RH850/U2C Group, U2B-E Group

Ethernet 10BASE-T1S Interface

Summary

This application note explains the connection procedure to the Ethernet PHY-LSI integrated with the RH850/U2C, U2B-E as well as the utilization of the Ethernet 10BASE-T1S Interface (ETNF) module.

This document and the program are intended to support understanding of the features implemented in the RH850/U2C, U2B-E, and are not intended for use in mass production designs.

In addition, it does not reflect the latest manuals, errata, technical updates, or updates to the development environment. When using the relevant functions, please treat this program as a reference and ensure that you proceed at the user's own responsibility with the latest documentation and development environment

Target device

- RH850/U2C Group, U2B-E Group

  Verified Microcontroller Models：R7F702613FABB

Target Integrated Development Environment (IDE).

- IDE：CS+ (Renesas Electronics Corporation) Ver. E8.11.00c3 [10 May 2024] (RH850/U2C)

  Device File：DR7F702606.DVF
- Compiler：Green Hills Compiler (Advanced Data Controls Corporation) V2023.5.4
- Build：GNU make Ver3.81 *Execute in Command Prompt.

BSW to be used

- MCAL：RH850/U2Cx AUTOSAR R22-11 MCAL *Uses Eth driver.

Evaluation Board *This is the operating environment for the sample software.

- Main Board: Y-COMMON-MB-T1-V1 (Renesas Electronics Corporation)
- Piggyback Board: Y-RH850-U2C-292PIN-PB-T1-V1 (Renesas Electronics Corporation)
- Ethernet Extension Board: LAN8680 Ethernet PHY Transceiver (Microchip Technology Inc)
- 10BASE-T1S Ethernet PHY Transceiver Evaluation Board: EVB-LAN8670-USB (Microchip Technology Inc)
- E2 Emulator：RTE0T00020KCE00000R、Conversion Adapter：RTE0T00020KCA00000R

Reference Documents

- RH850/U2C Group User’s Manual (Rev.0.80): R01UH1018EJ0080

This application note has been created with reference to the above manuals.

Details regarding device functions and electrical characteristics are provided in the User's Manual – Hardware Edition.

Contents

1. Overview

This document explains the utilization of the Ethernet 10BASE-T1S Interface (ETNF) module implemented on the RH850/U2C.

1.1 Terms and Acronyms

Table 1-1 Terms and Acronyms (1/2)

Terms and Acronyms	Full Term（description）
AFE	Analog Front-End
AHM	Analog Hard Macro
AN	Auto-Negotiation
BCI	Bulk Current Injection
BER	Bit Error Rate
BT	Bit-Time (100 ns)
BW	Bandwidth
CAN	Controller Area Network
CDR	Clock & Data Recovery
Clause 4	Definition of the basic architecture and operating principles of Ethernet.
Clause 22	Defines the specifications of the MDIO interface.
Clause 45	Defines the extended specifications of the MDIO interface.
Clause 90	Defines specifications for supporting Ethernet time synchronization protocols.
Clause 148	Defines the specifications related to PLCA.
CRS	Carrier Sense
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
DME	Differential Manchester Encoding
DPI	Direct Power Injection
EMC	Electro-Magnetic Compatibility
EMI	Electro-Magnetic Immunity
ENI	Enhanced Noise Immunity
Eth	Abbreviation of Ethernet
ETNF	Ethernet 10BASE-T1S Interface module implemented on the RH850/U2C
FIFO	First-In-First-Out
FSM	Finite State Machine
IFG	Inter-Frame Gap
IP	Intellectual Property
MAC	Media Access Control (Please refer to Clause 4 of [REF05])
MDIO	Management Data Input Output (Please refer to Clause 22 of [REF05])
MII	Media Independent Interface (Please refer to Clause 22 of [REF05])
NRZ	Non-Return to Zero

Table 1-2 Terms and Acronyms (2/2)

Terms and Acronyms	Full Term（description）
OA	Open Alliance
OS	Operating System
PCS	Physical Coding Sublayer
PHY	Physical Layer Entity
PLCA	Physical Layer Collision Avoidance (Please refer to Clause 148 of [REF03])
PMA	Physical Medium Attachment
PoDL	Power over Data Line
RS	Reconciliation Sublayer
RTL	Register Transfer Level
RZ	Return to Zero
SFD	Start of Frame Delimiter (Please refer to Clause 90 of [REF05])
TO	Transmit Opportunity
TSN	Time-Sensitive Networking
TXC	Open Alliance TC14 Transceiver Please refer to Clause 90 of [REF01])
HS IntOSC	High Speed Internal Oscillator
Main OSC	Main Oscillator
PLL	Phase Locked Loop
SSCG	Spread Spectrum Clock Generator
arxml	The file extension of XML format files defined by AUTOSAR.
GHS	Green Hills

1.2 Reference

Table 1-3 Reference document

Reference	Document Name	Document Number
[REF01]	Open Alliance SIG, "10BASE-T1S Transceiver Interface Rev 1.5," 2021.	-
[REF02]	RH850/U2C Group User's Manual (Rev.0.80):	R01UH1018EJ0080
[REF03]	IEEE Computer Society, "IEEE Std 802.3cg™-2019, Amendment 5: Physical Layer Specifications and Management Parameters for 10 Mb/s Operation and Associated Power Delivery over a Single Balanced Pair of Conductors," New York, 2019.	-
[REF04]	Open Alliance SIG, Advanced diagnostic features for 10BASE-T1S automotive Ethernet PHYs, 2021.	-
[REF05]	IEEE Computer Society, "IEEE Standard for Ethernet," New York, 2018.	-
[REF06]	Open Alliance SIG, "10BASE-T1S PLCA Management Registers Rev 1.1," 2019.	-
[REF07]	RH850/U2Cx AUTOSAR R22-11 MCAL User's Manual (Rev.0.40)	R20UT5443EJ0040
[REF08]	Common Main Board for RH850 and R-Car U5x Y-COMMON-MB-T1-V1 Y-COMMON-MB-T1-V1-JP	R20UT5305ED0200
[REF09]	RH850/U2C 292pin User's Manual: Piggyback Board Y-RH850-U2C-292PIN-PB-T1-V1	R20UT5390ED0202

1.3 Basic information of 10BASE-T1S

10BASE-T1S is a Single Pair Ethernet (SPE) technology based on the IEEE 802.3cg standard which is designed to enable efficient and low-cost communication in industrial and automotive networks.

This technology leverages the advantages of the conventional Ethernet while being optimized for specific applications, particularly those requiring reliable communication at lower data rates.

Technical specifications

- Communication speed and cable configuration：
  - The maximum communication speed is 10Mbps, which is slow but provides reliable communication.
  - The use of a single twisted pair cable enables simplified wiring and cost reduction.
- Multi-drop connection:
  - In addition to point-to-point connections, it supports multi-drop configurations with up to 8 nodes.
  - To reduce wiring costs, a bus topology is adopted instead of a star topology.
- PLCA（Physical Layer Collision Avoidance）:
  - Equipped with a collision avoidance function to enable efficient half-duplex communication.
  - Data collisions are prevented by sequentially assigning transmission opportunities to each node.
- Low power consumption:
  - Power-efficient design suitable for IoT and automotive environments.
- Compatibility:
  - Ethernet MAC and protocol stack can be used as is.
  - Utilize conventional Ethernet assets to improve design efficiency.
- Supports PoDL (Power over Data Line).:
  - In point-to-point configurations, power supply is provided through signal lines, which achieves lightweight wiring and a space-saving design.

1.4 Functional Overview of the ETNF Module Integrated in the RH850/U2C

The ETNF module integrated in the RH850/U2C supports RMII (full-duplex mode) and T1S (half-duplex mode) interfaces along with an Ethernet AVB controller. In addition, the built-in IP core is equipped with components of the 10BASE-T1S Ethernet physical layer, including the PMA, PCS, and PLCA.

This application note focuses specifically on the functionality of the ETNF module's 10BASE-T1S interface in half-duplex mode.

Functions Supporting 10BASE-T1S in ETNF.

- Protocol
  - Flow Control Conforming to IEEE 802.3x Standard.
- Data Transfer Rate
  - 10 [Mbps]
- Transfer Mode
  - T1S Mode (half-duplex mode)
- Functions
  - Compliant with the OATC14 10BASE-T1S transceiver interface.
  - Compliant with the OATC14 10BASE-T1S PLCA Management Register.
  - Compliant with the IEEE 802.3cg-2019 standard.
  - Descriptor Management System
- Transmit/Receive FIFO
  - When transmitting: 16 [KB]
  - When receiving: 8 [KB]
- Communication Interfaces
  - Open Alliance 10BASE-T1S transceiver interface

Ethernet AVB controller part of ETNF

- AVB-DMAC（DMA Transfer Controller）
  - Role：Handling data transfers between the data storage area in URAM and the receive/transmit FIFO buffers using the Direct Memory Access (DMA) functionality.
  - Operation：Direct read and write operations from the FIFO buffers are not permitted. Instead, data storage addresses for transmission and reception are managed using information known as descriptors. By organizing multiple descriptors in a descriptor list, it is possible to continuously transmit and receive multiple Ethernet frames.

- E-MAC（MAC Controller）
  - Role：Handles data transfers between the receive/transmit FIFO buffers and the T1S transceiver.
  - Operation：Supports the T1S transceiver and provides an interface to the PHY. Constructs Ethernet frames based on the data in the transmit FIFO and transmits them to the T1S transceiver.
    Performs CRC checks on frames received from the T1S transceiver and writes valid frames to the receive FIFO.

Ethernet PHY part of the ETNF

- Ethernet physical layer conforming to IEEE 802.3cg™ 10BASE-T1S
  - PMA（Physical Medium Attachment）
  - PCS（Physical Coding Sublayer）
  - PLCA（PHY-Level Collision Avoidance）Adjustment Sublayer
- Coordinates with the MAC
  - Uses the MII (Media Independent Interface) to operate with a standard IEEE CSMA/CD Ethernet MAC.
  - The integration of the PLCA RS enables MAC devices without PLCA MII extensions to utilize PLCA functions.
- Transceiver connection
  - The PMA connects to a standard OPEN Alliance 10BASE-T1S transceiver.
- Management Function
  - Equipped with an optional register file.
  - Provides management functions via Clause 22/Clause 45 registers.
  - Connects MDIO（Management Data Input/Output）slave modules to the PHY top-level module.

2. Functional overview of 10BASE-T1S and Software Development

This chapter provides a technical overview of the 10BASE-T1S interface and explains the software development environment and setup methods required to utilize its functionality.

2.1 Functional Description

The 10BASE-T1S MAC layer belongs to the data link layer and supports both point-to-point connections and multidrop connections of up to 8 nodes. By adopting PLCA (Physical Layer Collision Avoidance), it manages transmission opportunities instead of CSMA/CD, and enabling efficient communication while avoiding collisions. The MAC layer is responsible for Ethernet frame transmission and reception management, flow control, and error detection, and is suitable for applications that require real-time performance. Additionally, by connecting to the PHY layer via MDIO, it is possible to integrate with existing Ethernet infrastructure, but it is necessary to use an appropriate PHY.

In this application note, a three-pin interface is employed as the data link between the MAC layer and the PHY, sharing MDIO serial communication with Ethernet transmit and receive frame signaling (*1).

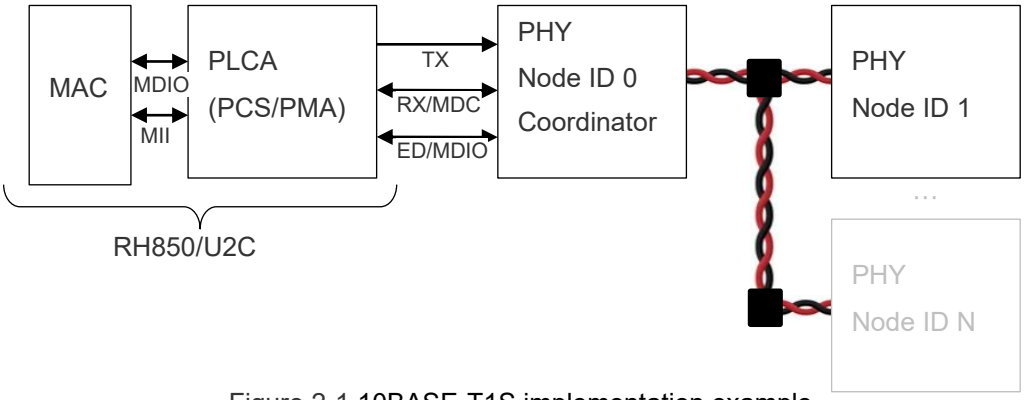


Figure 2-1 10BASE-T1S implementation example

Table 2-1 Functional Block Description

Functional Blocks	Description
MAC Layer	It manages Ethernet data transfers and is responsible for constructing data frames and analyzing sent and received frames.
PLCA Control Modules	Schedules transmission opportunities between nodes to ensure efficient data transfer. The operation of the node ID is managed by the beacon signal.
PHY Layer	Communication is achieved using electrical signals. It uses a single twisted pair cable that supports 10Mbps communication speeds and uses PLCA to avoid collisions.

(*1) In the ETNF module, a five-pin configuration that separates MDIO and Ethernet communication is also supported as an alternative to the three-pin interface.

2.2 Development Environment Setup

To implement 10BASE-T1S, it is necessary to properly configure and integrate the various elements of hardware preparation, network settings, and software configuration to build an operating environment. Completing these preparations will ensure stable operation. The following provides a detailed procedure for setting up the development environment.

- Hardware preparation
  - Set the jumpers on each board: Main Board, Piggyback Board, and Ethernet Extension Board. This sample software assumes a Main OSC oscillation frequency of 20MHz.

