TI Designs
Parallel Redundancy Protocol (PRP) Ethernet Reference Design for Substation Automation

TEXAS INSTRUMENTS

Design Overview

This TI Design is a solution for high reliability, low-latency network communications for substation-automation equipment in smart grid transmission and distribution networks. It supports the Parallel Redundancy Protocol (PRP) specification in the IEC 62439 standard. This solution is a lower-cost alternative to FPGA approaches and provides the flexibility and performance to add features such as IEC 61850 support without additional components.

Design Resources

TIDEP0054	Design Folder
AM3359	Product Folder
TLK110	Product Folder
TPS65910	Product Folder
TMDSICE3359	Tools Folder
SYSBIOSSDK-IND-SITARA	Tools Folder

TI E2E™ Community

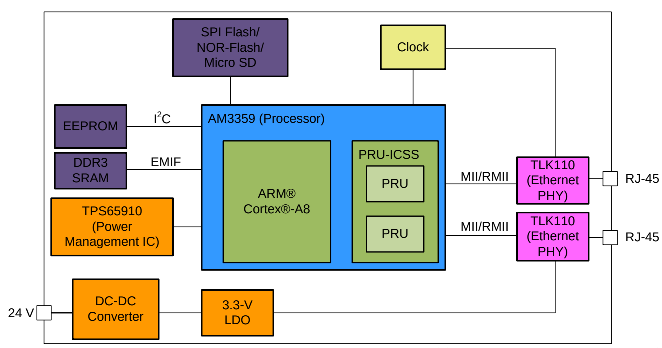
ASK Our E2E Experts

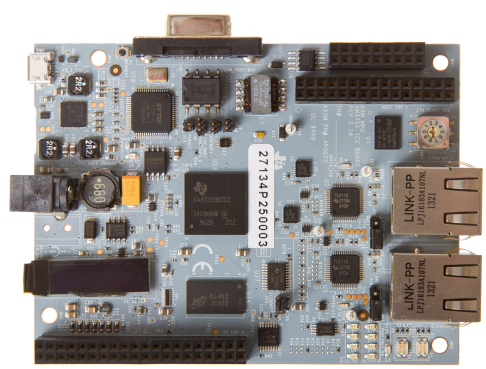
Design Features

- Compliant to IEC 62439-3 Clause 4 Specification for PRP-Ethernet Communications
- Traffic Filtering Based on Virtual Local-Area Network (VLAN) IDs, Multicast and Broadcast Support, and Built-in Storm Prevention and Supervision Mechanism
- Zero Recovery Time in Case of Network Failure
- Dual-Ported Full-Duplex 100-Mbps Ethernet
- Fully Programmable Solution Provides Platform for Integration of Additional Applications

Featured Applications

- Substation and Distribution Automation
- Protection Relays
- Smart-Grid Communication
- Factory Automation





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 Background

A substation is a key component of the electricity-grid infrastructure, located everywhere from power generation facilities throughout the distribution network to the low-voltage feeders serving residences and businesses. Substations are a primary factor in transforming voltage levels for transmission and performing important functions such as switching, monitoring, and protecting sub-systems in order to maintain grid efficiency and reliability. Traditional-substation systems focused on fault monitoring that can be manually fixed by switching to backup subsystems.

Consumers, regulators, and grid operators demand increasing reliability of electricity delivery. The introduction of automatic switching and protection of subsystems increases the demand for the automation of substation operations and communications to monitor grid conditions and communicate information to grid operators.

Operators need to continually monitor the health of networks and take action to maintain the operation with efficiency. This need leads to the requirement for reliable and low-latency communications between the control center of the operator and high-value nodes such as substations.

The International Electro-Technical Commission (IEC) released specifications for industrial-Ethernet communications under the IEC 62439 standard. The PRP specification is one of the IEC 62439-3 standards that provides a static redundancy Ethernet-based protocol that supports critical real-time systems that require continuous monitoring.

2 System Description

This design details a reliable high-speed PRP communication solution that is compliant with IEC 62439-3 Clause 4 for substation automation.

This is a cost-effective alternative to ASIC- or FPGA-based Ethernet solutions while delivering equal performance. The programmable nature of the solution allows operating different redundancy Ethernet protocols without hardware modification and adding applications such as IEC 61850 without requiring additional system costs.

Figure 1 shows the overall-system architecture. The PRP supports dual-port full-duplex Ethernet communication over star topology. The system includes Ethernet PHY as layer 1 and Ethernet MAC and PRP as layer 2. An application-example project, provided in the software-development kit (SDK) software package [5], enables developers to build applications on top of those protocols through direct API calls or by using network-development kit (NDK) libraries to import network stacks. Hard real-time application can be implemented by using direct API call, and a TCP/IP based application can be implemented by using NDK stacks provided by TI.

