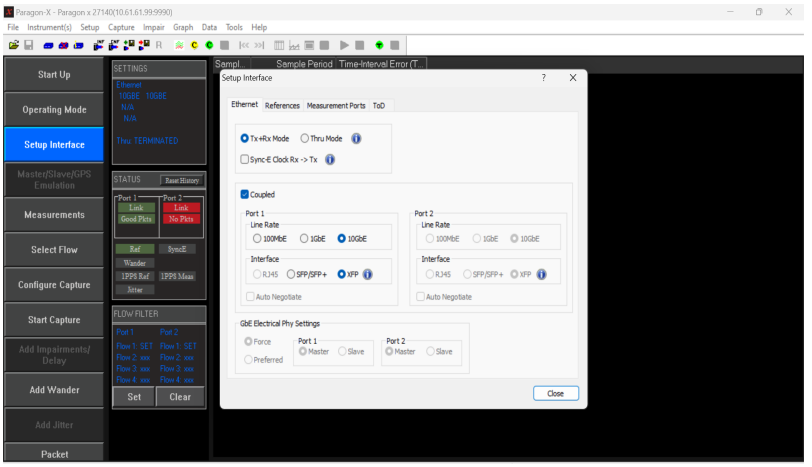
Figure 3-1. Validation Setup Diagram



To run the ESMC demo, perform the following steps:

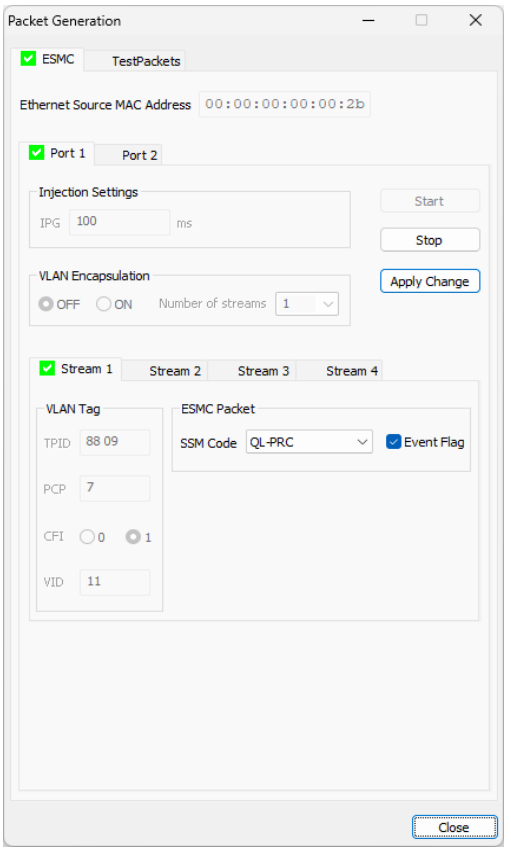
1. Extract the contents of the `mpf_an5466_df.zip` file.
2. For Programming the design onto the PolarFire Evaluation board, see Appendix 1: Generation of Design and Appendix 2: Programming the Device Using FlashPro Express sections.
3. For installation of ESMC GUI, double-click the `setup.exe` found in the location `mpf_an5466_df/GUI` folder.
4. Follow the instructions displayed on the installation wizard. After successful installation, ESMC_Demo_GUI appears on the **Start** menu of the host PC desktop.
5. The reference design is validated using the Paragon-X tester. Configure the Paragon-X tester to transmit 10G Ethernet data including ESMC PDU's and start the traffic on port 1.

Figure 3-2. Paragon-X Tester GUI



6. Open **Packet Generation** tab and set the **IPG** to 100 ms and select **SSM Code** as QL-PRC and click **Start**.

Figure 3-3. Configuring Paragon-X Tester



7. From the **Start** menu, click **ESMC_Demo_GUI** to start the GUI application.

The GUI detects the COM port number and automatically connects to the PolarFire Evaluation board. Port numbers may vary.