Table 2-2 Jumper settings for each board (example)

Main Board		
Jumper Name	Settings	Functions
JP1	3-2 connection	UART0 (pins 2-1) or LIN0 (pins 3-2) interface. Note: Pin 1 is on the right side.
Piggyback Board		
Jumper Name	Settings	Functions
JP1 GETH0BVCC	OPEN	Power supply for SGMII.
JP1 GETH0PVCC	OPEN	Power supply for SGMII.
JP1 (other)	-	Set to 3.3V side, 5V side, or OPEN.
JP2	OPEN	Core voltage selection.
JP3	CLOSED	Device selection ・JP3[OPEN]: U2C8 ・JP3[CLOSED]: U2C4
JP4	OPEN	Pull down port P00_7 to GND.
JP6	2-3 connection	Select the signal source for the TRST# signal. ・JP6[1-2]: Fix the TRST# signal to E0VCC. ・JP6[2-3]: The TRST# signal is the TRST_TOOL# signal from the E2 debug connector (pin 3 of connector CN9).
JP7	OPEN	Change the FLMD0 signal to “H”.
JP8	OPEN	Change the FLMD1 signal to "GND"
JP9	2-5 connection	Select the +3.3V power source. ・JP9[1-4]: +3.3V is obtained from the onboard voltage regulator. ・JP9[2-5]: +3.3V is obtained from the main onboard. ・JP9[3-6]: +3.3V is obtained from the external power supply through CN7.
JP10	CLOSED	Current measurement bridge +3.3V
JP11	CLOSED	Enable the +5.0V power supply from the main board.
JP12	CLOSED	Current measurement bridge +5.0V
JP13	OPEN	EVTO0#: Event trigger output for the debugger
JP14	OPEN	Select the pull-up voltage for the pull-up/pull-down connector CN12. ・JP14[1-2]: Pull up pins 1/3/5/7 to 5.0V. ・JP14[2-3]: Pull up pins 1/3/5/7 to 3.3V.
JP15	OPEN	Select the pull-up voltage for the pull-up/pull-down connector CN12. ・JP15[1-2]: Pull up pins 9/11/13/15 to 5.0V ・JP15[2-3]: Pull up pins 9/11/13/15 to 3.3V
JP17	OPEN	Connect and select port P20_0 to the main board. ・JP17[OPEN]：Connect P20_0 to B1_P20_0 (ETH0RXD0). ・JP17[CLOSED]：Connect P20_0 to B2_P20_0 (I2SMCLK).
JP18	OPEN	Selection of the Ethernet1 MDIO/MDC Port ・JP18[OPEN]: ETH1_MDIO = P21_0, ETH1_MDC = P20_11 ・JP18[CLOSED]: ETH1_MDIO = P04_9, ETH1_MDC = P04_8
JP19	OPEN	Jumpers for PCB production test
JP20	OPEN	Enable LED output
Ethernet Extension Board		
Jumper Name	Settings	Functions
Termination Resistor Jumper	CLOSED	Enable the termination resistor (near the twisted-pair cable connector)

- Connect the Main Board, Piggyback Board and Ethernet Extension Board. When making connections, ensure that the connectors are fully inserted to prevent poor contact. When connecting the twisted pair cable to the screw terminal block, make sure to connect properly to prevent short circuits caused by the core wire protruding or insufficient tightening. In particular, when connecting two twisted pair cables to a single screw terminal block, make sure to confirm that the terminals are securely fixed to prevent poor contact or signal degradation caused by insufficient tightening or cable misalignment. *When the number of nodes is 3 or more (e.g. 2 sets of boards + monitor)

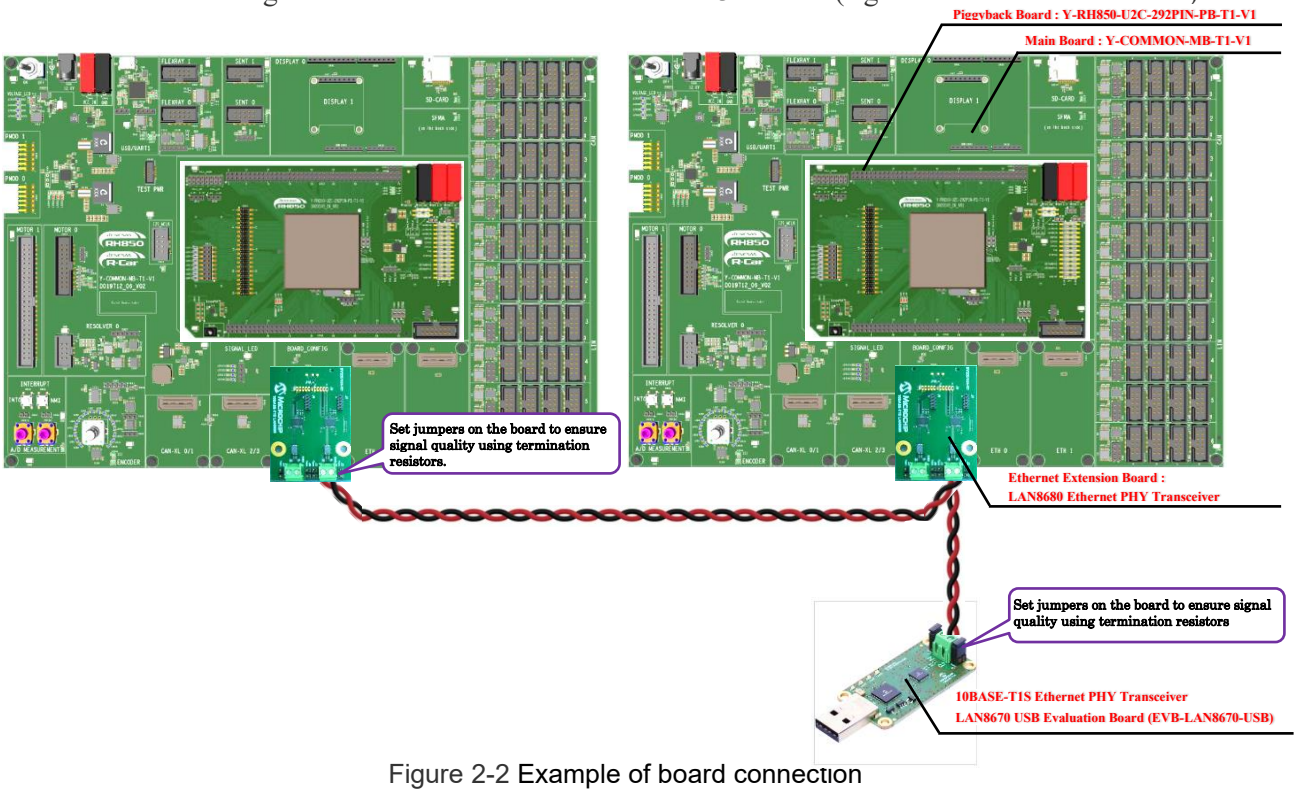


Figure 2-2 Example of board connection

- Connect the E2 emulator to the Piggyback Board via a conversion adapter (set SW1 on the conversion adapter to the '1' side). After connecting the USB cable for board configuration to the Main Board, supply 12V power (using a power adapter or stabilized power supply). It has been confirmed that this sample software operates normally with the default settings (Set default button) in the board configuration tool.

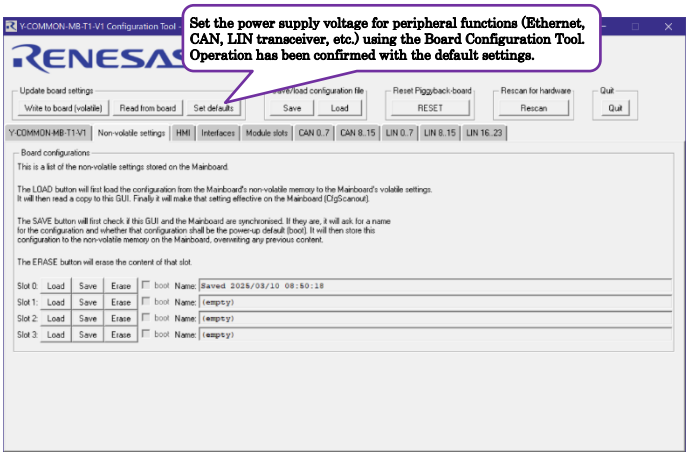


Figure 2-3 Board configuration tool

- The EVB-LAN8670-USB 10BASE-T1S Ethernet PHY Transceiver Evaluation Board is used to monitor Ethernet communications. On a Windows PC, it operates as a network adapter to connect the USB 2.0 interface to a 10BASE-T1S network interface. Additionally, adjust the physical layer parameters in the advanced settings. (Refer to the example in the following figure)

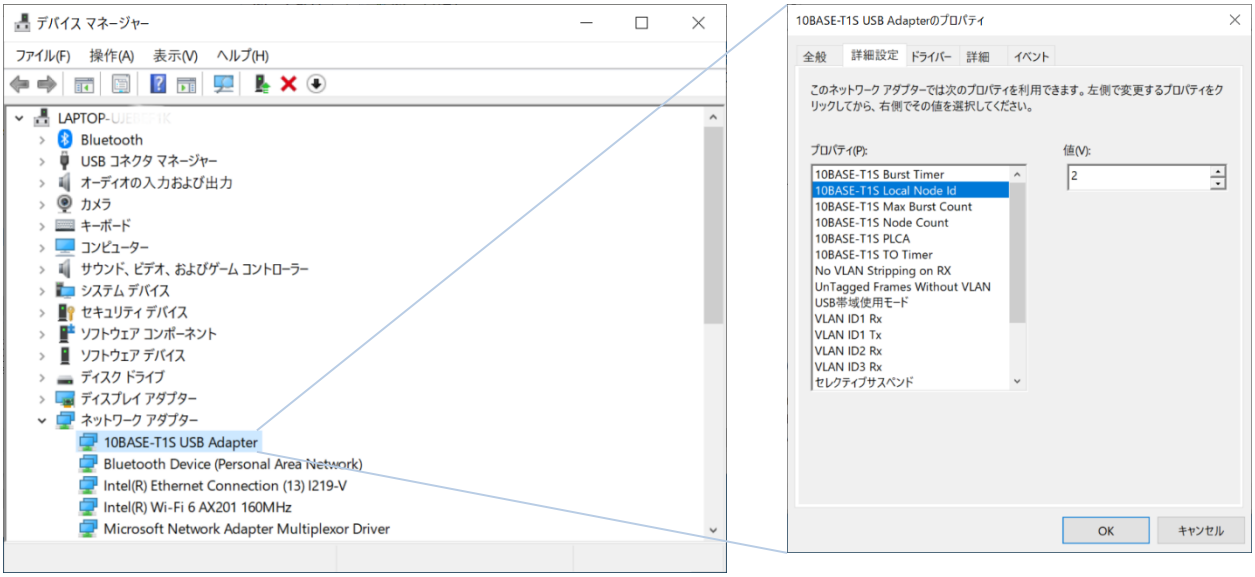


Figure 2-4 Network Adapters

The following table shows a list of the "Detailed Settings" properties for the 10BASE-T1S USB Adapter.

Table 2-3 10BASE-T1S USB Adapter Properties (Example)

Property	Value
10BASE-T1S Burst Timer	128
10BASE-T1S Local Node id	2
10BASE-T1S Max Burst Count	0
10BASE-T1S Node Count	8
10BASE-T1S PLCA	Enabled
10BASE-T1S TO Timer	32
No VLAN Stripping on RX	Enabled
UnTagged Frames Without VLAN	Enabled
USB Bandwidth Usage Mode	Auto
VLAN ID1 Rx	0
VLAN ID1 Tx	0
VLAN ID2 Rx	0
VLAN ID3 Rx	0
Selective Suspend	Disabled
Selective Suspend Idle Timeout	10
Network Address	Unavailable
Flow Control	Disabled
Speed and Duplex Mode (Half/Full)	10Mbps/Half Duplex
Priority VLAN	Priority VLAN Disabled

- Preparing the build environment
  - ➢ Install the integrated development environment and the C compiler. For details about the CS+ integrated development environment and the GHS compiler, please refer to the corresponding manuals.
  - ➢ Please prepare MCAL. This sample software uses the Ethenat (Eth) driver.
  - ➢ GNU Make 3.81 is used for building. Please obtain it from the download (official distribution) site and install it.
  - ➢ Set the environment variable (GNU MAKE) in the advanced system settings of your Windows PC. Specify the installation folder for the Make command.

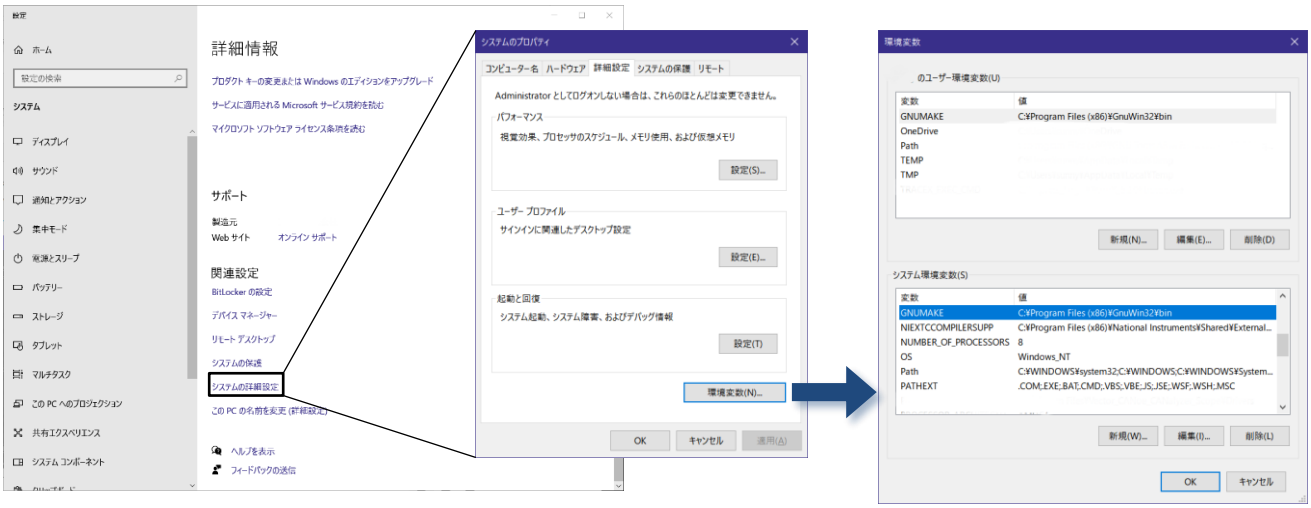


Figure 2-5 Setting the Environment Variable for GNU Make

  - ➢ As an environment variable setting, the GNU Make path settings are specified in the build batch file (T1S_SampleApp.bat).

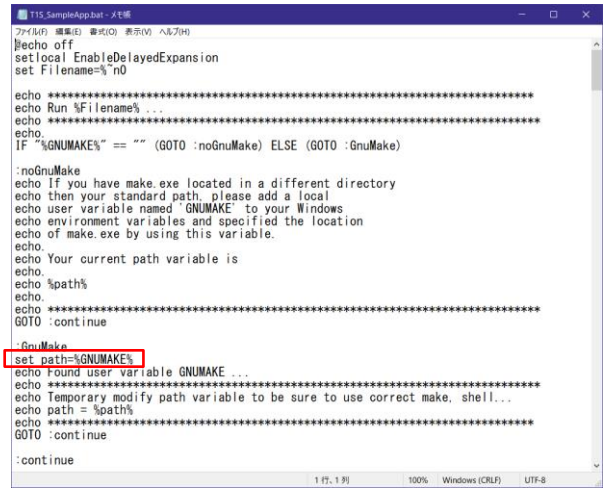


Figure 2-6 Setting the Path for GNU Make

- Procedure for verification of sample software operation.

  ➢ Execute ***command.bat*** to launch the command prompt. To build this sample software, navigate to the target project in the command prompt, then execute the build batch file ***T1S_SampleApp.bat***.

```
>cd node_id0[Enter]
>T1S_SampleApp Eth 22_11 702613FABB No No unset[Enter]
```

*For the Node ID 0 project

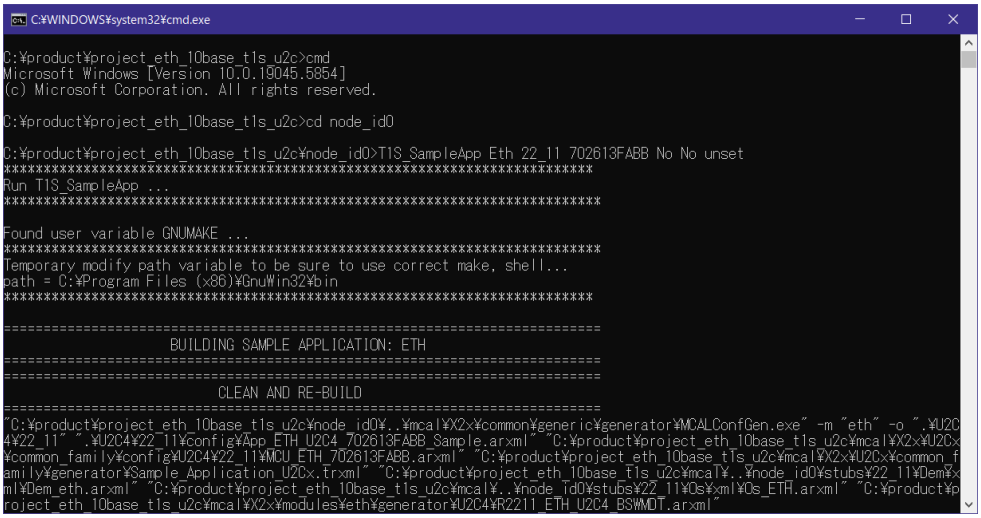
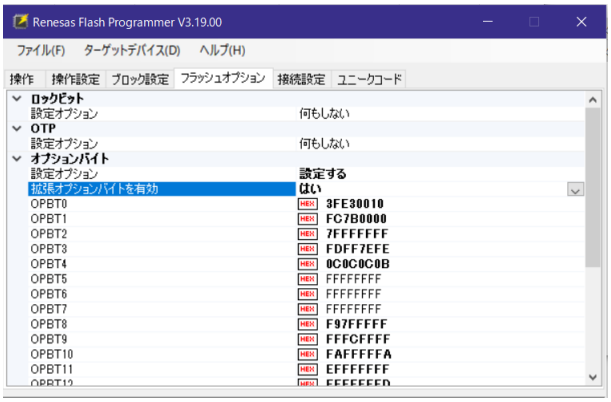


Figure 2-7 Example of build execution

The arguments of the batch file are the same as those of the MCAL sample software, please refer to the MCAL manual for details. Additionally, when executing the GHS compiler, please make sure that the dongle and license file are correctly recognized. For the file configuration of this sample software, please refer to Table 4-1.