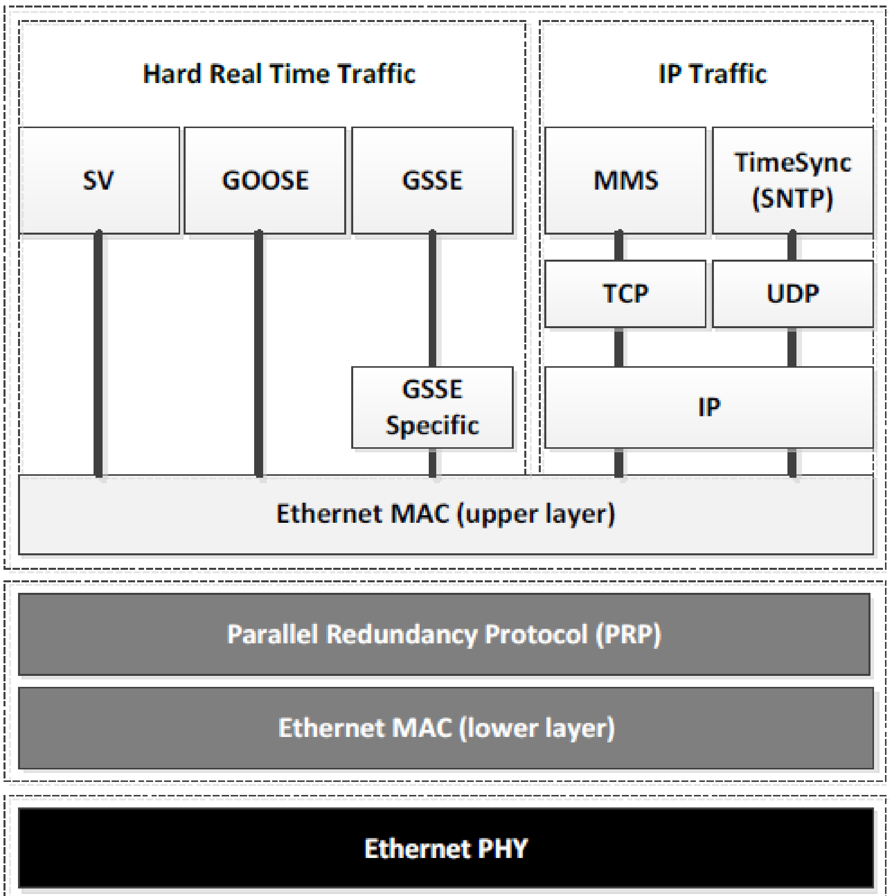


Figure 1. System Architecture

2.1 Parallel Redundancy Protocol (PRP)

PRP is a redundancy protocol for Ethernet networks (standardized using IEC 62439-3 Clause 4) that is selected as one of the redundancy protocols for substation automation in the IEC 61850 standard. PRP is application-protocol independent and can be used by most industrial-Ethernet applications that require reliable high-speed communications.

The PRP supports star topology, which ensures a fixed-hop delay with cost of infrastructure.

Figure 2 shows PRP operation over star topology. Once a packet is generated at Node 1, the same packet is sent through LAN A and B, and the destination (Node 3) receives duplicated packets if both deliveries succeed. The redundancy provides zero-recovery time if the packet fails to be delivered in one direction.

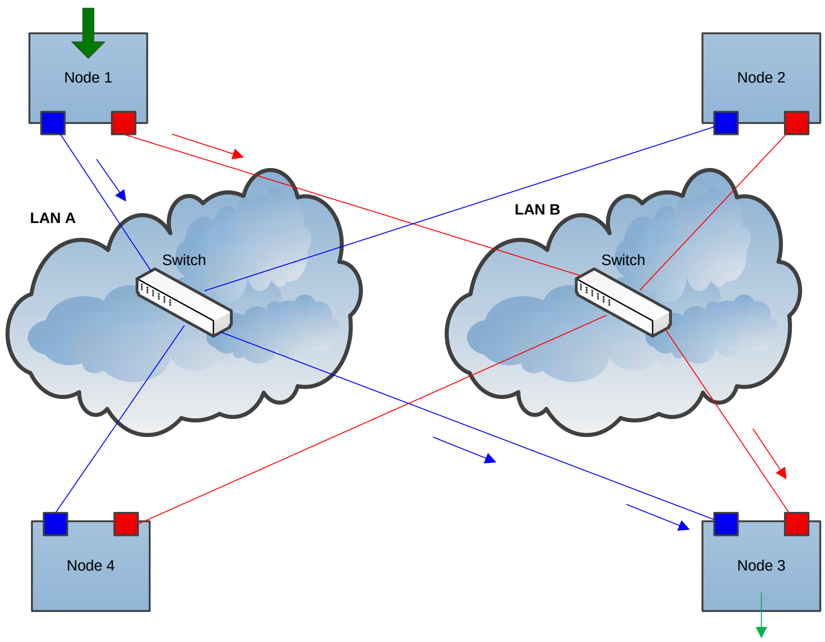


Figure 2. PRP Star Topology

2.2 Ethernet PHY

The Ethernet PHY, based on IEEE standard 802.3, is responsible for transmitting and receiving data over Ethernet lines.

3 Block Diagram

Figure 3 shows the block diagram. The primary devices for this design are the AM3359, TLK110, and the TPS65910. The AM3359 ARM® Cortex®-A8 microprocessor is the host processor used to support PRP and user applications running under a TI real-time operating system (RTOS) environment. The TLK110 is the Ethernet PHY device and two TLK110s are used to create redundant-Ethernet communications in this design. The TPS65910 is an integrated-power management IC (PMIC) with four DC-DCs, eight low dropouts (LDOs), and a real-time clock (RTC).

These devices were chosen because of the:

- Programmable real-time unit subsystem and industrial-communication subsystem (PRU-ICSS) that allows independent operation for real-time communication stacks
- High-performance ARM core that allows supporting the real-time applications for substation automation
- Programmable-flexible design that can be upgraded to different Ethernet-based redundancy protocols without hardware modification
- High-performance Ethernet PHY to meet the requirements of IEEE 802.3 with high margins in terms of cross-talk and alien noise