3.1 Running the Demo (Ask a Question)

To program the PolarFire device, perform the following steps:

1. Ensure that the jumper settings on the board are as listed in the following table.

Table 3-1. Jumper Settings for PolarFire Evaluation Board

Jumper	Description
J18, J19, J20, J21, and J22	Short pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Short pin 1 and 2 for programming through the on-board FlashPro Express
J26	Short pin 1 and 2 for programming through the FTDI SPI
J27	Short pin 1 and 2 for programming through the FTDI SPI
J39	Short pin 1 and 2 for enabling the TX
J4	Short pin 1 and 2 for manual power switching using SW3

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the host PC to the J5 connector (FTDI port) on the board using a USB cable.
4. Turn on the board using the SW3 slide switch.

The following figures show the validation setup diagram for programming the device and running the reference design.

Figure 3-4. PolarFire ECMC Demo GUI—Before the Connection

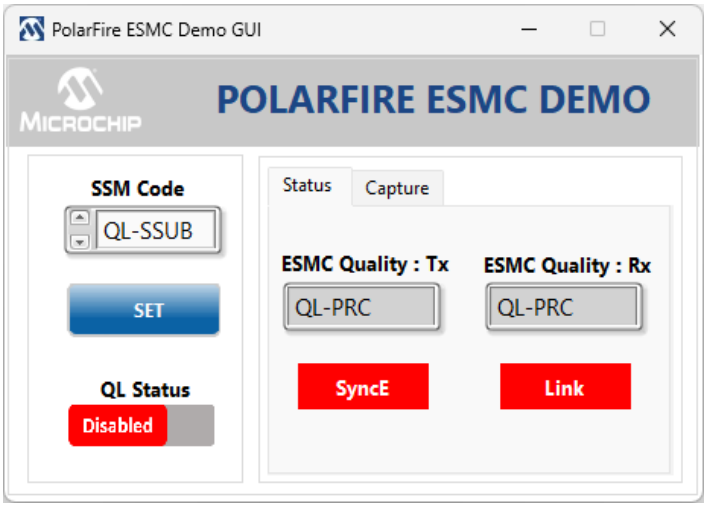
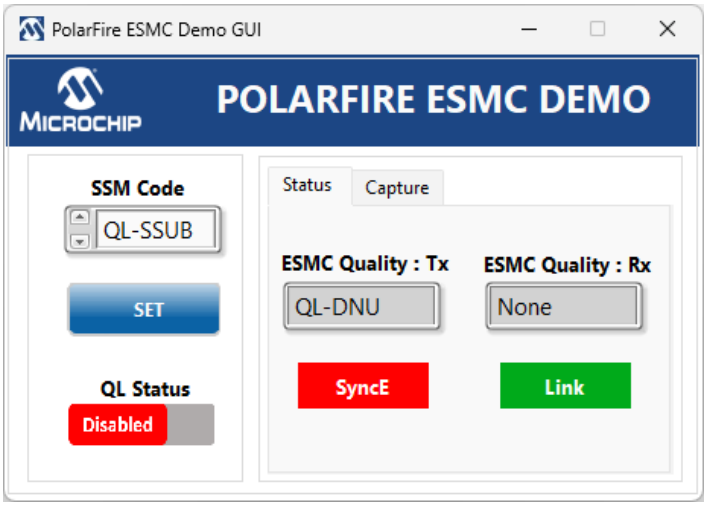
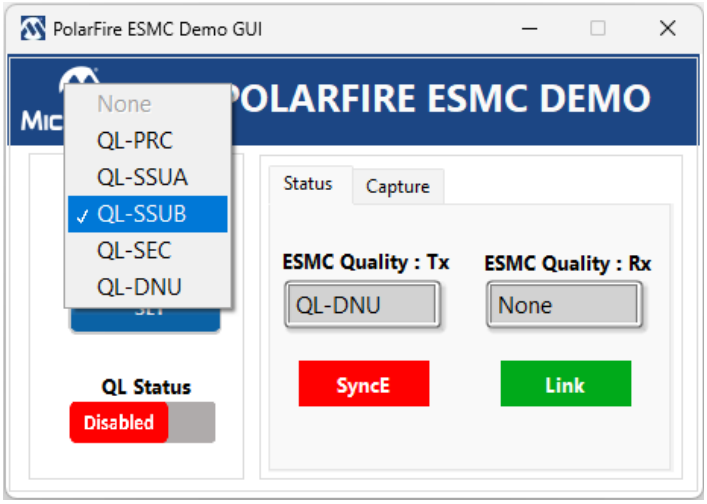