➢ Verify the option bytes using the E2 emulator. When using the Renesas Flash Programmer, set the FLMD1 pin level of the Piggyback Board to JP8[CLOSED]: FLMD1 = GND. Additionally, if you intend to use P06_7 (LIN0RX) after setting the option byte, change the setting back to JP8 [OPEN]: FLMD1 = OPEN.

OPBT8 の設定例：0xF97F FFFF、bit30='1': T1S
OPBT10 の設定例：0xFAFF FFFA、bit28='1'、bit[26:24]='010': 20MHz
OPBT11 の設定例：0xEFFF FFFF、bit[31:29]='11x': 320MHz、bit28='0': enabled
OPBT12 の設定例：0xFFFF FFFD、bit[1:0]='01': Single Map Mode



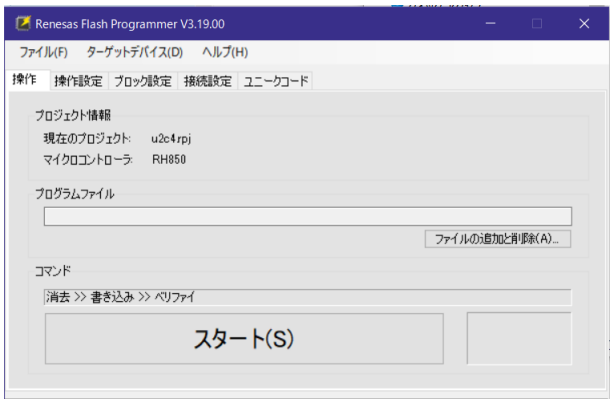


Figure 2-8 When the Flash Options tab is not displayed

If the Flash Option tab does not appear in the Renesas Flash Programmer, preventing direct verification or setting of the extended option bytes, you can temporarily convert the target area into a Motorola S-record format file to perform the verification and setting. When modifying values in the Motorola S-record file, please pay attention to byte endianness and checksum values.

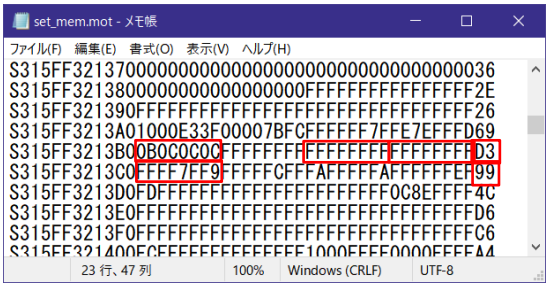


Figure 2-9 Example of option byte setting

- Start two CS+ that support RH850/U2C and open the sample software projects in each (Project for node ID 0: ***run_node_id0.mtpj***, or project for node ID 1: ***run_node_id1.mtpj***).
  Please note that double-clicking a project file may accidentally launch an incompatible version of CS+ for RH850/U2C.

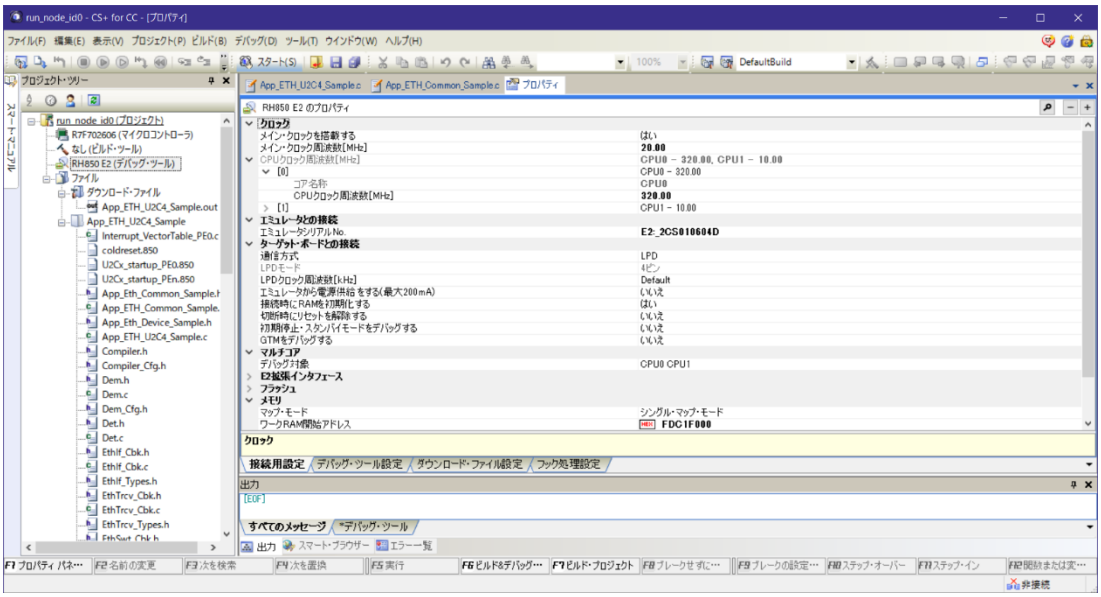


Figure2-10 CS+Startup

➢ Download and execute each of the load module files generated by the build as shown below.

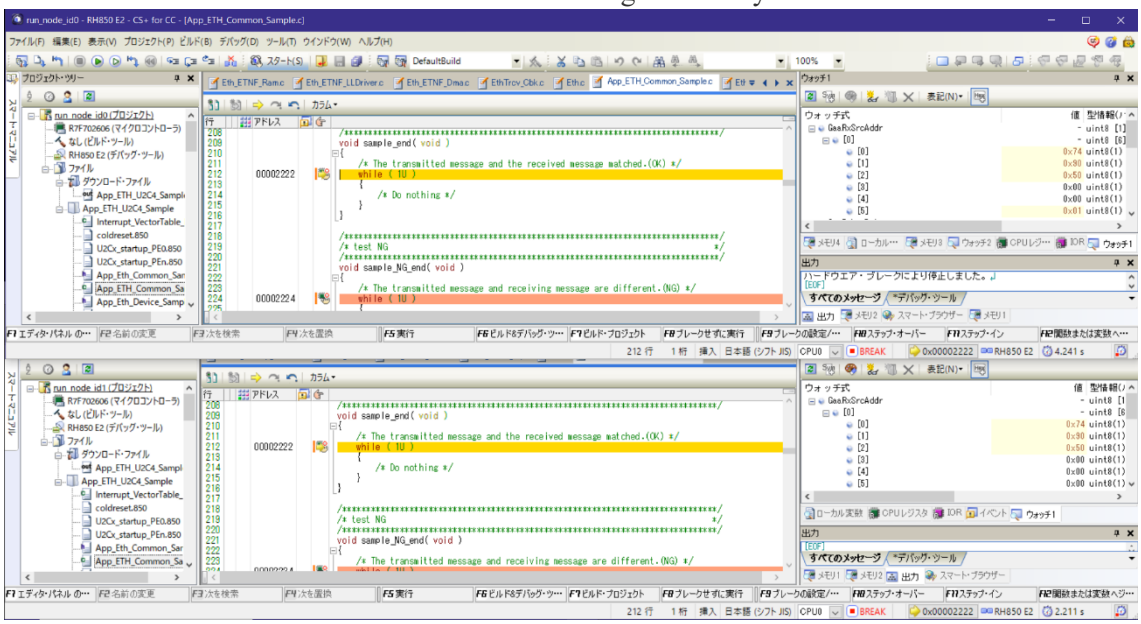


Figure 2-11 Running the sample software (when two boards are set)

- Verification of 10BASE-T1S Network Communication

➢ Use Wireshark to capture and analyze Ethernet communication packets between the two boards via the EVB-LAN8670-USB.

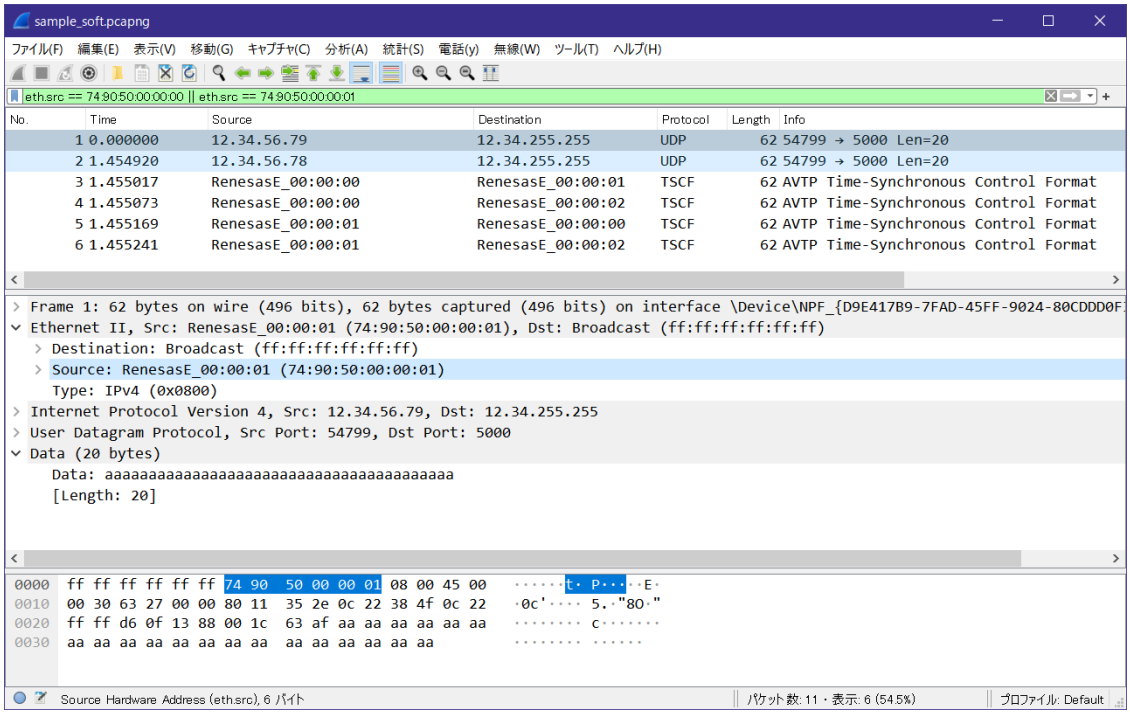


Figure 2-12 Monitoring Ethernet communications
(example of capturing communications between two sets of boards)

3. Usage of 10BASE-T1S

The following section explains the utilization of the ETNF module using the MCAL Eth driver for software processing to support 10BASE-T1S. This application note omits a detailed explanation of the 10BASE-T1S communications protocol (Ethernet frame structure, MAC address management) and focuses on the utilization of software processing.

The following table lists the configuration parameters of the Ethernet transceiver. This table provides an overview of each setting item and enables appropriate parameter selection.

Table 3-1 Configuration parameters for the Ethernet transceiver

Definition Name	Description
T1S_PHY_ADDR	Set the transceiver PHY address.
ETH_PHY_ADDR	Set the internal PHY address on the microcontroller.
T1S_PHY_CONTROL_CFG	Set the CONTROL register.
T1S_PHY_T1SPMACTRL_CFG	Set the T1SPMACTRL register.
T1S_PHY_T1SPMATSTM_CFG	Set the T1SPMATSTM register.
T1S_PHY_T1SPCSCTRL_CFG	Set the T1SPCSCTRL register.
T1S_PHY_T1STWEAKS_CFG	Set the T1STWEAKS register.
T1S_PHY_T1SPLCAEXT_CFG	Set the T1SPLCAEXT register.
T1S_PHY_T1SPLCAEXT_CFG	Set the T1SPLCAEXT register.
T1S_PHY_T1SPMATUNE0_CFG	Set the T1SPMATUNE0 register.
T1S_PHY_T1SPMATUNE1_CFG	Set the T1SPMATUNE1 register.
T1S_PHY_PLCACTRL0_CFG	Set the PLCACTRL0 register.
T1S_PHY_PLCACTRL1_CFG	Set the PLCACTRL1 register.
T1S_PHY_PLCATOTMR_CFG	Set the PLCATOTMR register.
T1S_PHY_PLCABURST_CFG	Set the PLCABURST register.
Bit Definition Name	Description
T1S_PHY_ADDR1_REG18H_SET_CFG	Set the set bit value of the XCVR_ANALOG_SET_2.
T1S_PHY_ADDR1_REG18H_CLR_CFG	Set the clear bit value of the XCVR_ANALOG_SET_2.
T1S_PHY_ADDR1_REG1FH_SET_CFG	Set the set bit value of the XCVR_ANALOG_SET_9.
T1S_PHY_ADDR1_REG1FH_CLR_CFG	Set the clear bit value of the XCVR_ANALOG_SET_9.
T1S_PHY_REG000000_RESET	Set the Soft Reset.
T1S_PHY_REG000000_LOOP_OFF	Set Loopback to disabled.
T1S_PHY_REG000000_LCTL_ON	Set PHY link control to enabled.
T1S_PHY_REG000000_LPWR_OFF	Set entering Low Power Mode to disabled.
T1S_PHY_REG000000_ISOM_OFF	Set entering Isolation Mode to disabled.
T1S_PHY_REG000000_CTEST_OFF	Set Collision Execution Mode to disabled.
T1S_PHY_REG2108F9_LPWR_OFF	Set Low Power Mode to disabled.
T1S_PHY_REG2108F9_LOOP_OFF	Set Loopback Mode to disabled.
T1S_PHY_REG2108FB_NORMAL	Set to Normal operation.
T1S_PHY_REG2308F3_LOOP_OFF	Set Loopback Mode to disabled.
T1S_PHY_REG3F8001_PKTLOOP_OFF	Set to not reflect TX frames to MII RX.
T1S_PHY_REG3F8001_ENIE_OFF	Set Extended Noise Resistance Mode to disabled.
T1S_PHY_REG3F8001_UNJT_OFF	Set Auto recovery to disabled.
T1S_PHY_REG3F8001_RXNRZ_OFF	Set the AFE RX signal to be considered as RZ encoded.
T1S_PHY_REG3F8001_SCRD_OFF	Set PCS scrambling and descrambling functions to disabled.
T1S_PHY_REG3F8001_NCOLM_ON	Collision detection is set to unmasked.
T1S_PHY_REG3F8001_RXDLY_ON	Set the MII RX signal to be delayed to avoid simultaneous communication.
T1S_PHY_REG3F8002_PREN_OFF	Set PLCA RS to operate in standard mode.
T1S_PHY_REG3F8002_LDEN_OFF	Set the PLCA reader to be selected by node ID.
T1S_PHY_REG3F8002_LDR_OFF	If LDEN=1, set as PLCA slave.
T1S_PHY_REG3F8003_NNTHR	Set the Threshold value for the NN comparator.
T1S_PHY_REG3F8003_DRFTW	Set the Time Window for the Drift Compensator.
T1S_PHY_REG3F8004_JJHHTHR	Set the Threshold value for the JJHH comparator.
T1S_PHY_REG3F8004_JJTHR	Set the Threshold value for the JJ comparator.
T1S_PHY_REG3FCA01_EN_ON	Set PLCA RS function to enabled.
T1S_PHY_REG3FCA02_NCNT	Set the total number of nodes NCNT.
T1S_PHY_REG3FCA02_ID	Set the node ID.
T1S_PHY_REG3FCA04_TOTMR	Set the PLCA Transmit Opportunity Timer.
T1S_PHY_REG3FCA05_MAXBC	Set the number of additional packets that can be transmitted within the transmit opportunity.
T1S_PHY_REG3FCA05_BTMR	Set the waiting time until connection to the burst.