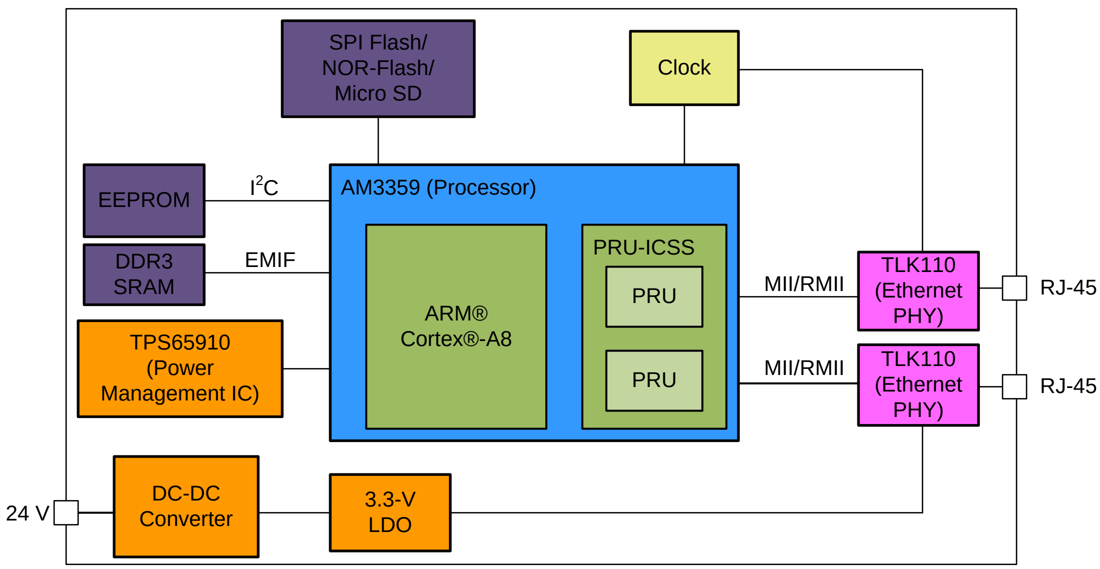


Figure 3. Block Diagram

3.1 AM3359

The AM3359 microprocessors (based on the ARM Cortex-A8 processor) are enhanced with image, graphics processing, peripherals, and an industrial-interface option for PRP and high-availability seamless redundancy (HSR). The PRU-ICSS is separate from the ARM core, allowing for independent operation and clocking for increased efficiency and flexibility. The PRU-ICSS unit contains two PRUs; each PRU includes a 32-bit RISC processor capable of running at 200 MHz that supports real-time protocol for PRP and HSR. The PRUs support additional media-independent interfaces (MIIs) and reduced media-independent interfaces (RMIIs) to connect to the Ethernet PHY devices directly.

The programmable nature of the PRU-ICSS, the access to pins, events, and all system-on-chip (SoC) resources provide flexibility when implementing fast, real-time, specialized-data handling (see Figure 4).

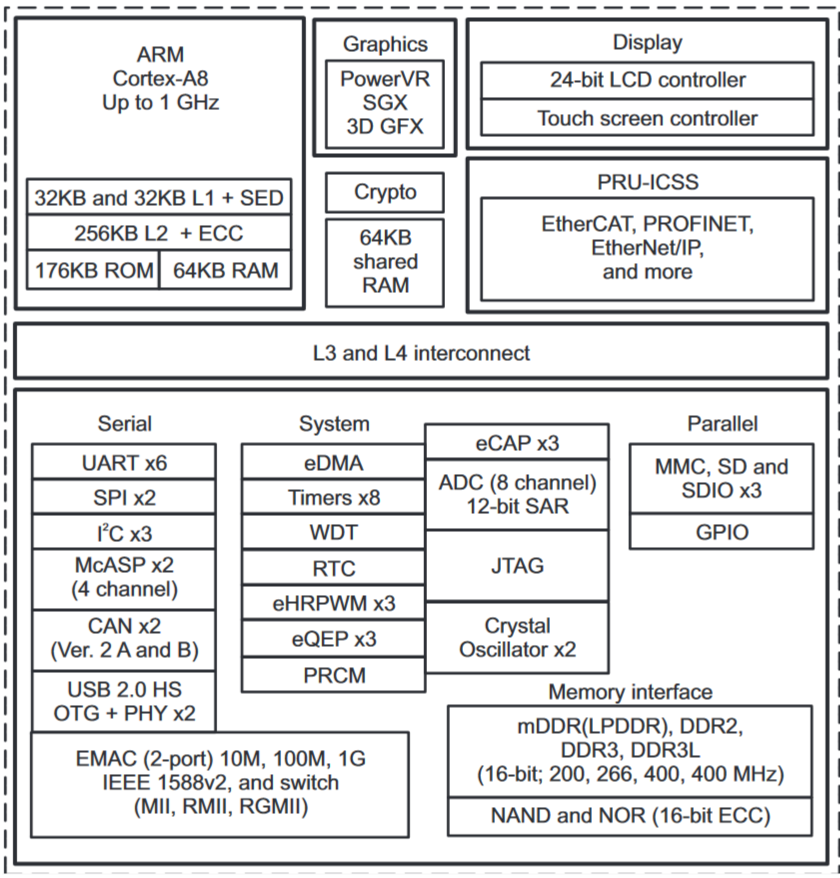


Figure 4. AM3359 Functional Block Diagram

3.2 TLK110

The TLK110 (see Figure 5) is a single-port Ethernet PHY for 10BASE-T and 100BASE-TX signaling. This device integrates all of the physical-layer functions needed to transmit and receive data on standard twisted-pair cables. The TLK110 supports the standard MII and RMII for direct connection to a media-access controller (MAC). The TLK110 is designed for power-supply flexibility and can operate with a single 3.3-V power supply. The TLK110 can also operate with combinations of 3.3- and 1.55-V power supplies for reduced power operation. The TLK110 uses mixed-signal processing to perform equalization, data recovery, and error correction to achieve robust operation over CAT 5 twisted-pair wiring. This device meets the requirements of IEEE 802.3 and maintains high margins in terms of cross-talk and alien noise.