Figure 3-5. PolarFire ECMC Demo GUI—After the Connection



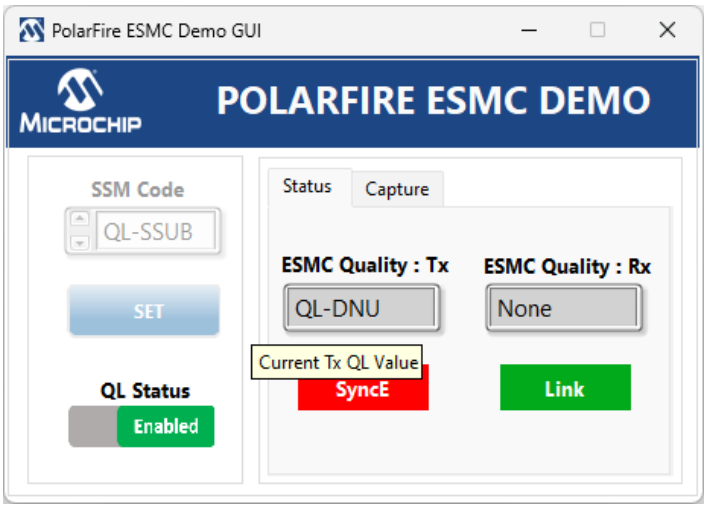
The preceding figure shows that the connection is established by changing the GUI title background to deep blue color. The **Link** button turns green to indicate that the link is established between PolarFire evaluation kit and paragon-x tester.

Figure 3-6. PolarFire ESMC SSM Code



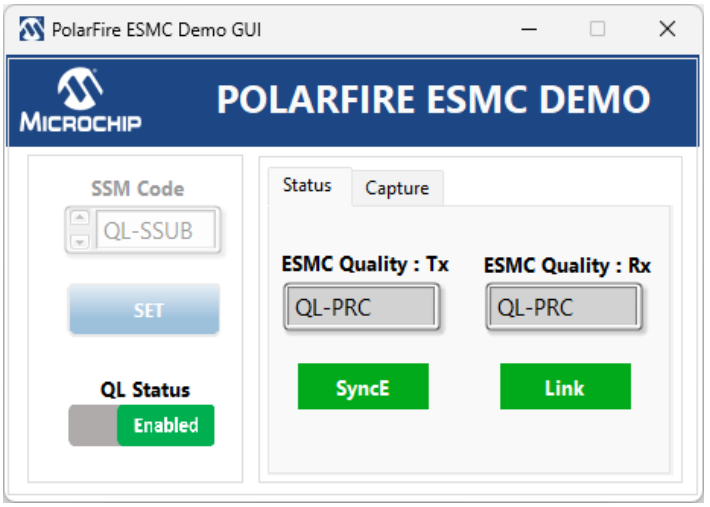
Selecting the **SSM Code**: To configure the ESMC IP with QL value, select any one SSM code from the drop-down list and click **SET** button.