Please refer to Chapter 4 for the parameter settings of this sample software. For detailed configuration parameters of the MCAL (Eth driver), refer to the manual ([REF07]).

This chapter describes the practical utilization of software processing, including an overview of the target Eth driver's functions and the procedures involving the Ethernet transceiver.

Prerequisite

- Ethernet communication is controlled using the AUTOSAR MCAL Ethernet driver.

Overview of the Eth driver

- Basic Specifications
  - Complies with AUTOSAR R22_11.
  - Modules such as Det, Dem, and EthIf are implemented as stubs (refer to the Table 4-1: File Structure [excerpt]).
    - Make any necessary adjustments or modifications to the stub-implemented modules according to the operational requirements.
- Thread management
  - Multithreading is not supported
    - Usable only in single-threaded as specified by AUTOSAR.
- Initialization management
  - MCU-specific initialization is not implemented in the Eth driver.
    - In this sample software, this is implemented as processing other than the Eth driver.
  - APIs that can be used before initialization.
    - Only `Eth_GetVersionInfo` and `Eth_MainFunction` can be called before `Eth_Init`.

Network Management for Eth Drivers

- Address Filtering
  - Eth_UpdatePhysAddrFilter API
    - It can be called when the controller is in active or down state.
    - The broadcast address (`FF:FF:FF:FF:FF:FF`) is only enabled in the down state.
    - Promiscuous mode
      - Specify `FF:FF:FF:FF:FF:FF` to enable.
      - Specify `00:00:00:00:00:00` to disable.
      - While in promiscuous mode, all multicast addresses are ignored.
    - Address registration restrictions
      - Up to 32 multicast addresses can be registered.

- Unicast addresses cannot be registered
- Passing `00:00:00:00:00:00` will delete all registered multicast addresses.

➢ Eth_SetPhysAddr API

✧ Only unicast addresses can be specified.

✧ Multicast, Broadcast are not allowed.

Management of Eth Driver Data Transmission and Reception.

- Receive Buffer Management

➢ EthIf_RxIndication API

✧ The `LenByte` parameter sets the payload size excluding the Ethernet frame header.

✧ The receive buffer is released immediately after processing and will be overwritten after one buffering cycle.

- Transmit buffer management

➢ Eth_Transmit API

✧ The transmit buffer is released after transmission is complete.

✧ If internal resources are insufficient, `E_NOT_OK` is returned, requiring the application to retry.

✧ Polling mode

- The transmit buffer is released by `Eth_TxConfirmation`.
- Even if `EthIf_TxConfirmation` is not required, it is necessary to call `Eth_TxConfirmation` periodically.

Management of Eth Driver Controllers.

- Management of controller states.

➢ Eth_SetControllerMode API

✧ If `ETH_MODE_DOWN` is passed, the function will block until the ongoing transmission is completed.

✧ The maximum block time can be set with `EthTimeout`.

➢ Active down state of the controller.

✧ The functions `Eth_ProvideTxBuffer`, `Eth_Transmit`, `Eth_Receive`, `Eth_TxConfirmation` are only available in the active state.

✧ `Eth_SetPhysAddr` is only enabled in the down state.

Eth driver QoS and Priority Control

- Management of Transmit queue

➢ AVB（Audio Video Bridging）

✧ Up to 4 transmit queues can be configured.

➢ Priority setting for transmit queues.

- ✧ Controlled via the EthTxQueueIdx parameter.
  - ● `0: `Best effort
  - ● `1: `Network Control of Precision Time Protocol
  - ● `2: `BS Algorithm (Class B)
  - ● `3: `CBS Algorithm (Class A)

➢ QoS Support

- ✧ If `EthQosSupport` is disabled, all transmit queues are treated as best effort.
- ✧ Applies only to AVB standard.

Bandwidth management for Eth driver

- ● CBS (Credit Based Shaper)
  - ➢ Setting the `EthTxQueuePolicy` parameter to `ETH_CBS` enables the CBS algorithm.
  - ➢ The `EthCtrlTxQueueBwFraction` parameter allows configuration of the bandwidth allocation ratio for transmit queue.
  - ➢ Enabling CBS will contribute to resolving latency issues

Eth driver Control Frame

- ● Broadcast/Multicast Storm Detection.
  - ➢ Set the broadcast storm detection threshold using `EthBroadcastStormThreshold`.
  - ➢ Set the multicast storm detection threshold using `EthMulticastStormThreshold`.
- ● Flow control
  - ➢ EthPauseFrame Parameter
    - ✧ When set to true, transmit and receive flow control is enabled.
    - ✧ Set the pause time with `EthPauseTime`.
    - ✧ Set the Retransmission Time with `EthPauseFrameRetransmissionTime`

Timestamp management for Eth driver.

- ● Timestamp of the received frame
  - ➢ The timestamp of a received frame can be obtained using `Eth_GetIngressTimeStamp`.
  - ➢ When `EthBeTimeStampStore` is set to true, timestamp information can be added to the best-effort receive descriptors.
  - ➢ When `EthStreamTimeStampStore` is set to true, timestamp information can be added to the stream receive descriptors.

3.1 Initialization process

The following flowchart explains the initialization procedure using the ETNF module's Ethernet MAC function and PHY interface for 10BASE-T1S communication. Furthermore, the flowchart assumes the use of the AUTOSAR MCAL Eth driver.