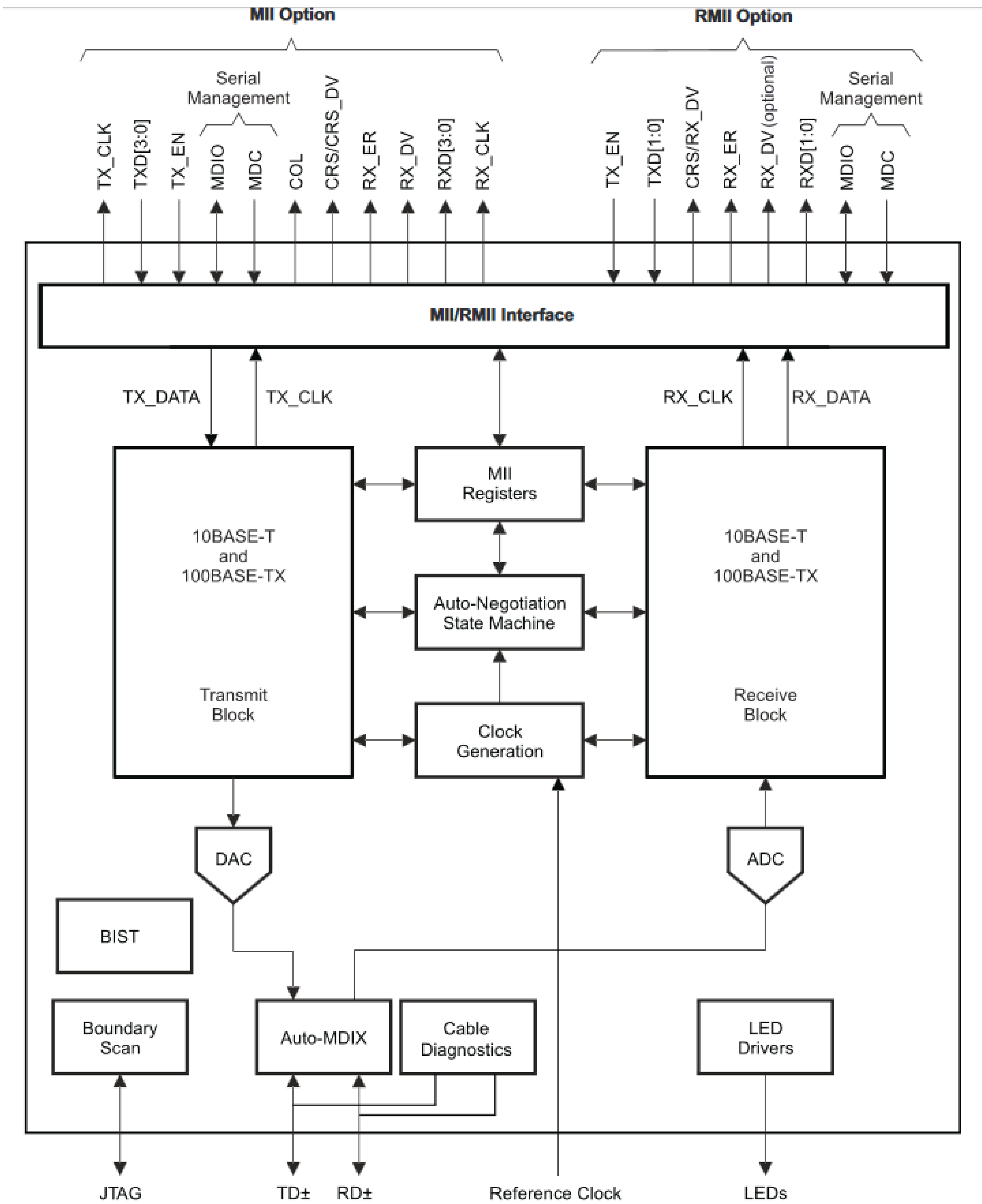


Figure 5. TLK110 Functional Block Diagram

3.3 TPS65910

The TPS65910 (see Figure 6) is an integrated power-management IC that provides three step-down converters, one step-up converter, eight LDOs, and is designed to support the specific power requirements of OMAP-based applications. Two of the step-down converters provide power for dual processor cores and are controllable by a dedicated class-3 SmartReflex™ software for optimum-power savings. The third converter provides power for the IOs and memory in the system. The device includes eight general-purpose LDOs providing a wide range of voltage and current capabilities. The LDOs are controllable by the I^2C interface. The use of the LDOs is flexible; they are intended to be used in the following ways:

- Two LDOs are designated to power the phase-locked loop (PLL) and video digital to analog converter (DAC) supply rails on the OMAP-based processors
- Four general-purpose auxiliary LDOs are available to provide power to other devices in the system
- Two LDOs are provided to power double-data rate (DDR) memory supplies in applications requiring these memories

The device contains an embedded-power controller (EPC) to manage the power sequencing requirements of the OMAP systems and an RTC.