Figure 3-7. PolarFire ESMC QL Status



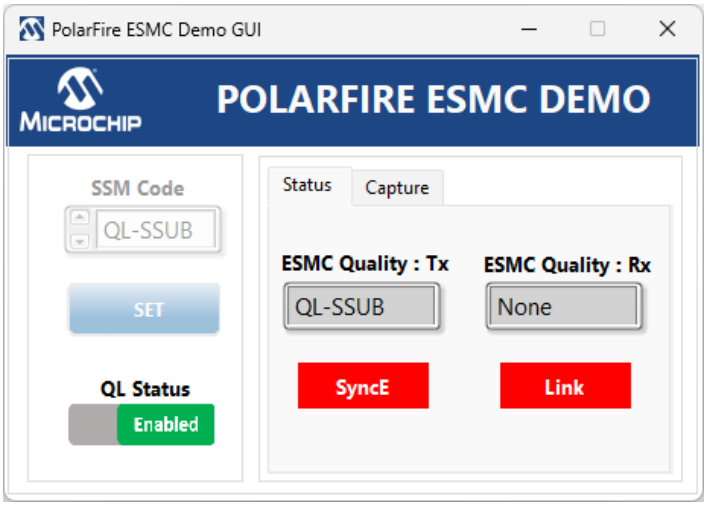
QL Status: To enable the QL mode for the ESMC IP to start accepting the ESMC PDU's, slide the **QL Status** button to the right.

Figure 3-8. PolarFire ESMC SyncE



SyncE: **SyncE** button turns green when the ESMC IP has locked to the external clock source as the other node has better clock quality.

Figure 3-9. PolarFire ESMC Link



Link: When there is no link fault the **Link** button remains green and turns red in case of a link fault.

3.2 Validating the G.8262 Short term Phase Transient Response (Ask a Question)

The device is forced into holdover for 15 seconds by manually disconnecting the optical cable form paragon tester to the PF_Evalkit. For more information, see Break-link and Make-link Conditions Handling.

3.2.1 Results (Ask a Question)

According to G.8262, short term phase transient response for option 1 the initial phase error must not exceed 120 ns within a duration of 16 ms from the event of link loss. The maximum phase error must not exceed 1000 ns at the end of 15 seconds window.

The following TIE and MTIE graphs are captured for the preceding validation steps:

Figure 3-10. TIE Analysis

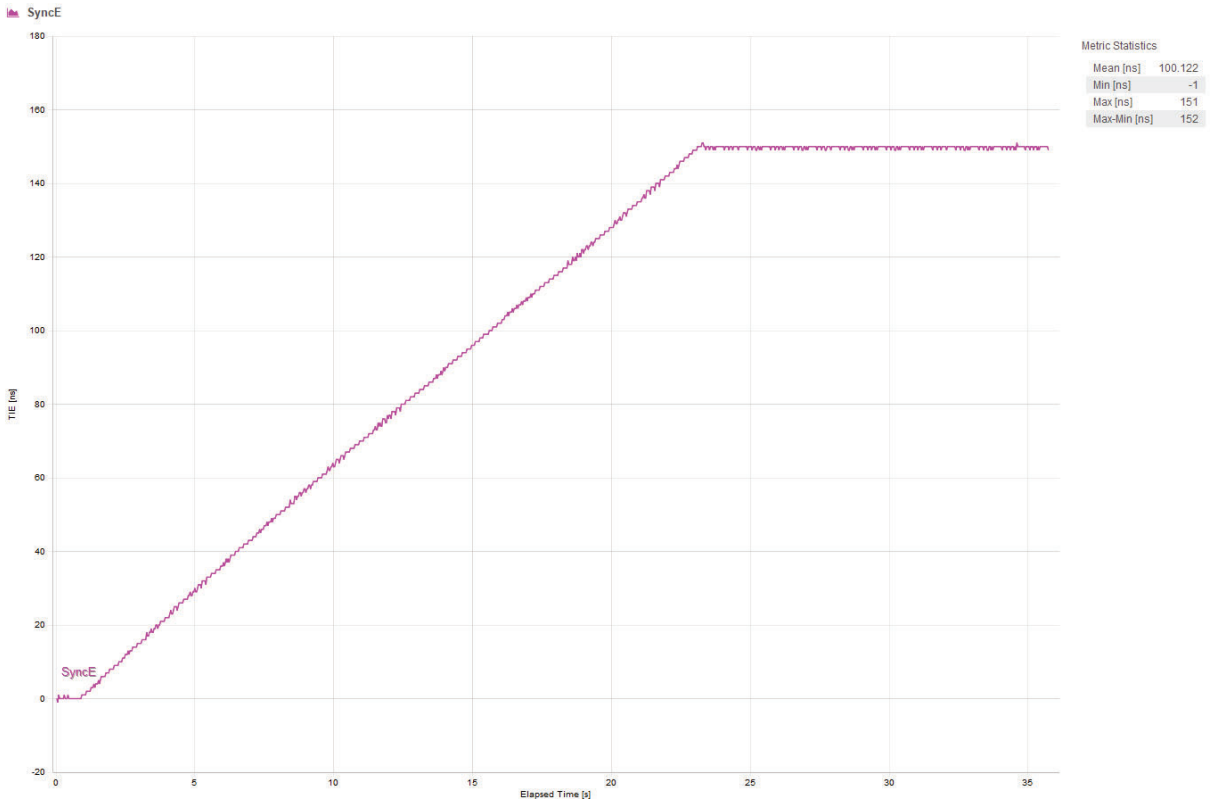
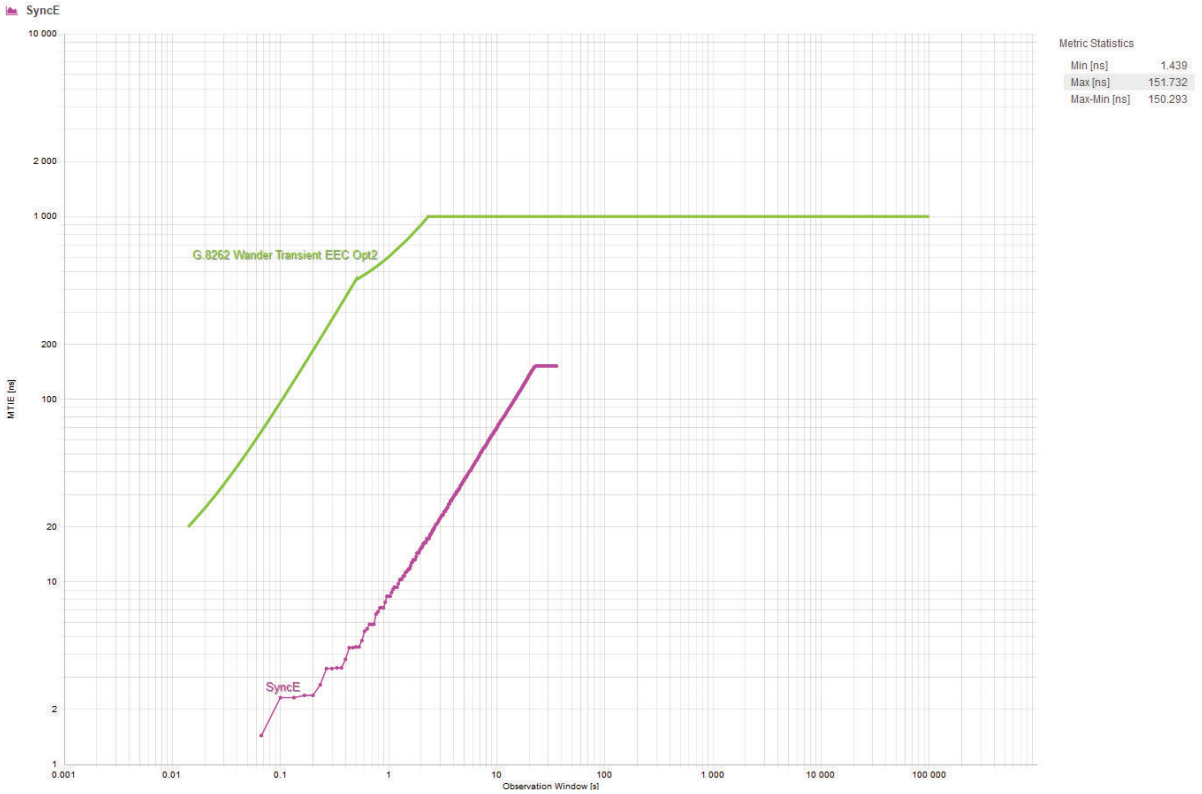