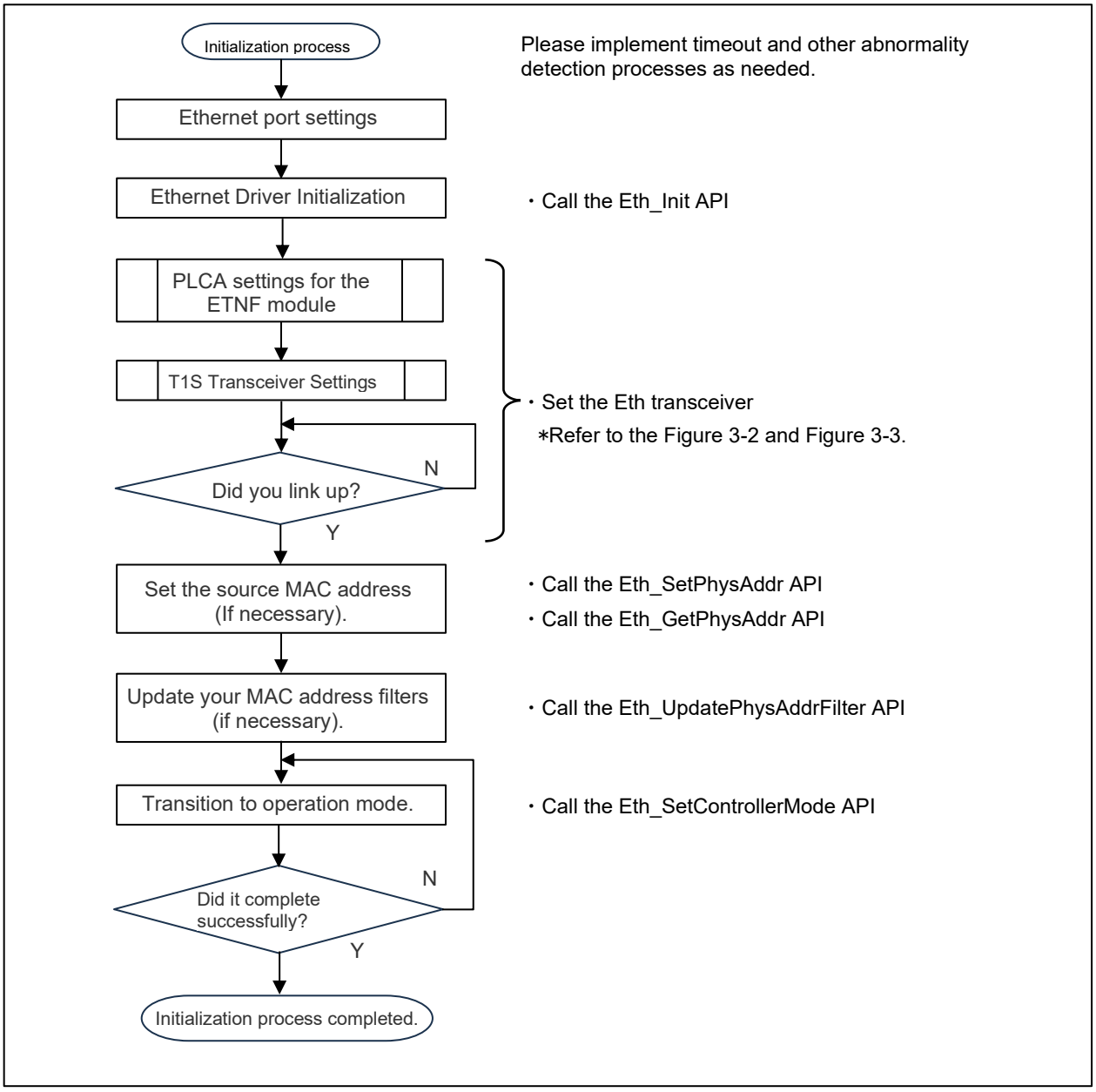


Figure 3-1 Example of T1S initialization process

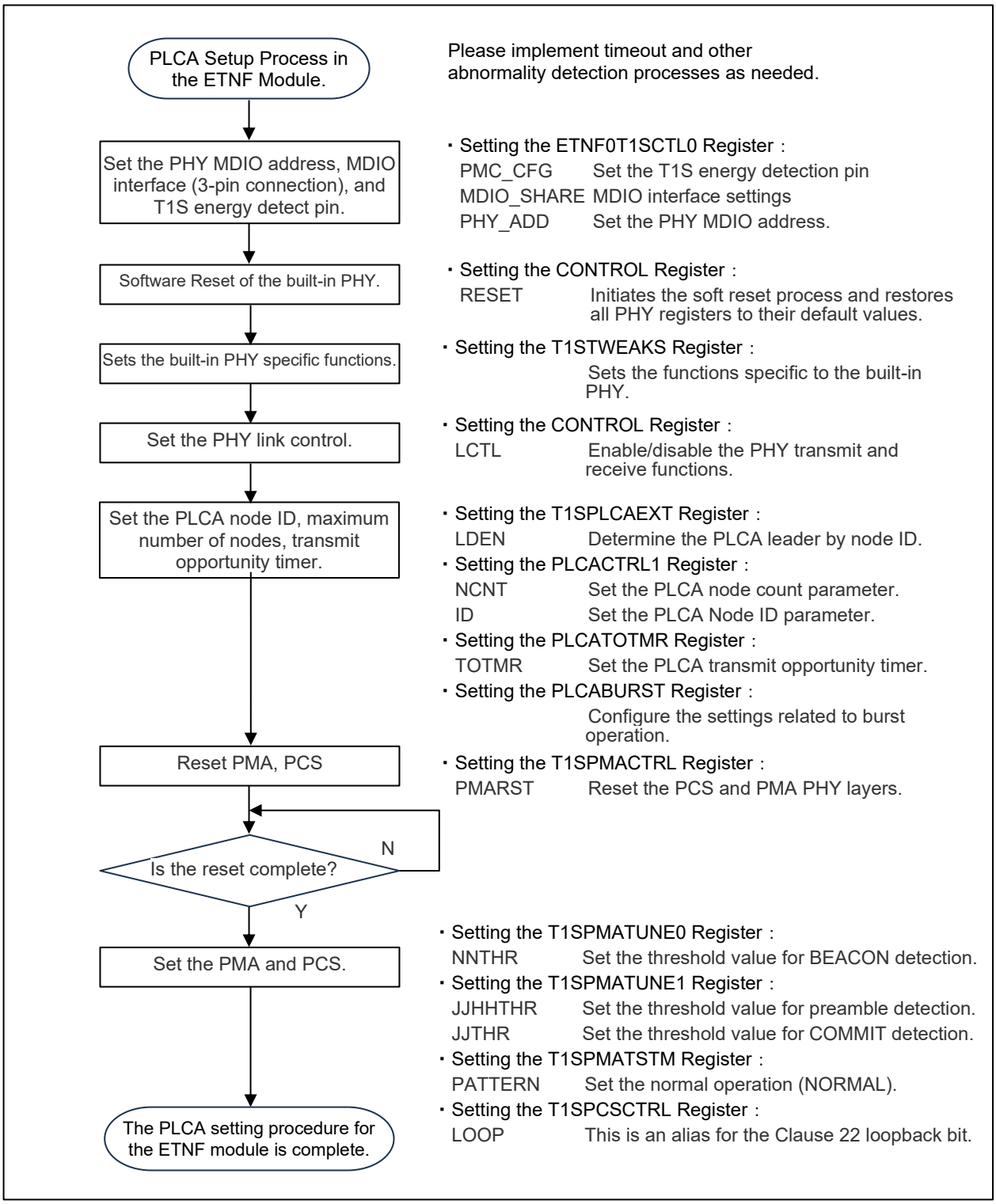


Figure 3-2 Example of PLCA setting process flow

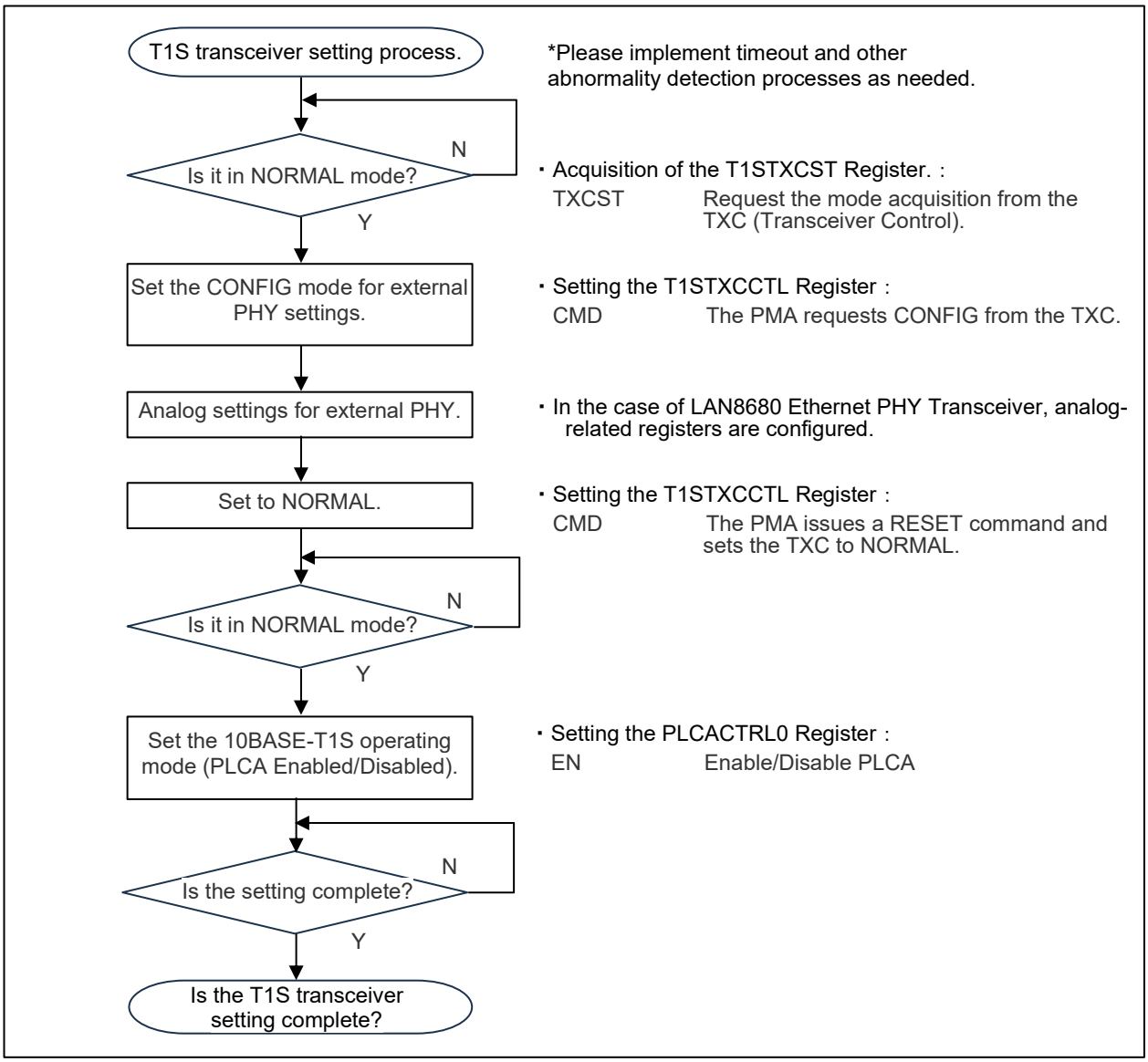


Figure 3-3 Example of T1S transceiver setting processing flow

3.2 Data Transmission process

The following flowchart explains the 10BASE-T1S data transmission processing, assuming the use of the AUTOSAR MCAL Eth driver.

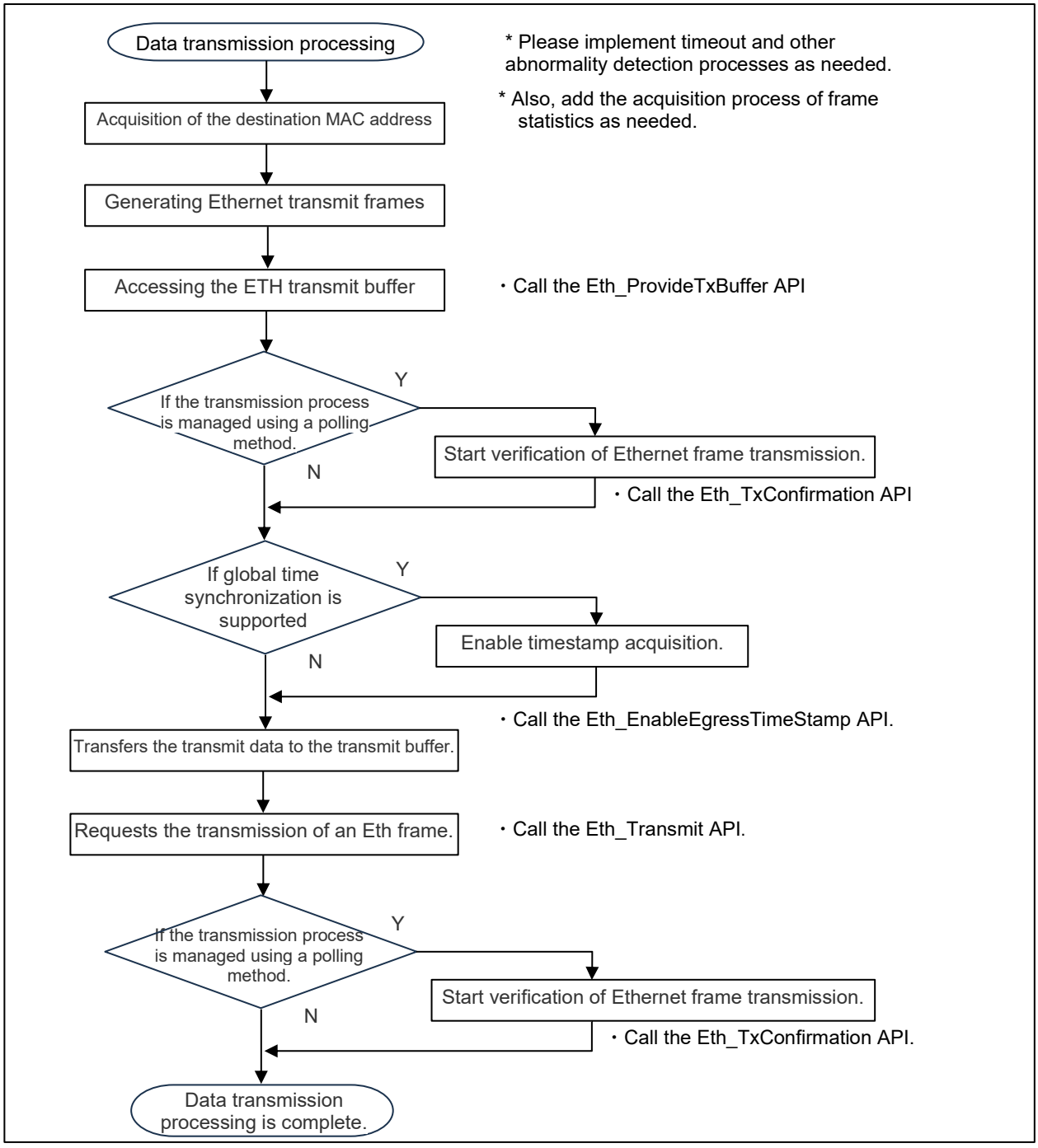


Figure 3-4 Example of data transmission processing flow

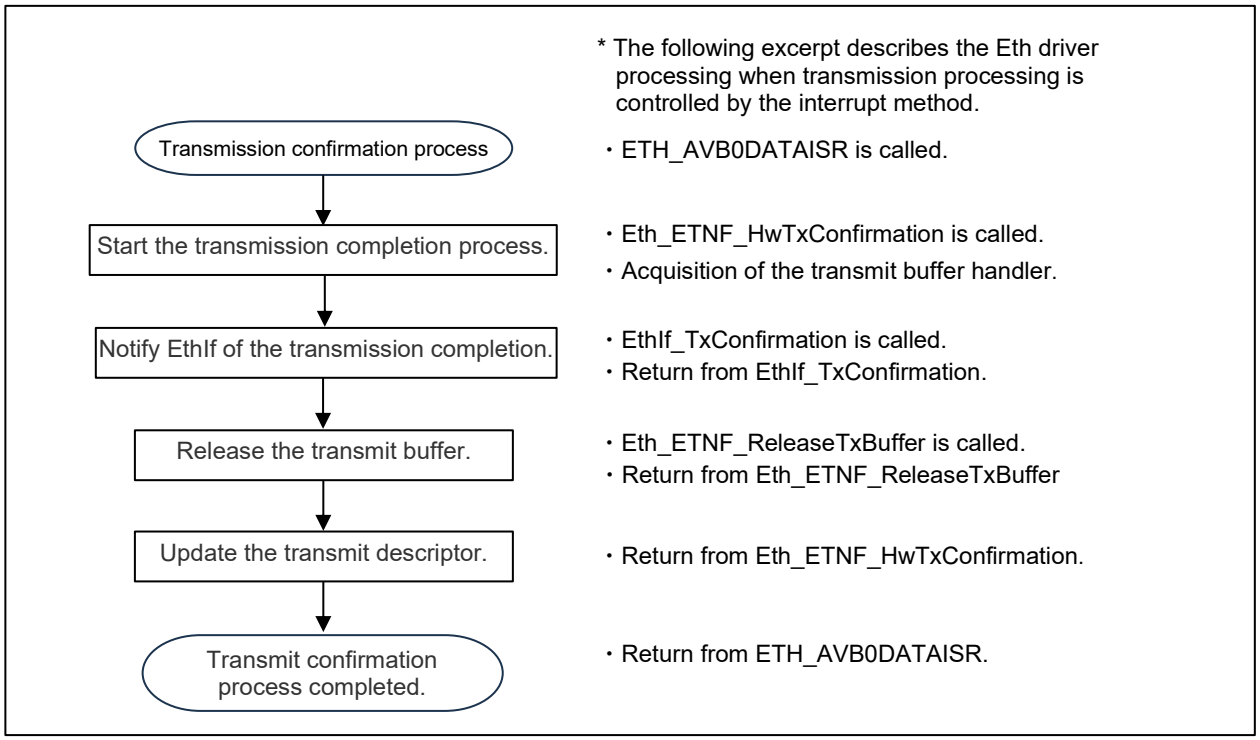


Figure 3-5 Transmission interrupt processing flow

3.3 Data Reception Process

The following flowchart explains the data reception process for 10BASE-T1S, assuming the use of the AUTOSAR MCAL Ethernet driver.

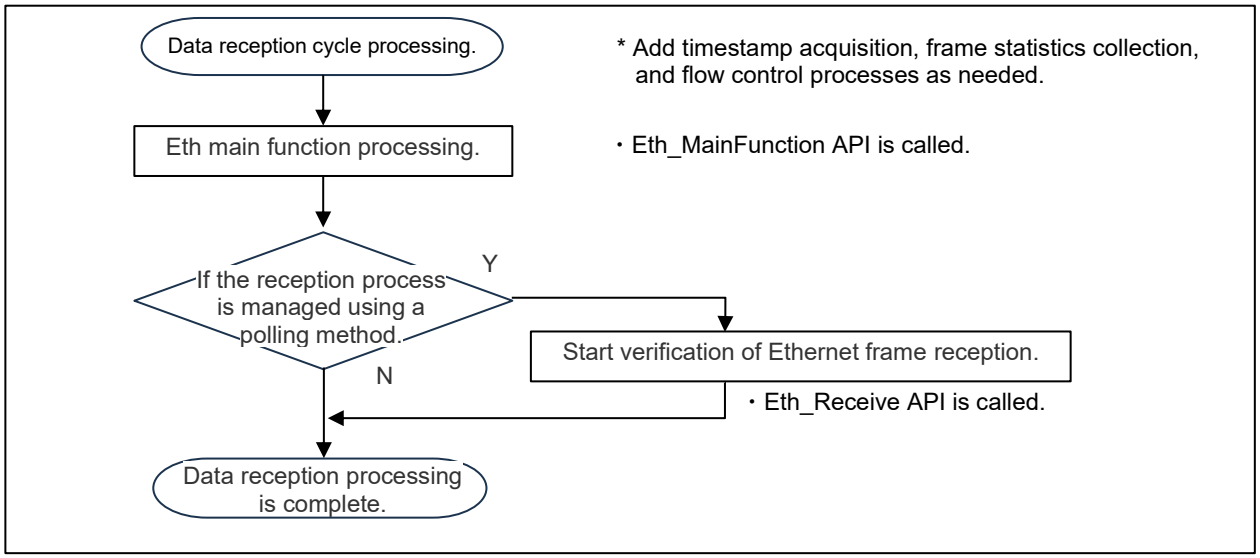


Figure 3-6 Example of data receive processing flow

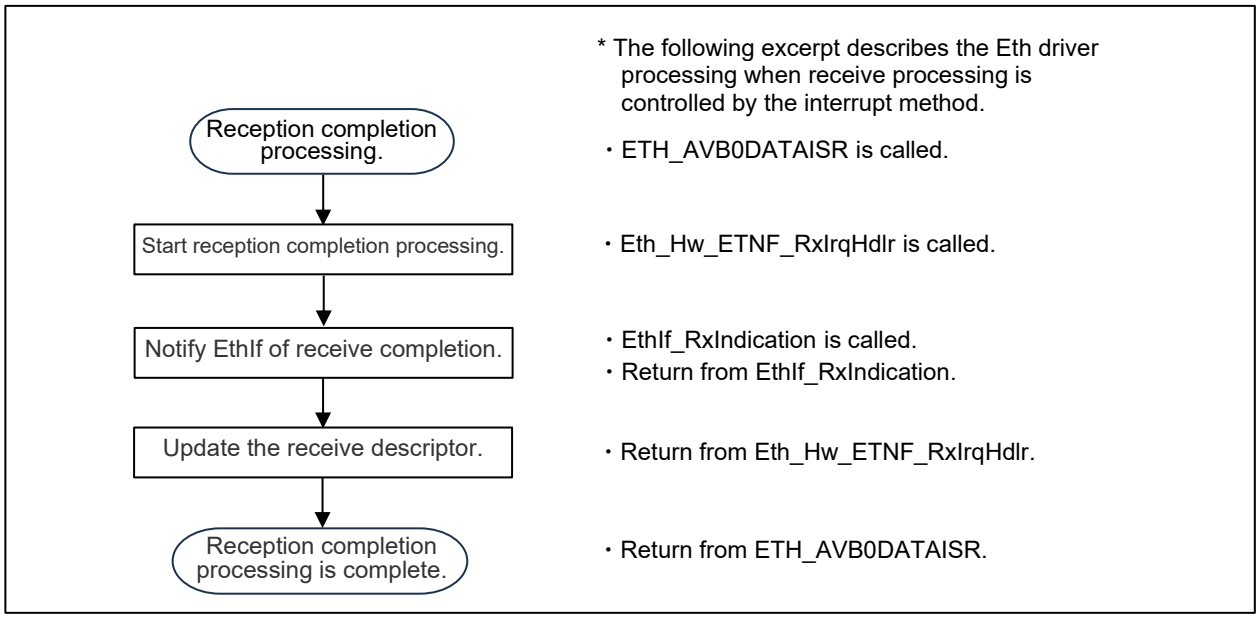


Figure 3-7 Receive completion interrupt processing flow

3.4 Error Detection Process

The following flowchart explains the 10BASE-T1S error detection processing, assuming the use of the AUTOSAR MCAL Eth driver.