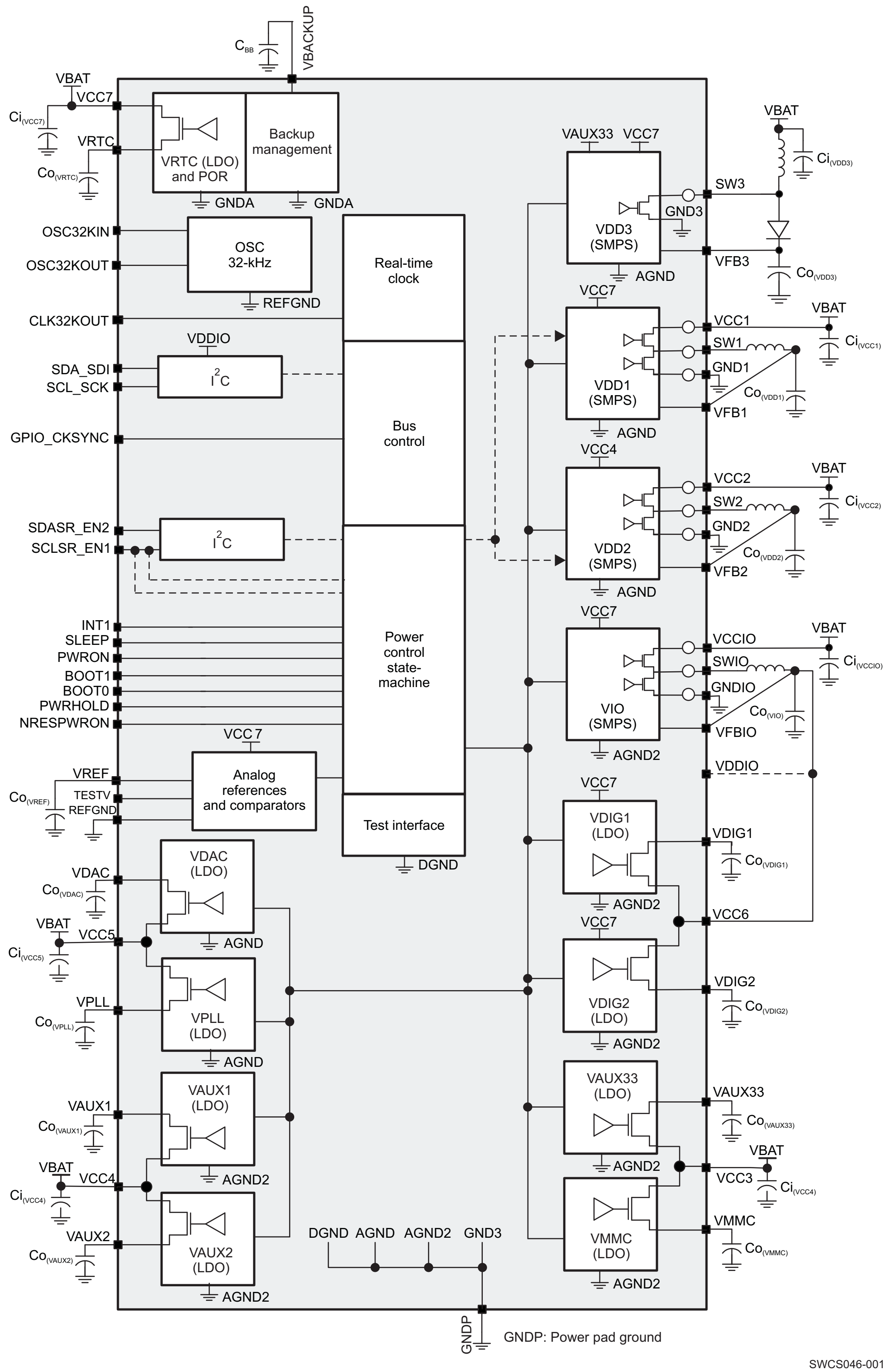


Figure 6. TPS65910 Functional Block Diagram

4 Getting Started Hardware

Figure 7 shows AM3359 ICE (Industrial Communication Engine) EVM revision 2.1A. The EVM includes various flash devices that are supported by the AM3359, DDR memory, and power-management devices. The EVM is designed to support multiple communication standards by providing various interfaces such as Ethernet, CAN, and RS-485.

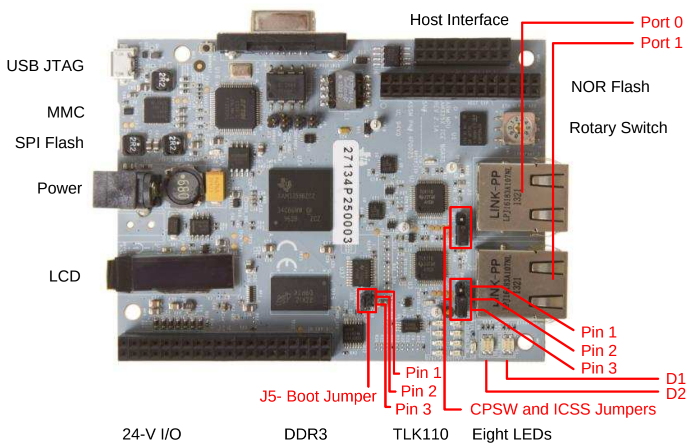


Figure 7. ICE EVM Revision 2.1A

4.1 EVM Configuration

The ICE EVM provides multiple boot options of NOR-flash, SPI-flash, multi-media card (MMC), and secure-digital (SD) boot. The boot mode can be configured with J5. Table 1 details the jumper settings.

Table 1. Boot Options

BOOT MODE	JUMPER (J5) CONFIGURATION
NOR-Flash	Close pin 1 to 2
SPI-Flash	Close pin 2 to 3
MMC and SD	Close pin 2 to 3

4.2 An Example: Two-Node Setup

Each node has two Ethernet ports, and each is connected to a different Ethernet switch. Figure 8 shows an example of a two-node setup. An ordinary Ethernet switch can be used for testing and each node is connected to a serial-terminal program with a baud rate of 115,200 bps for monitoring and configuration.

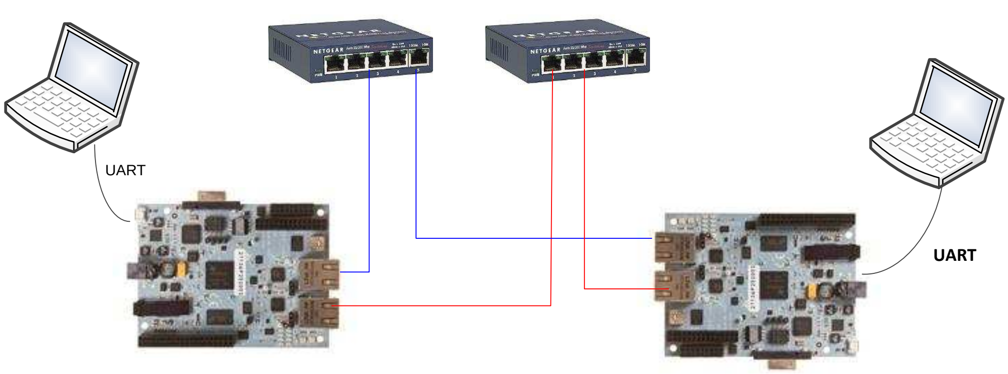


Figure 8. Two-Node Setup

5 Getting Started Firmware

This section provides a step-by-step procedure to develop an application using PRP protocol based on the SYSBIOS industrial SDK for the Sitara™ Processor.

1. Check out the SDK package *SYSBIOSSDK-IND-SITARA*.
2. Install the SDK package.
3. Set an environment variable of “IA_SDK_HOME” to the SDK installation directory. Figure 9 shows the default SDK installation directory.