Figure 3-11. MTIE Analysis

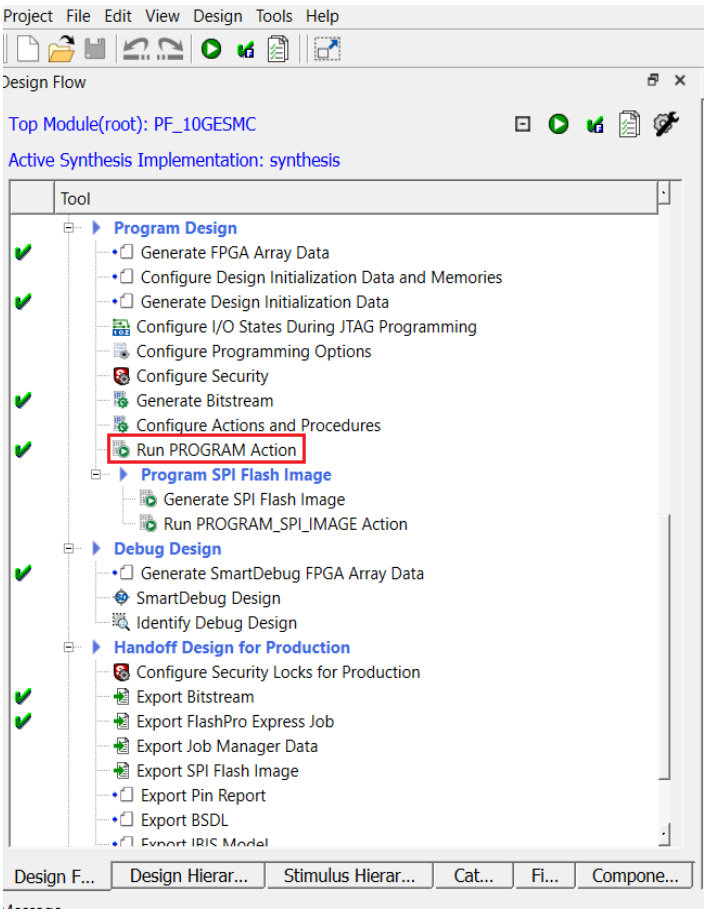


4. Appendix 1: Generation of Design (Ask a Question)

To execute TCL Script, perform the following steps:

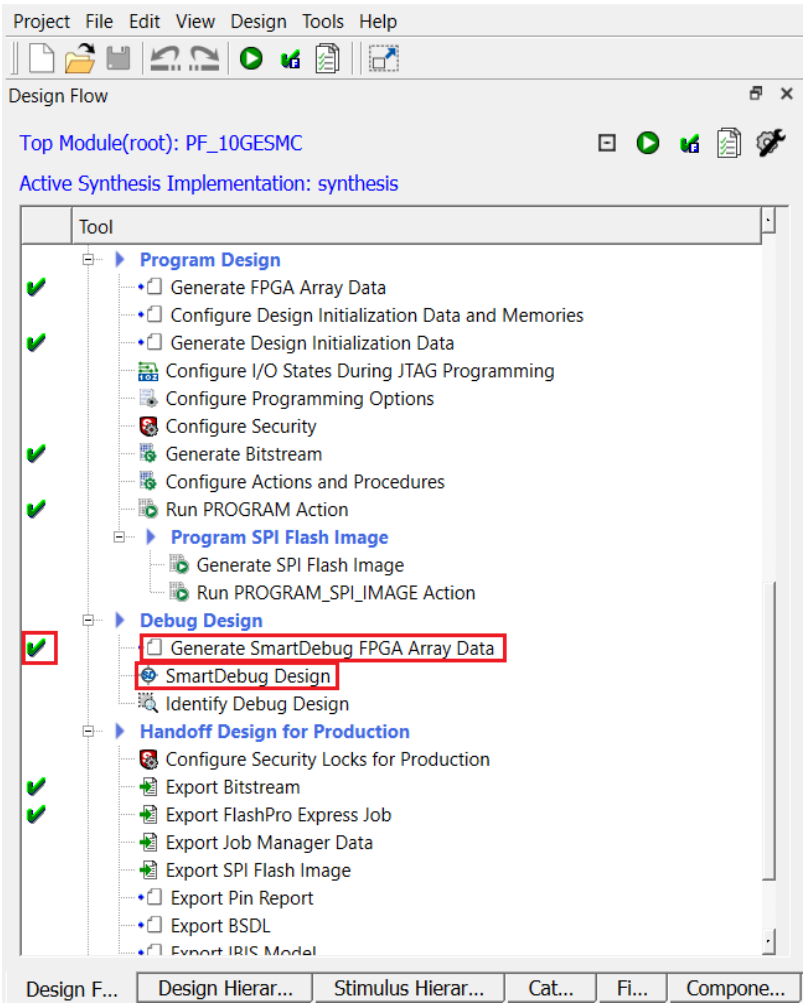
1. Launch Libero® v2024.1 or recent tool.
2. Navigate to **Project** > **Execute Script**.
3. Select the `script.tcl` file located in `mpf_an5466_df\HW\TCL_Scripts\` folder of the extracted files. After successful execution of TCL scripts, Libero project is created within `TCL_Scripts` directory.
   The script will be executed creating a Libero Project in the directory `mpf_an5466_df\HW\TCL_Scripts\PF_10GESMC`.
4. Close the **Script Execution** report and click **Project** > **Open Project**.
5. In the directory `mpf_an5466_df\HW\TCL_Scripts\PF_10GESMC`, select `PF_10GESMC.prjx`. The created project is opened.
6. To program the evaluation kit while it is connected to the PC and slider switch **SW3** is turned ON, click **Run PROGRAM Action**.

Figure 4-1. Run PROGRAM Action



7. After programming is completed, click **Generate SmartDebug FPGA Array Data** option.

Figure 4-2. Generate SmartDebug FPGA Array Data



8. Launch SmartDebug Design.
9. In the SmartDebug Interface, click on **File** > **Execute Script** and select the `japll_params.tcl` located in the directory `mpf_an5466_df\HW\JAPLL_Configuration\` of the extracted files.

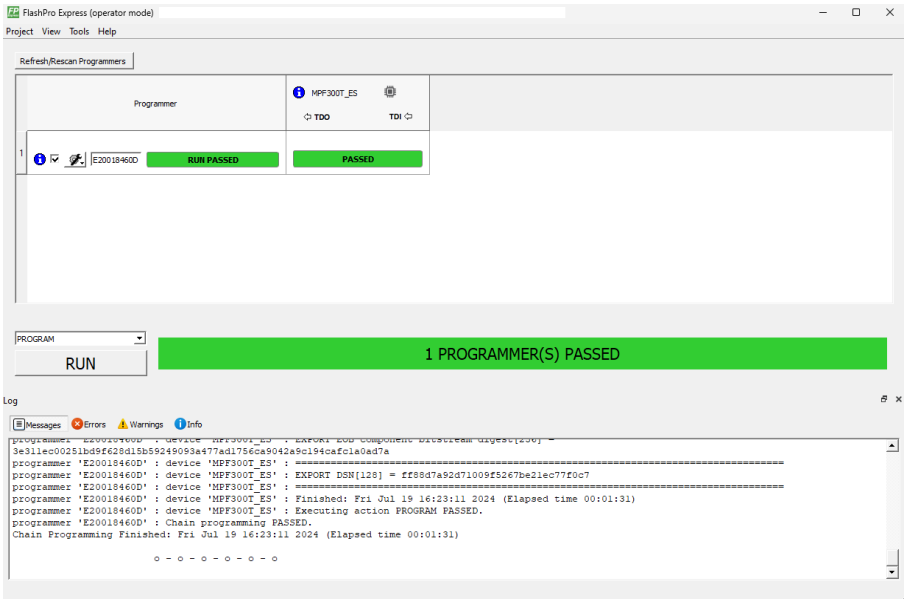
Important: Steps 9 and 10 must be repeated after every power cycle of the evaluation kit.