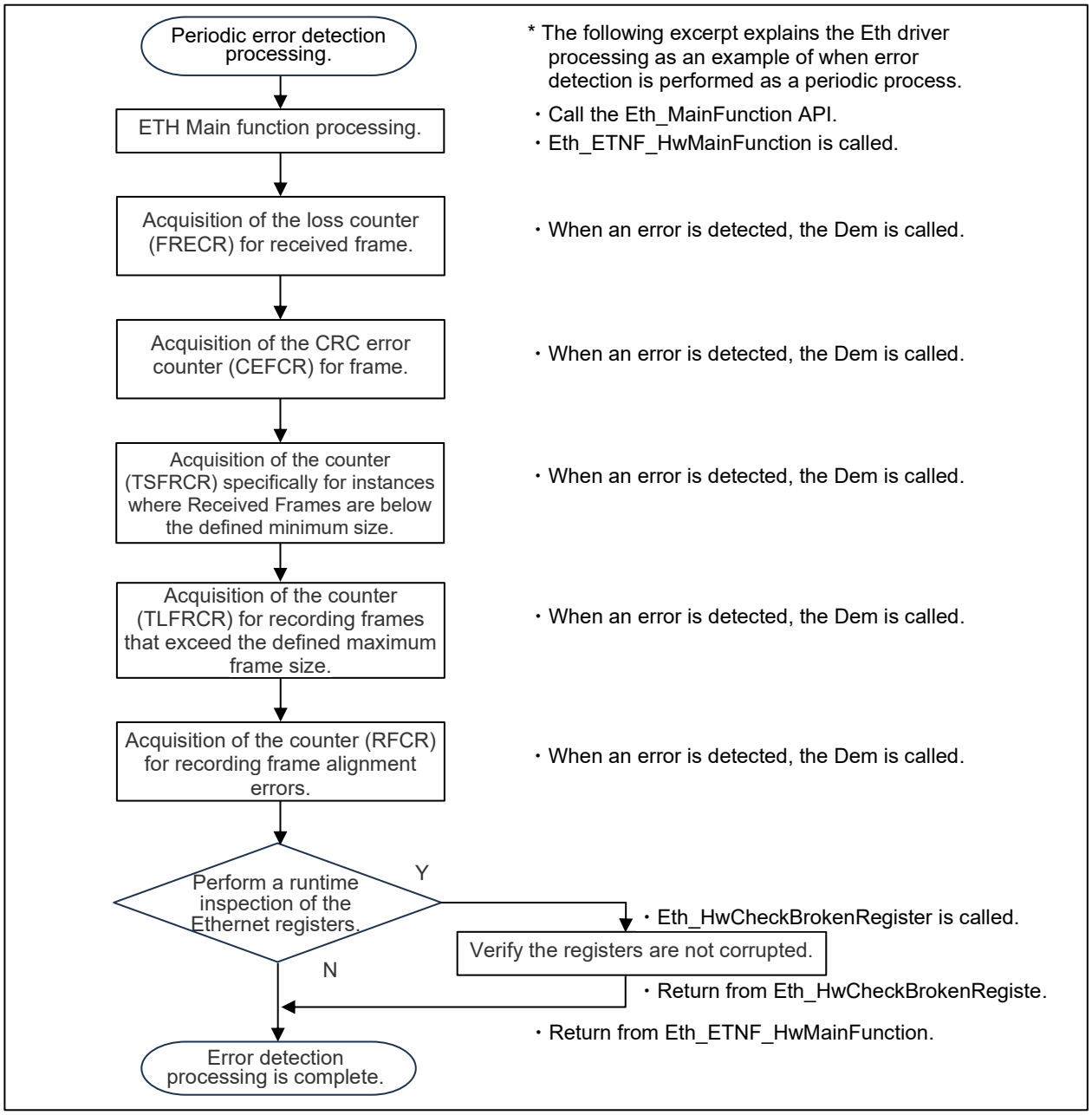


Figure 3-1 Error detection processing flow

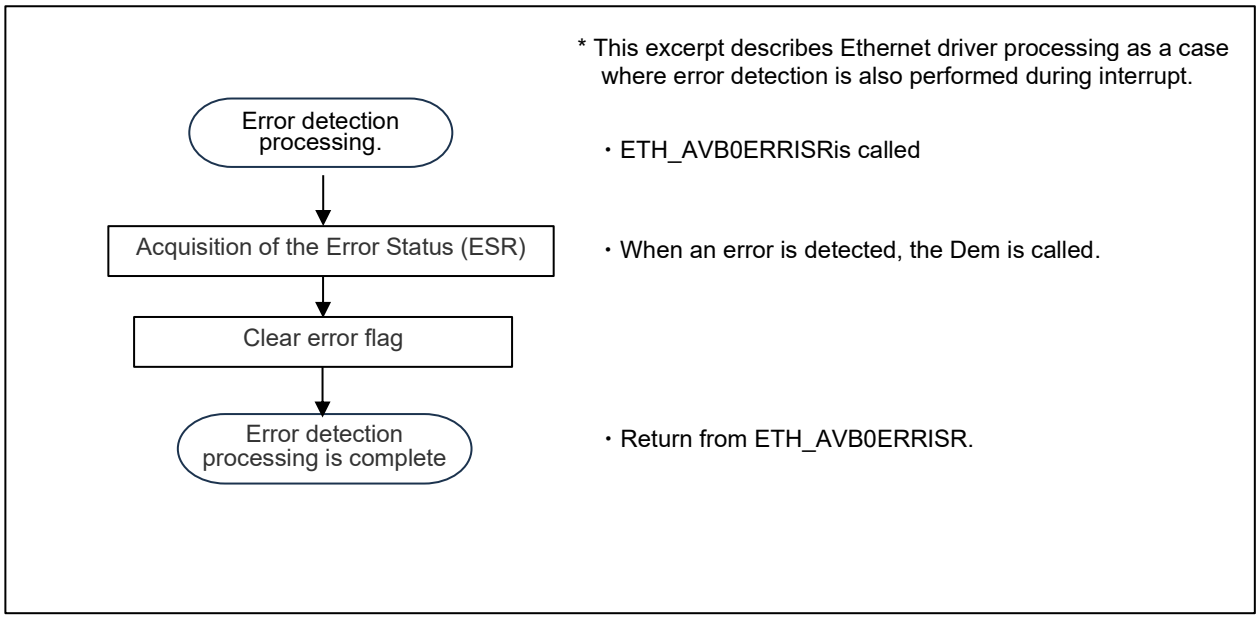


Figure 3-9 Error interrupt processing flow

4. Sample Software

This section explains the processing of sample software.

4.1 Application Processing Overview

The following explains an overview of the sample application processing.

- Sample application processing for Node ID 0.
  - Initialization process
  - The Eth transmission process will begin after a predetermined waiting period. (Multiple frames will be transmitted.) *Waiting for Node ID1 to be able to receive Eth
  - Confirmation processing will be performed for the completion of Eth reception from Node ID1 (receiving multiple frames).
  - Ends the application process. (Infinite loop processing of own address)
- Sample application processing for Node ID1
  - Initialization process
  - Start Eth transmission processing. (transmit one frame)
  - Waiting for completion of Eth reception. (receive multiple frames)
  - After confirmation of Eth reception from Node ID0, Eth transmission processing will begin. (transmit multiple frames)
  - Ends the application process. (Infinite loop processing of own address)

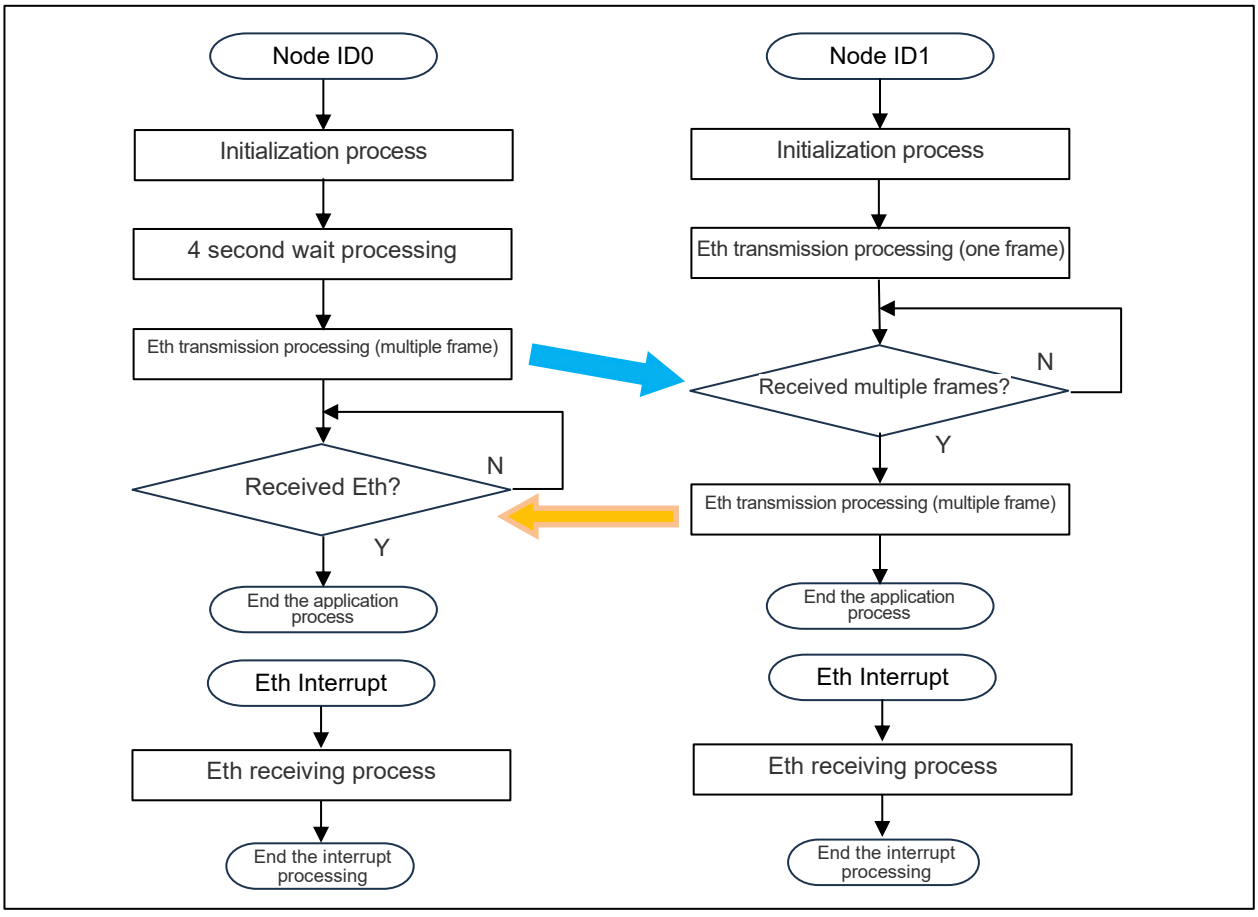


Figure 4-1 Sample application processing flow

4.2 Description of Sample Software Processing

The following is a list of the file structure (excerpt) of this sample software.

Table 4-1 File structure (excerpt)

Folder Name/File Name	Description
\project_eth_10base_t1s_u2c\	Current directory
├command.bat	Batch file for launching cmd window
├\Common\	Common directory
├\mcal\	MCAL directory *Use the ETH driver
├\node_id0\	Node ID0 storage directory
│ ├run_node_id0.mtpj	CS+ project file for Node ID0
│ ├T1S_SampleApp.bat	Batch file for building Node ID0
│ ├\include\	Include directories
│ │ └App_ETH_Common_Sample.h	Common C include file for sample applications
│ ├\src\	C source storage directory
│ │ └App_ETH_Common_Sample.c	Common C source file for sample applications
│ ├\stubs\	Stub storage directory *Dem, Det, EthIf, etc.
│ └\U2C4\	MCAL Config storage directory for RH850/U2U4
│ ├\22_11\	MCAL Config storage directory for AUTOSAR R22-11
│ │ ├\config\	MCAL ETH Config file storage directory
│ │ │ └App_ETH_U2C4_702613FABB_Sample.arxml	Eth sample arxml file for R7F702613FABB
│ │ ├\include\	MCAL ETH Config header storage directory
│ │ │ └Eth_Cfg.h	Sample application Eth Config header file (*1)
│ │ └\src\	MCAL ETH Config C source storage directory
│ │ └Eth_PBcfg.c	Sample application Eth Config C file (*1).
│ ├\obj\	Sample application object storage directory
│ │ └\ghs\	Sample application ghs object storage directory
│ │ ├App_ETH_U2C4_Sample.map	Sample application map file (*1)
│ │ └App_ETH_U2C4_Sample.out	Sample application executable file (*1)
│ ├\include\	Sample application C include directory
│ │ └App_ETH_U2C4_Sample.h	Sample application C include file
│ └\src\	Sample application C source storage directory
│ └App_ETH_U2C4_Sample.c	Sample application C source file
└\node_id1\	Directory for node ID1 *From here onwards, same as ID0

(*1) Files generated by the build

The following section describes the processing of the sample software.(including processing other than Ethernet)

- The procedure for startup processing is as follows.

  (1) Initialize RAM

  (2) Set up stack

  (3) Set interrupt vector

  (4) Set up standard library

- The procedure for clock initialization settings is as follows.

  (1) Wait until the HS IntOSC oscillation stabilizes.

  (2-1) Enable Main OSC.

  (2-2) Wait until the Main OSC oscillation stabilizes.

(3-1) Enable PLL.

(3-2) Wait until the PLL oscillation stabilizes.

(3-3) Based on the system clock gear-up sequence (Chapter 13.6.4 of [REF02]), increase the PLL frequency to 800MHz and the SSCG frequency to 640MHz.

- The procedure for interrupt initialization is as follows.

  (1) Enable the INTETNF0DATA, INTETNF0ERR, INTETNF0MNG, and INTETNF0MAC interrupts.

  (2) Enable the clock supply to the ETNF module.

  (3) Enable global interrupts.

- The procedure for port initialization is as follows:

  (1) Set the Eth-related port control registers PCR20_4, PCR20_3, and PCR20_13.

  (2) The output levels of pins P20_3, P20_4, and P20_13 are changed to 'H' level via the port register output settings (the setting bit is '1').

- The procedure for Ethernet initialization settings is as follows.

  (1) Call Eth_Init to initialize the Ethernet driver (initialize the ETNF module).

  (2) Call Eth_InitT1s to initialize the Ethernet transceiver (initialize the PMA/PCS/PLCA built into the ETNF and the external PHY).

  (3) Call Eth_ReadMii to check the status of the PHY device.

- The procedure for ETH transmission of the application processing is as follows.

  (1-1) Call Eth_SetPhysAddr to set the MAC address.

  (1-2) Call Eth_GetPhysAddr to verify the MAC address.

  (1-3) Call Eth_UpdatePhysAddrFilter to update the address filter.

  (2-1) Call Eth_SetControllerMode to set the controller mode to ACTIVE (transition to operating mode with ETNF).

  (2-2) Call Eth_GetControllerMode to check the state of current operating mode status.

  (3) If the PHY Node ID is 0, a predefined wait period is performed to wait for the startup of PHY Node ID1.

  (4) A UDP frame is transmitted via broadcast.

  ・Call Eth_ProvideTxBuffer for the acquisition of an available buffer.

  ・Copy the data to be transmitted into the buffer.

  ・Call Eth_Transmit to transmit the frame.

  (5) If the PHY Node ID is 1, the system waits until two Ethernet frames have been received.

(6) IEEE 1722 frame (user-defined ACF message) is transmitted via unicast.

・Call Eth_ProvideTxBuffer for the acquisition of an available buffer.

・Copy the data to be transmitted into the buffer.

・Call Eth_Transmit to transmit the frame.

(7) When an Eth message is received (After receiving the Eth message), call Eth_GetCounterValues, Eth_GetTxErrorCounterValues, Eth_GetRxStats, and Eth_GetTxStats to collect frame statistics.

(8-1) If the frame statistics are as expected, close the application normally.

(8-2) If the frame statistics are not as expected, exit the application due to an issue.