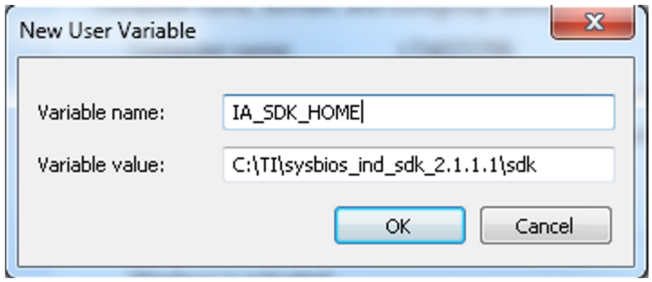


Figure 9. Default SDK Installation Directory

For more details on the software architecture and system and tool requirements, refer to the user guide documents in *$(IA_SDK_HOME)\docs*.

5.1 Overview

The baseline for application development is the prp_app CCS project in *$(IA_SDK_HOME)\examples\hsr_prp_app\prp*.The example project initializes the low-level drivers and creates and runs some example tasks. These tasks can be modified depending on end-user applications. The pre-built binaries for prp_app project are available in *$(IA_SDK_HOME)\examples\hsr_prp_app\prp\am335x_debug*.

Section 5.2 shows how to build the prp_app CCS project. Section 5.3 is an overview of the prp_app example project. Section 5.4 shows how to flash and debug the applications.

5.2 Building prp_app CSS Project

The prp_app example project includes lower-level driver initialization, PRP protocol configuration, task creations, and host-level applications. There are four tasks running in the example project: taskPruss, taskLedBlink, taskRedDebug, and taskTX. The taskPruss is a task to load PRP firmware into the DDR memory and initialize necessary NDK stacks (see SNMP). The taskLedBlink is responsible for controlling LEDs. The taskRedDebug is a task to print statistics and node tables periodically. The taskTX includes a host-level application to send Ethernet frames periodically through direct call to the TX API.

Use these steps to build the prp_app project:

1. Open the CCS project in *$(IA_SDK_HOME)\examples\hsr_prp_app\prp*.
2. Set the build configuration to am335x_debug and then build.

Successful compilation generates three types of binaries: *prp_app.out*, *prp_app.bin*, and *prp_app_ti.bin* found in *$(IA_SDK_HOME)\examples\hsr_prp_app\prp_app\am335x_debug*. The *prp_app.out* can be used to run in a debug mode, and the *prp_app_ti.bin* binary can be flashed into SPI_Flash to run as a standalone mode.

5.3 Example Project—prp_app

This section shows the overview of the prp_app example project to transmit and receive packets.

5.3.1 Transmit Packet

There are two ways to transmit packets: call the TX API directly or use the NDK stack lib [4]. An example of direct API call-to-transmit packet is given in the prp_app example. The taskTX is the primary task to transmit Ethernet frames, which creates a 100-B broadcast Ethernet frame and sends the frame through TX API every one second. The TX API is registered in the main(), shown in the following code.

```
((((ICSSEMAC_Object *)emachandle->object)->callBackHandle)->txCallBack)->callBack
= (ICSS_EmacCallBack)RedTxPacket;
```

The first argument to be set for the API call is the port number; this indicates the port number to be used for transmission. The PRP firmware ignores this because it transmits a packet through both ports.

The queue priority ranges from 0 (ICSS_EMAC_QUEUE1) to 3 (ICSS_EMAC_QUEUE4). Lower queue numbers are higher priority, and transmission occurs in the order of priority. The packet length is the MAC PDU frame size. The *txArg.srcAddress* leads to the address where the TX packet is located. The TX packet is in forms of Ethernet MAC frame that consist of the Ethernet MAC header and payload. The Ethernet MAC header includes a 6-byte destination address, a 6-byte source address, and a 2-byte Ethernet Type. The 4-byte cyclic-redundancy check (CRC) is not required at the application level. The underlying PRP appends the CRC after adding the PRP trailer to the MAC frame. In this example, a 2-byte Ethernet type is set to 0x0800 (IPv4).

The Ethernet type can be set to a VLAN tagged frame with an optional 802.1-Q header. Using a VLAN tag allows prioritized packet reception based on the PCP (priority-code point) subfield in the 802.1Q header. When a packet is received in the PRP firmware, the 3-bit PCP subfield of a VLAN tag is read and the packet is copied to the appropriate RX queue based on fixed mapping that maps two levels (out of eight) of quality of service (QoS) to one queue. When the Ethernet type is set to 0x0800, the packet is received at RX queue three.

5.3.2 Receive Packet

Similar to the TX packet, there are two ways to receive packets: hook the RX callback directly or use a NDK stack lib [7]. Using the NDK stack lib allows the application to hook the RX callback when the NDK lib stacks automatically. If the application is time-critical and there is not a need for network stacks, it is recommended to use the RX callback directly. An example of how to use the direct RX callback is given in the prp_app example.

There are three steps to hook the RX callback directly. Typically, the first two steps can be completed in the initialization routine (in the main). In the first step, the ethPrioQueue allows the PRP firmware to determine whether the RX interrupts to the NDK lib stacks or the RX callback when the RX receives frames. The RX maintains four RX queues, and in this example the ethPrioQueue is set to four.

This means that the PRP firmware alerts the packet reception to the RX callback when the frames are received in a queue number less than four. Because the RX maintains four queues, this configuration forces the received frames to send to the direct RX callback. However, if the ethPrioQueue is set to 0, the PRP RX sends all of the received frames to NDK stacks. This provides an effective way to enable the application to handle the mixed traffic of hard real-time traffic (through direct call) and TCP/IP traffic (through NDK stacks).

Step two is to hook the RX callback as shown in the following code, which can be found in taskPruss in the prp_app example codes. Step three is to copy the RX packet into a local buffer when an interrupt occurs. In this step, it is mandatory to call the RX API as shown in the following code to flush and copy the RX Queue. Otherwise, it blocks subsequent-packet receptions.