5. Appendix 2: Programming the Device Using FlashPro Express (Ask a Question)

This section describes how to program the PolarFire device with the programming job file using FlashPro Express. To program the device, perform the following steps:

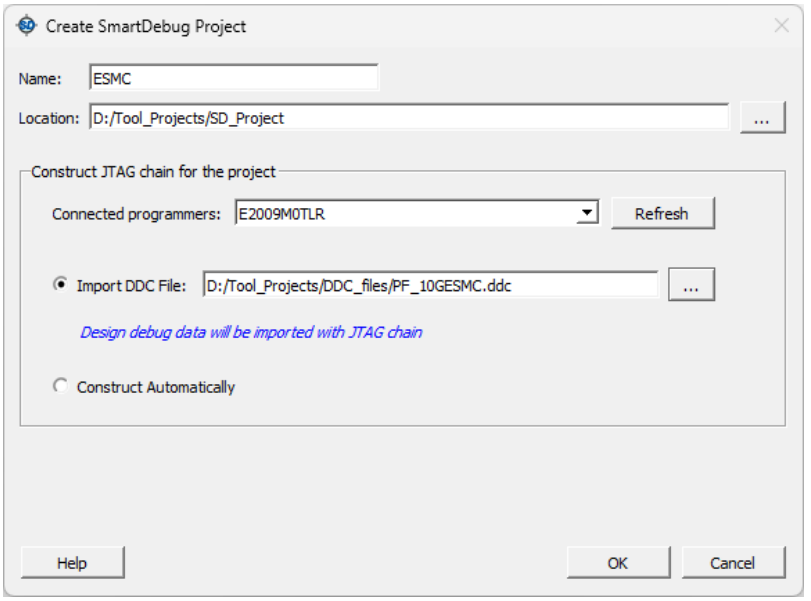
1. On the Host PC, launch the **FlashPro Express** software from its installation directory.
2. To create a new job project, click **New** or select **New Job Project** from **Project** menu.
3. Enter the following in the **New Job Project** from **FlashPro Express Job** dialog box:
   - **Import FlashPro Express job file**: Click **Browse**, and navigate to the location where the job file (`top.job`) is located and select the file. The default location is `mpf_an5466_df\Programming_File`.
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.
4. Click **OK**. The required programming file is selected and ready to be programmed in the device.
5. The FlashPro Express window appears as shown in following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**.
6. Click **RUN** after ensuring that SW3 switch is turned ON. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 5-1. FlashPro Express—RUN PASSED



7. Close **FlashPro Express** or click **Exit** in the **Project** tab.
8. Launch **SmartDebug** on the host PC, and to create a new SmartDebug project, select **New**.
9. Enter the name and project location where the project should be stored and select the **Import DDC File** option. Select the **DDC File** provided in the directory `mpf_an5466_df\SmartDebug_File` and click **OK**, as shown in the following figure.

Figure 5-2. SmartDebug Project Creation



10. Once the SmartDebug project opens, navigate to **File** > **Execute Script**. Select the `japll_params.tcl` script in the directory `mpf_an5466_df\HW\JAPLL_Configuration` and click **Run**.
11. Upon successful completion of the script execution, close **SmartDebug**.

Important: Repeat step 10 and 11 after every power cycle.

6. References (Ask a Question)

The following references are used in the document:

- Timing characteristics of synchronous equipment slave clock
- PolarFire Family Transceiver User Guide
- PolarFire FPGA Datasheet
- Core10GMAC User Guide
- Distribution of timing information through packet networks (G.8264)
- Timing and synchronization aspects in packet network

7. Revision History (Ask a Question)

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

Table 7-1. Revision History

Revision	Date	Description
A	10/2024	The following is the list of changes in revision A of the document: • The document was migrated to Microchip template. • The document number was updated to DS00005466A from 50200898. • The document ID was updated to AN5466 from DG0898.
1.0	11/2021	Initial release

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

ISBN: 978-1-6683-0489-1

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.