- The procedure for receiving Eth in Eth interrupt processing is as follows.

(1-1) MAC address acquisition.

(1-2) Determine whether it is broadcast or unicast, and count it as frame statistics.

(2) Frame type acquisition.

(3) Length acquisition.

(4) Call EthIf_RxIndication to update the frame statistics.

The following table shows the association between the Node ID and MAC address of this sample software.

Table 4-2 MAC Addresses of the sample software

Ethernet Node ID	MAC Address
0	74:90:50:00:00:00
1	74:90:50:00:00:01

The following table shows the configuration parameter settings of the Ethernet transceiver and Eth driver used in this sample software.

Table 4-3 Ethernet transceiver settings (1/2)

Definition Name	Setting Values	Source Comments
T1S_PHY_ADDR	Set the PHY address of the transceiver. Setting value : 0x01U	/* Specify the PHY address of the transceiver */
ETH_PHY_ADDR	Set the PHY address of the microcontroller. Setting value : 0x1FU	/* Specify the built-in PHY address */
T1S_PHY_CONTROL_CFG	Set the CONTROL register. Setting value : T1S_PHY_REG000000_LOOP_OFF T1S_PHY_REG000000_LCTL_ON T1S_PHY_REG000000_LPWR_OFF(fixed) T1S_PHY_REG000000_ISOM_OFF(fixed) T1S_PHY_REG000000_CTEST_OFF	/* CONTROL register configuration settings */ /* Loopback is disabled*/ /* Enable PHY Link Control: Start normal operation */ /* Does not enter low power mode */ /* Does not enter isolation mode */ /* Collision execution mode disabled: Normal operation */
T1S_PHY_T1SPMACTRL_CFG	Set the T1SPMACTRL register. Setting value : T1S_PHY_REG2108F9_LPWR_OFF(fixed) T1S_PHY_REG2108F9_LOOP_OFF	/* Configuration setting of T1SPMACTRL register. */ /* Do not enable low power mode */ /* Do not enable loopback mode */
T1S_PHY_T1SPMATSTM_CFG	Set the T1SPMATSTM register. Setting value : T1S_PHY_REG2108FB_NORMAL	/* Configuration setting of T1SPMATSTM register */ /* Normal operation: Normal operation */
T1S_PHY_T1SPCSCTRL_CFG	Set the T1SPCSCTRL register. Setting value : T1S_PHY_REG2308F3_LOOP_OFF	/* Configuration setting of T1SPCSCTRL register */ /* Loopback is disabled */
T1S_PHY_T1STWEAKS_CFG	Set the T1STWEAKS register. Setting value : T1S_PHY_REG3F8001_PKTLOOP_OFF T1S_PHY_REG3F8001_ENIE_OFF T1S_PHY_REG3F8001_UNJT_OFF T1S_PHY_REG3F8001_RXNRZ_OFF T1S_PHY_REG3F8001_SCRD_OFF T1S_PHY_REG3F8001_NCOLM_ON T1S_PHY_REG3F8001_RXDLY_ON	/* Configuration setting of T1STWEAKS register */ /* TX frames are not reflected to MII RX */ /* Extended noise immunity mode is disabled */ /* Automatic recovery is disabled */ /* Signal is considered to be RZ-encoded */ /* PCS scrambling is disabled */ /* Collision detection is not masked*/ /* MII RX signal is delayed */
T1S_PHY_T1SPLCAEXT_CFG	Set the T1SPLCAEXT register. Setting value : T1S_PHY_REG3F8002_PREN_OFF T1S_PHY_REG3F8002_LDEN_OFF T1S_PHY_REG3F8002_LDR_OFF	/* Configuration setting of T1SPLCAEXT register */ /* PLCA RS operates in standard mode */ /* PLCA reader is selected by node ID */ /* If LDEN = 1, Operate as a PLCA slave */
T1S_PHY_T1SPLCAEXT_CFG	Set the T1SPLCAEXT register. Setting value : T1S_PHY_REG3F8002_PREN_OFF T1S_PHY_REG3F8002_LDEN_OFF T1S_PHY_REG3F8002_LDR_OFF	/* Configuration setting of T1SPLCAEXT register */ /* PLCA RS operates in standard mode */ /* PLCA reader is selected by node ID */ * If LDEN = 1, Operate as a PLCA slave */
T1S_PHY_T1SPMATUNE0_CFG	Set the T1SPMATUNE0 register. Setting value : T1S_PHY_REG3F8003_NNTHR T1S_PHY_REG3F8003_DRFTW	/* Configuration setting of T1SPMATUNE0 register */ /* Set the threshold for the NN comparator */ /* Set the time window for the drift compensator */
T1S_PHY_T1SPMATUNE1_CFG	Set the T1SPMATUNE1 register. Setting value : T1S_PHY_REG3F8004_JJHHTHR T1S_PHY_REG3F8004_JJTHR	/* Configuration setting of T1SPMATUNE1 register */ /* Set the threshold for the JJHH comparator */ /* Set the JJ Comparator Threshold */

Table 4-4 Ethernet transceiver settings (2/2)

Definition Name	Setting Values	Source Comments
T1S_PHY_PLCACTRL0_CFG	Set the PLCACTRL0 register. Setting value : T1S_PHY_REG3FCA01_EN_ON	/* Configuration setting of PLCACTRL0 register */ /* PLCA RS is enabled */
T1S_PHY_PLCACTRL1_CFG	Set the PLCACTRL1 register. Setting value : T1S_PHY_REG3FCA02_NCNT T1S_PHY_REG3FCA02_ID	/* Configuration setting of PLCACTRL1 register */ /* Set the maximum number of nodes */ /* Set PLCA Node ID */
T1S_PHY_REG3FCA02_NCNT	Set NCNT. Setting value : 8	-
T1S_PHY_REG3FCA02_ID	Set the ID. Setting value : 0 or 1 *Varies depending on the project.	-
T1S_PHY_PLCATOTMR_CFG	Set the PLCATOTMR register. Setting value : T1S_PHY_REG3FCA04_TOTMR	/* Configuration setting of PLCATOTMR register */ /* Set PLCA Transmission Opportunity Timer */
T1S_PHY_PLCABURST_CFG	Set the PLCABURST register. Setting value : T1S_PHY_REG3FCA05_MAXBC T1S_PHY_REG3FCA05_BTMR	/* Configuration setting of PLCABURST register */ /* Set the number of additional packets that can be transmitted */ /* Set the wait time for burst concatenation */
T1S_PHY_ADDR1_REG18H_SET_CFG	Set the set bit value of the XCVR_ANALOG_SET_2 Setting value : 0x6000U	/* XCVR_ANALOG_SET_2 settings */
T1S_PHY_ADDR1_REG18H_CLR_CFG	Set the clear bit value of the XCVR_ANALOG_SET_2. Setting value : ~0x0000U	-
T1S_PHY_ADDR1_REG1FH_SET_CFG	Set the set bit value of the XCVR_ANALOG_SET_9. Setting value : 0x0018U	/* XCVR_ANALOG_SET_9 settings */
T1S_PHY_ADDR1_REG1FH_CLR_CFG	Set the clear bit value of the XCVR_ANALOG_SET_9. Setting value : ~0x0000U	-

Table 4-5 Eth driver settings (1/3)

Parameter Name (Tree notation)	Settings	Overview
/Renesas/	-	-
└/EcucDefs_Eth/	-	-
└/Eth/	-	-
├/EthGeneral/	-	-
│ ├ EthDevErrorDetect	false	Enable/Disable development error detection and notification
│ ├ EthMultiCoreSupport	false	Enable/Disable Multicore support mode
│ ├ EthGlobalTimeSupport	false	Enable/Disable the GlobalTime API
│ ├ EthIndex	0	Specify the Instance ID
│ ├ EthMainFunctionPeriod	0.001	Eth_MainFunction Specify the period in seconds
│ ├ EthMaxCtrlsSupported	2	Total number of controllers
│ ├ EthVersionInfoApi	false	Enable/Disable Version information API
│ ├ EthVersionCheckExternalModules	false	Enable/Disable AUTOSAR version check
│ ├ EthCriticalSectionProtection	true	Specify CPU load reduction for the ETH driver
│ ├ EthEnableInputClockRefImmediateValue	true	Enable/Disable PBUS direct input clock values
│ ├ EthInputClockRefImmediateValue	100000000	PBUS clock setting in the ETH macro
│ ├ EthTimeout	0.012	Specify timeout period
│ ├ EthGetRxStatsApi	true	Enable/Disable Eth_GetRxStats APIs
│ ├ EthGetTxErrorCounterValuesApi	true	Enable/Disable Eth_GetTxErrorCounterValues APIs
│ ├ EthGetCounterValuesApi	true	Enable/Disable Eth_GetCounterValues APIs
│ ├ EthInterruptConsistencyCheck	false	Enable/Disable ISR interrupt consistency check
│ ├ EthGetTxStatsApi	true	Enable/Disable Eth_GetTxStats API
│ ├ EthSwitchManagementSupport	false	Enable/Disable Ethswt management
│ ├ EthDeInitApi	true	Enable/Disable Eth_DeInit API
│ ├ EthQosSupport	true	Enable/Disable API for transmission-support
│ ├ EthRegisterCheckInitTime	false	Enable/Disable register checking in Eth_Init
│ ├ EthRegisterCheckRunTime	false	Enable/Disable register checking in Eth_MainFunction
│ ├ EthUpdatePhysAddrFilter	true	Enable/Disable the Eth_UpdatePhysAddrFilter API
│ ├ EthGetDropCountApi	false	Enable/Disable the Eth_GetDropCount API
│ ├ EthGetEtherStatsApi	false	Enable/Disable the Eth_GetEtherStats API
│ ├ EthDeviceName	R7F702613FABB	Device name
│ ├ EthIsrCategory	CAT1	ISR category by module
│ └/EthCtrlOffloading/	-	-
│ ├ EthCtrlEnableOffloadChecksumIPv4	false	Discard IPv4 frames with checksum mismatch
│ ├ EthCtrlEnableOffloadChecksumICMP	false	Discard ICMP frames with checksum mismatch
│ ├ EthCtrlEnableOffloadChecksumTCP	false	Discard TCP frames with checksum mismatch
│ └ EthCtrlEnableOffloadChecksumUDP	false	Discard UDP frames with checksum mismatch
└/EthConfigSet/	-	-
└/EthCtrlConfig/	-	-
├ EthBeTimeStampStore	false	Enable/Disable inclusion of timestamp information
├ EthCtrlConfigSwBufferHandling	false	Enable/Disable SW buffer management
├ EthCtrlEnableMii	true	Enable/Disable MII access for the transceiver
├ EthCtrlEnableRxInterrupt	true	Enable/Disable Receive interrupts
├ EthCtrlEnableSpiInterface	false	Enable/Disable SPI interface
├ EthCtrlEnableTxInterrupt	true	Enable/Disable transmit interrupts and Enable/Disable the Eth_TxConfirmation API
├ EthCtrlIdx	0	Specify the controller instance ID
├ EthEnableCBS	true	Enable/Disable Credit-based shaping
├ EthInternalLoopBack	false	Internal Loopback mode
├ EthNwCntrlFiltering	false	Enable/Disable Network filtering
├ EthRamSize	102400	Size of user RAM
├ EthSRPTalkerFiltering	true	Enable/Disable individual filtering function
├ EthStreamTimeStampStore	false	Enable/Disable addition of timestamp information
├ EthEnableCRSCheck	false	Enable/Disable CRS assert check

Table 4-6 Eth driver settings (2/3)

Parameter Name (Tree notation)	Settings	Overview
├ EthCtrlMacLayerSpeed	ETH_MAC_LAYER_SPEED_10M	Define the baud rate of the MAC layer
├ EthCtrlMacLayerType	ETH_MAC_LAYER_TYPE_XMII	Defining MAC layer types
├ EthCtrlPhyAddress	74:90:50:00:00:00 (*2)	Specify the byte order of the MAC address
├ EthDuplexMode	ETH_T1S_MODE	Specify the duplex mode settings
├ EthStreamFiltering	ETH_AVBNMQUE0	Specify the stream filtering options
├ EthTxQuePriority	ETH_AVBDEF	Specify the transmit synchronization mode 0 (best effort)
├ EthUnitSelection	ETH_ETNF0	Select Ethernet TSN and Ethernet AVB units
├ EthCtrlMacLayerSubType	STANDARD	
├ EthPulseGenerationMode	ETH_PPS_GPTP_TIMER_VALUE	Set the pulse per second generation mode
├/EthCtrlConfigEgress/	-	-
│ ├/EthCtrlConfigEgressQueue/	-	-
│ │ ├ EthCtrlConfigEgressQueueBufLenByte	0	Egress FIFO length. U2C not supported
│ │ ├ EthCtrlConfigEgressQueueBufTotal	0	Number of Egress FIFO buffers. U2C not supported
│ │ └ EthCtrlConfigEgressQueueIdx	0	Egress FIFO index. U2C not supported
│ ├/EthCtrlConfigScheduler/	-	-
│ │ └/EthCtrlConfigSchedulerPredecessor/	-	-
│ │ └ EthCtrlConfigSchedulerPredecessorOrder	0	Define the scheduler order. U2C not supported
│ └/EthCtrlConfigShaper/	-	-
│ ├ EthCtrlConfigShaperMaxCredit	0	The maximum amount of credits that can be accumulated in the queue. U2C not supported
│ └ EthCtrlConfigShaperMinCredit	0	Minimum amount of credits that can be accumulated in the queue. U2C not supported
├/EthCtrlPriority/	-	-
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping）	-（Multiplexing）
│ │ └ EthCtrlPriorityValue	0	Specific traffic class priority
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_001）	-（Multiplexing）
│ │ └ EthCtrlPriorityValue	1	Specific traffic class priority
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_002）	-（Multiplexing）
│ │ └ EthCtrlPriorityValue	2	Specific traffic class priority
│ └/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_003）	-（Multiplexing）
│ └ EthCtrlPriorityValue	3	Specific traffic class priority
├/EthMacConfig/	-	-
│ ├/EthFlowControlConfig/	-	-
│ │ ├ EthPauseFrame	false	Pause frame control
│ │ ├ EthPauseFrameRetransmissionTime	0	Pause frame retransmission time
│ │ ├ EthPauseTime	1	Pause time
│ │ └ EthPauseTimeZero	false	Enable/disable transmission and reception of pause frames at TIME = 0.
│ └ EthLinkVerificationTime	10	Link verification timer. Defined in milliseconds
├/EthPhyConfig/	-	-
│ ├ EthMdcClockSelection	19	Specify the MDC clock cycle
│ ├ EthMdioCaptureTime	MDIO_CAPTURE_TIME_1_CLK_CYCLE	Specify the capture time of MDIO communication
│ └ EthMdioHoldTime	MDIO_HOLD_TIME_1_CLK_CYCLE	Specify the hold time for MDIO communication
├/EthRxCommonConfig/	-	-
│ ├ EthRxMaxFrameSize	1518	Specify the maximum receive frame size

Table 4-7 Eth driver settings (3/3)