The prp_app example includes some codes to handle both precision time protocol (IEEE 1588 PTP or PTP) frames and application data through address comparison (shown in the processHighPrioPackets).

```
//Register packet processing callback (in taskPruss)
    ((((ICSSEMAC_Object *)
       emachandle->object)->callBackHandle)->rxRTCallBack)->callBack =
           (ICSS_EmacCallBack)processHighPrioPackets;

// RX API call to copy/flush the RX Queue (in processHighPrioPackets)
((((ICSSEMAC_Object *)
       emachandle->object)->callBackHandle)->rxCallBack)->callBack =
           (ICSS_EmacCallBack)RedRxPktGet;
```

NOTE: The SYSBIOSSDK-IND-SITARA v2.1.2-software package does not support PTP for the PRP protocol. For the PTP, refer to 9.

5.4 Flashing Binaries to SPI Flash on ICE v2.1 Using Code Composer Studio™

This section covers how to flash *prp_app_ti.bin* binary to SPI flash using Code Composer Studio™ software. More flashing options can be found in 4.

5.4.1 Launching and Debugging Applications in CCS

This section shows the step-by-step procedure to run the application in debug mode. For product-related questions or bug reports, contact the TI E2E™ online community.

1. Connect the USB cable to the ICE EVM
2. Select *View*, then *Target Configurations*
3. Right-click on the required configuration in the list and select *Launch Selected Configuration*
4. Right-click on the Cortex A9/A8 listed in the Debug View and select *Connect* (Click *View→Debug* to view the debug window if it is not visible)
5. Select *System Reset* by clicking *Run→Reset→System Reset*
6. Select *Suspend* by clicking *Run→Suspend* in the menu
7. Load the GEL file and execute initialization script
   (a) Click *Tools→GEL Files*
   (b) Remove the existing GEL file (*TMDXICE3359.gel*) and load the GEL file *TMDXICE3359_v2_1A.gel* in the directory *$(IA_SDK_HOME)\tools\gel\AM335x*. Once the GEL file is loaded, the available scripts are shown in *Menu→Scripts*
   (c) Click the initialization script by selecting *Menu→Scripts→AM335x System Initialization→AM3359_ICE_Initialization*
8. Select *Run→Load→Load Program* once the initialization is completed

Once step 8 is completed, the application binary (for example: *prp_app.out*) can be loaded to run in the CCS debug mode. When a user begins this step with a new ICEv2 EVM, the user must select *CPU RESET (HW)* by clicking *Run→Reset→CPU RESET (HW)* between steps 5 and 6. This is a one-time requirement.

5.4.2 Erasing and Flashing Binaries to SPI Flash on ICE v2.1

1. Complete steps 1 through 8 in Section 5.4.1
2. Load the pre-built *isdk_spi_flasher.out* in the directory *$(IA_SDK_HOME)\tools\flasher\spi_flasher*
3. Run *isdk_spi_flasher.out*

The CCS console window shows what steps to follow. Refer to the section titled *Flashing Binaries to SPI Flash on ICE V1 and V2 Using CCS* in 4 for more details.

The pre-built bootloader binary can be found in *$(IA_SDK_HOME)\starterware\binary\bootloader\bin\am335x-evm\gcc*.

6 Test Setup

Figure 10 shows the test set-up with two nodes, and each node has two Ethernet ports, each of which is connected to different Ethernet switch. For these experiments, the prp_app project application was modified to measure the performance of the delivery ratio and latency. For the target TX and RX, a PC is attached to configure test modes and to capture the test results through a serial-terminal program. The underlying firmware generates background traffic, such as supervision frames, to discover neighbors.

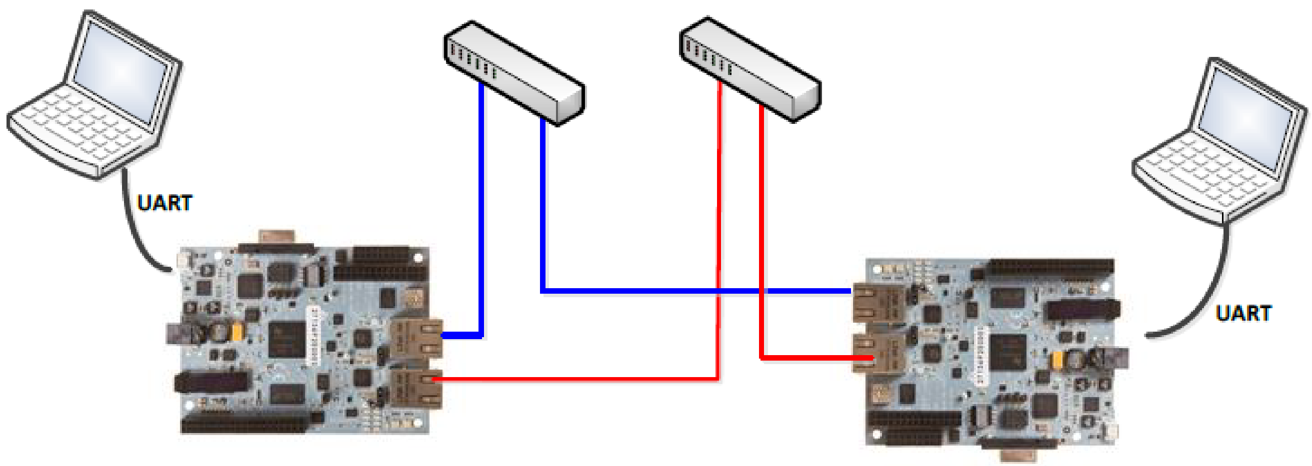


Figure 10. Two-Node Test Setup

7 Test Data

The goal of these experiments is to evaluate if the TI PRP solution meets the performance requirement for substation automation. Table 2 summarizes the performance requirement for substation automation. The required communication-recovery time is the time duration in which a network recovers from failure. The application recovery-tolerated delay (or grace time) is the time duration the substation tolerates an outage of the automation system. The sampled values (SVs) are sampled at a nominal value of 4 kHz. Therefore, the target application recovery-tolerated delay for SV is 500 µsec (= 2 × ¼ kHz).

Table 2. Recovery-Delay Requirements in IEC 61850-5

COMMUNICATING PARTNERS	SERVICE	APPLICATION RECOVERY – TOLERATED DELAY	REQUIRED COMMUNICATION – RECOVERY TIME
SCADA to IED, client-server	IEC 61850-8-1	800 ms	400 ms
IED to IED interlocking	IEC 61850-8-1	12 ms (trim set to 4 ms)	4 ms
IED to IED, reverse blocking	IEC 61850-8-1	12 ms (trim set to 4 ms)	4 ms
Protection trip excluding busbar protection	IEC 61850-8-1	8 ms	4 ms
Busbar protection	IEC 61850-9-2 on station bus	< 1 ms	Bumpless
Sampled values	IEC 61850-9-2 on processed bus	Less than two consecutive samples	Bumpless

7.1 Latency

The goal of this experiment is to validate if the latency performance meets the requirement for substation-automation applications. To measure the latency, one node is configured as TX from 100 B to 1500 B in increments of 100 B, and another node is configured as the echo-back mode to send the received packet back to the TX packet. The round-trip delay is measured at the originator of the packet by calculating the time gap between TX and RX. Then, the latency is calculated by a half of the round-trip delay. The latency measurement was performed five times and then averaged. The latency is considered a one-way delay based on the definition in the IEC and TR 61850-90-4.

Figure 11 shows latency performance as a function of frame size. The X-axis shows frame size in bytes, and the Y-axis shows latency in msec. The results shows that latency performance with a maximum frame size of 1500 B in 391 usec, which meets the delay requirement (< = 500 µsec) for a sampled-values application.

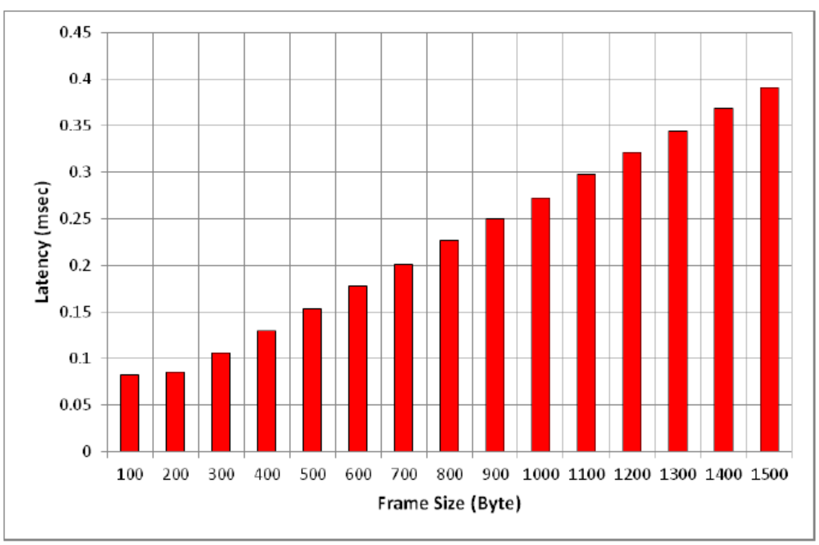


Figure 11. Latency Performance

7.2 Delivery Ratio

The goal of this experiment is to verify the zero network-recovery time requirements for substation automation. The delivery ratio performance was measured while emulating the link failures by disconnecting a link intentionally in the middle of data transmissions.

For this experiment, TX node sent 10,000 packets with a 1528-B frame size and a 1-msec packet interval. During the experiment, one of links was disconnected to emulate link failure. To further verify the impact of packet types on delivery ratio, the same experiment was performed for unicast and broadcast traffic. The delivery ratio was calculated by the number of TX packets divided by the number of RX packets.

Table 3 shows delivery ratio performance. For all scenarios, the results show a 100% delivery ratio with link failure. This result means that link failure is recovered immediately. This result is expected because redundant communication recovers the link failure immediately.

Table 3. Delivery Ratio Performance

TEST SCENARIO	DELIVERY RATIO
Unicast	100%
Broadcast	100%

8 Design Files

8.1 Schematics

To download the schematics, see the design files at TIDEP0054.

8.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDEP0054.

8.2.1 Layout Prints

To download the layer plots, see the design files at TIDEP0054.

8.2.2 Assembly Drawings

To download the assembly drawings, see the design files at TIDEP0054.

9 Software Files

To download the software files, see the design files at sysbiossdk-ind-sitara.

10 References

1. AM335x Sitara™ Processors (AM3359)
2. Texas Instruments, *PHYTER ® Industrial Temperature 10/100Mbs Ethernet Physical Layer Transceiver*, PHYTER® data sheet (TLK110)
3. Texas Instruments, *TPS65910x Integrated Power-Management Unit Top Specification*, TPS65910x data sheet (SWCS046)
4. Texas Instruments, *SYSBIOS Industrial SDK Getting Started Guide*, SYSBIOS guide (SYSBIOS)
5. Texas Instruments, *SYSBIOS Industrial Software Development Kit (SDK) for Sitara™ Processors*, Software-development kit (SYSBIOSSDK-IND-SITARA)
6. Texas Instruments, *ICSS EMAC LLD Developers Guide*, ICSS EMAC guide (ICSS_EMAC_LLD)
7. Texas Instruments, *Network Developer's Kit (NDK) Support Package Ethernet Driver*, Design guide (SPRUFP2)
8. Texas Instruments, *High-availability Seamless Redundancy (HSR) Ethernet for Substation Automation Reference Design*, TI Design (TIDUB08)
9. Texas Instruments, *10/100 Mbps Industrial Ethernet Brick with IEEE 1588 Precision Time Protocol (PTP) Transceiver*, TI Design (TIDUAT6)

11 About the Author

WONSOO KIM is a systems engineer at Texas Instruments, where he is responsible for driving system solutions for Smart Grid applications, defining future requirements in TI product roadmaps, and providing system-level support and training focusing on communication software and systems for Smart Grid customers. He received his Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin in Austin, TX.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265