Parameter Name (Tree notation)	Settings	Overview
│ ├/EthTasCommonConfig/	-	-
│ │ ├ EthTasEnable	false	TAS gate setting
│ │ └ EthTasTransmissionJitter	75	TAS Transmit Jitter [%]
│ └ EthFragmentSize	ETH_FRAGMENT_64_BYTE	Outbound fragment size setting
├/EthTxQueueConfig/	（EthTxQueueConfig）	-（Multiplexing）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	Select the supported queue priority algorithm
│ ├ EthTxQueueBufs	8	Setting the number of transmit descriptors
│ ├ EthTxQueueIdx	0	Index of the transmit queue to be enabled
│ ├ EthTxQueueMaxFrameSize	1518	Specify the maximum frame size for each transmit queue
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Select the frame type for the Tx queue
├/EthTxQueueConfig/	（EthTxQueueConfig_001）	-（Multiplexing）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	Select the supported queue priority algorithm
│ ├ EthTxQueueBufs	8	Setting the number of transmit descriptors
│ ├ EthTxQueueIdx	1	Index of the transmit queue to be enabled
│ ├ EthTxQueueMaxFrameSize	1518	Specify the maximum frame size for each transmit queue
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Select the frame type for the Tx queue
├/EthTxQueueConfig/	（EthTxQueueConfig_002）	-（Multiplexing）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ ├ EthCtrlTxQueueBwFraction	10	Percentage of bandwidth allocated to queue [%]
│ │ └ EthTxQueuePolicy	ETH_CBS	Select the supported queue priority algorithm
│ ├ EthTxQueueBufs	8	Setting the number of transmit descriptors
│ ├ EthTxQueueIdx	2	Index of the transmit queue to be enabled
│ ├ EthTxQueueMaxFrameSize	1518	Specify the maximum frame size for each transmit queue
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Select the frame type for the Tx queue
├/EthTxQueueConfig/	（EthTxQueueConfig_003）	-（Multiplexing）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	Select the supported queue priority algorithm
│ ├ EthTxQueueBufs	8	Setting the number of transmit descriptors
│ ├ EthTxQueueIdx	3	Index of the transmit queue to be enabled
│ ├ EthTxQueueMaxFrameSize	1518	Specify the maximum frame size for each transmit queue
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Select the frame type for the Tx queue
├/EthRxQueueConfig	（EthRxQueueConfig）	-（Multiplexing）
│ ├ EthRxQueueBufs	8	Setting the number of receive descriptors
│ ├ EthRxQueueIdx	0	Index of the receive queue to be enabled
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	Specify the pattern address used for filtering
├/EthRxQueueConfig	（EthRxQueueConfig_001）	-（Multiplexing）
│ ├ EthRxQueueBufs	8	Setting the number of receive descriptors
│ ├ EthRxQueueIdx	1	Index of the receive queue to be enabled
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	Specify the pattern address used for filtering
├/EthRxQueueConfig	（EthRxQueueConfig_002）	-（Multiplexing）
│ ├ EthRxQueueBufs	8	Setting the number of receive descriptors
│ ├ EthRxQueueIdx	2	Index of the receive queue to be enabled
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	Specify the pattern address used for filtering
└/EthRxQueueConfig	（EthRxQueueConfig_003）	-（Multiplexing）
├ EthRxQueueBufs	8	Setting the number of receive descriptors
├ EthRxQueueIdx	3	Index of the receive queue to be enabled
└ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	Specify the pattern address used for filtering

(*2) Parameter value of Node ID=0

Appendix.A Sample Application Code

File name: \node_id0\src\App_ETH_Common_Sample.c

```
/*==================================================================================*/
/* Project      = AUTOSAR Renesas U2C MCAL Components                               */
/* Module       = App_ETH_Common_Sample.c                                           */
/* SW-VERSION   = 1.0.0                                                             */
/*==================================================================================*/
/*                                     COPYRIGHT                                    */
/*==================================================================================*/
/* (c) 2025 Renesas Electronics Corporation. All rights reserved.                   */
/*==================================================================================*/
/* Purpose:                                                                         */
/* This file contains sample application for ETH Driver Component                   */
/*                                                                                  */
/*==================================================================================*/
/*                                                                                  */
/* Unless otherwise agreed upon in writing between your company and                 */
/* Renesas Electronics Corporation the following shall apply!                       */
/*                                                                                  */
/* Warranty Disclaimer                                                              */
/*                                                                                  */
/* There is no warranty of any kind whatsoever granted by Renesas. Any              */
/* warranty is expressly disclaimed and excluded by Renesas, either expressed       */
/* or implied, including but not limited to those for non-infringement of           */
/* intellectual property, merchantability and/or fitness for the particular         */
/* purpose.                                                                         */
/*                                                                                  */
/* Renesas shall not have any obligation to maintain, service or provide bug        */
/* fixes for the supplied Product(s) and/or the Application.                        */
/*                                                                                  */
/* Each User is solely responsible for determining the appropriateness of           */
/* using the Product(s) and assumes all risks associated with its exercise          */
/* of rights under this Agreement, including, but not limited to the risks          */
/* and costs of program errors, compliance with applicable laws, damage to          */
/* or loss of data, programs or equipment, and unavailability or                    */
/* interruption of operations.                                                      */
/*                                                                                  */
/* Limitation of Liability                                                          */
/*                                                                                  */
/* In no event shall Renesas be liable to the User for any incidental,              */
/* consequential, indirect, or punitive damage (including but not limited           */
/* to lost profits) regardless of whether such liability is based on breach         */
/* of contract, tort, strict liability, breach of warranties, failure of            */
/* essential purpose or otherwise and even if advised of the possibility of         */
/* such damages. Renesas shall not be liable for any services or products           */
/* provided by third party vendors, developers or consultants identified or         */
/* referred to the User by Renesas in connection with the Product(s) and/or         */
/* the Application.                                                                 */
/*                                                                                  */
/*==================================================================================*/
/* Environment:                                                                     */
/*              Devices:        U2C                                                 */
/*==================================================================================*/

/***********************************************************************************
**                      Revision Control History                                  **
***********************************************************************************/
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
/***********************************************************************************/

/************************************************************************************
**                      Include Section                                           **
************************************************************************************/

#include "App_Eth_Common_Sample.h"
#include "App_Eth_Device_Sample.h"
#include <stdio.h>
#include <string.h>
#if ( ETH_MACRO_ETNB == STD_ON )
#include "Eth_ETNB_Ram.h"
#endif /* ( ETH_MACRO_ETNB == STD_ON ) */
#if ( ETH_MACRO_ETNF == STD_ON )
#include "Eth_ETNF_Ram.h"
#endif /* ( ETH_MACRO_ETNF == STD_ON ) */
/************************************************************************************
**                             Macros                                             **
************************************************************************************/
#define SAMPLE_TOTAL                        (3U)
#define ETH_FAILED                          (0)
#define ETH_PASSED                          (1)
#define ETH_AVBTP_TYPE                      (0x22F0U)
#define ETH_IPV4_TYPE                       (0x0800U)
#define PHY_TRCV_IDX                        (0x1FU)
#define PHY_TRCV_CTRL_REG                   (0U)
#define PHY_TRCV_STATUS_REG                 (1U)
#define FRAME_LEN                           (48U)       /* Without header and FCS.    */
#define NORMAL_HEADER_LEN                   (14U)
#define SRC_MACADDR_LEN                     (6U)
#define DST_MACADDR_LEN                     (6U)
#define STREAMID_LEN                        (8U)
#define ETHTYPE_LEN                         (2U)
#define ETH_VLFRAME_SIZE                    (1522U)
#define PHY_LINKUP_BIT                      (0x002CU)

#define RX_QUEUE_0                          (0U)
#define RX_QUEUE_1                          (1U)
#define RX_QUEUE_2                          (2U)
#define RX_QUEUE_3                          (3U)
/* Time-out */
#define ETH_TIMEOUT                         ((uint32)0x200000)

#define CRC_ERROR                           (1U)
#define UNDERSIZE_PACKET_ERROR              (2U)
#define OVERSIZE_PACKET_ERROR               (3U)
#define ALIGNMENT_ERROR                     (4U)
#define LATE_COLLISION                      (13U)

#define DROP_EVENTS                         (0U)
#define BROADCAST_PACKETS                   (3U)
#define MULTICAST_PACKETS                   (4U)
#define CRC_ALIGN_ERRORS                    (5U)
#define UNDERSIZE_ERRORS                    (6U)
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
#define OVERSIZE_ERRORS                     (7U)
#define LATE_COLLISIONS                     (10U)

#define RX_UNICAST_FRAME_TOTAL              (2U)
#define OCTET_TOTAL                         (186U)

/*******************************************************************************
**                        Global variables                                    **
*******************************************************************************/
volatile uint8          GaaRxEthFrame[ETH_TOTAL_CTRL_CONFIG][ETH_VLFRAME_SIZE];
volatile uint8          GaaRxSrcAddr[ETH_TOTAL_CTRL_CONFIG][SRC_MACADDR_LEN];
volatile uint8          GucTxConfirmed[ETH_TOTAL_CTRL_CONFIG];
volatile uint16         GusMsgLength[ETH_TOTAL_CTRL_CONFIG];
volatile uint16         GusRxFrameCnt[ETH_TOTAL_CTRL_CONFIG];
volatile uint16         GusRxLenByte[ETH_TOTAL_CTRL_CONFIG];
volatile Eth_FrameType  GusRxFrameType[ETH_TOTAL_CTRL_CONFIG];
volatile boolean        EthRcvBroadcastMsg;
volatile uint8          EthPassedCount[70];
volatile uint8          EthCheckCount;
volatile uint8          EthModeActiveCnt;
volatile uint8          EthModeDownCnt;

#if ( ETH_GLOBAL_TIME_SUPPORT == STD_ON )
Eth_TimeStampQualType TxTimeQual;
Eth_TimeStampType TxTimeStamp[ETH_TOTAL_CTRL_CONFIG];
#endif /* ( ETH_GLOBAL_TIME_SUPPORT == STD_ON ) */

/* Board MAC address */
static uint8 mac_board_addr[2U][SRC_MACADDR_LEN] =
{
    { 0x74U, 0x90U, 0x50U, 0x00U, 0x00U, 0x00U },
    { 0x74U, 0x90U, 0x50U, 0x00U, 0x00U, 0x00U }
};

/* Target MAC address */
static uint8 mac_tgt_addr[DST_MACADDR_LEN]      = { 0xFFU, 0xFFU, 0xFFU, 0xFFU, 0xFFU, 0xFFU };

/* IEEE 1722 frames MAC address*/
static uint8 mac_tgt_addr0[DST_MACADDR_LEN]     = { 0x74U, 0x90U, 0x50U, 0x00U, 0x00U, 0x01U };
static uint8 mac_tgt_addr1[DST_MACADDR_LEN]     = { 0x74U, 0x90U, 0x50U, 0x00U, 0x00U, 0x02U };

/*-------------------------------------------------*/
/* Receive and Transmit Buffers */
/*-------------------------------------------------*/
static uint8 TxEthFrame[SAMPLE_TOTAL][FRAME_LEN] =
{
    /* Index 0: UDP frame expected to get filter in Rx Queue 0 */
    {
        0x45U, 0x00U,
        0x00U, 0x30U, 0x63U, 0x27U, 0x00U, 0x00U, 0x80U, 0x11U, 0x35U, 0x2EU,
          12U,   34U,   56U,   78U,   12U,   34U, 0xFFU, 0xFFU, 0xD6U, 0x0FU,
        0x13U, 0x88U, 0x00U, 0x1CU, 0x63U, 0xAFU, 0xAAU, 0xAAU, 0xAAU, 0xAAU,
        0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU,
        0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU, 0xAAU
    },
```

File name: App_ETH_Common_Sample.c

```
    /* Index 1: IEEE 1722 Frame expected to get filter in Rx Queue 1 */
    {
        0x05U, 0x80U,
        0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U,
        0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x18U,
        0x00U, 0x00U, 0xFEU, 0x06U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U,
        0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U,
        0x55U, 0x55U, 0x55U, 0x55U, 0x55U, 0x55U
    },
    /* Index 2: IEEE 1722 Frame expected to get filter in Rx Queue 2 */
    {
        0x05U, 0x80U,
        0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U,
        0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x18U,
        0x00U, 0x00U, 0xFEU, 0x06U, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU,
        0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU,
        0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU, 0xCCU
    }
};

#if ( ETH_VERSION_INFO_API == STD_ON )
/* Variable used to store the Module Version Info */
static Std_VersionInfoType              GddVersionInfo;
uint8                                   GucVerCheckStatus;
#endif /* ( ETH_VERSION_INFO_API == STD_ON ) */
uint8                                   GucTxFrameSentCnt;

/**********************************************************************************
**                        User function prototypes                                **
**********************************************************************************/
static void SendData
(
    uint8                               CtrlIdx,
    uint8                              *Buf,
    uint16                              LenByte,
    Eth_FrameType                       FrameType,
    uint8                               Priority,
    const uint8                        *PhysAddrPtr
);

/**********************************************************************************/
/* Test OK                                                                        */
/**********************************************************************************/
void sample_end( void )
{
    /* The transmitted message and the received message matched.(OK) */
    while ( 1U )
    {
        /* Do nothing */
    }
}

/**********************************************************************************/
/* Test NG                                                                        */
/**********************************************************************************/
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
void sample_NG_end( void )
{
    /* The transmitted message and receiving message are different.(NG) */
    while ( 1U )
    {
        /* Do nothing */
    }
}

/*****************************************************************************/
/* Sample application for ETH Driver Component                               */
/*****************************************************************************/
int main( void )
{
    volatile uint32                     LoopCount;
    Std_ReturnType                      LucReturnValue;
    uint8                               LucCtrlCnt;
#if ( ETH_CTRL_ENABLE_MII == STD_ON )
    uint32                              LulDelayCounter;
#endif /* ( ETH_CTRL_ENABLE_MII == STD_ON ) */
    Eth_ModeType                        LenEthCtrlMode[ETH_TOTAL_CTRL_CONFIG];
#if ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON )
    Eth_RxStatusType                    LenEthRxStatus[ETH_TOTAL_CTRL_CONFIG];
 #if ( ETH_AR_VERSION >= ETH_AR_431_VERSION )
    Eth_RxStatusType                    LenEthStatusRxQueue0[ETH_TOTAL_CTRL_CONFIG];
  #if ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) )
    Eth_RxStatusType                    LenEthStatusRxQueue1[ETH_TOTAL_CTRL_CONFIG];
    Eth_RxStatusType                    LenEthStatusRxQueue2[ETH_TOTAL_CTRL_CONFIG];
    Eth_RxStatusType                    LenEthStatusRxQueue3[ETH_TOTAL_CTRL_CONFIG];
  #endif /* ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) ) */
 #endif /* ( ETH_AR_VERSION >= ETH_AR_431_VERSION ) */
#endif /* ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON ) */
#if ( ETH_GLOBAL_TIME_SUPPORT == STD_ON )
    Eth_TimeStampQualType               LenEthTimeQualPtr;
    Eth_TimeStampType                   LstCurrentTime;
#endif /* ( ETH_GLOBAL_TIME_SUPPORT == STD_ON ) */
#if ( ETH_GET_DROP_COUNT_API == STD_ON )
    uint32                              DropCount[15U] = { 0U };
    uint8                               CountValues;
#endif /* ( ETH_GET_DROP_COUNT_API == STD_ON ) */
#if ( ETH_GET_COUNTER_VALUES_API == STD_ON )
    Eth_CounterType                     CounterPtr;

    CounterPtr.DropPktBufOverrun    = (uint32)0U;
    CounterPtr.DropPktCrc           = (uint32)0U;
    CounterPtr.UndersizePkt         = (uint32)0U;
    CounterPtr.OversizePkt          = (uint32)0U;
    CounterPtr.AlgnmtErr            = (uint32)0U;
#endif /* ( ETH_GET_COUNTER_VALUES_API == STD_ON ) */
#if ( ETH_GET_TX_ERROR_COUNTER_VALUES_API == STD_ON )
    Eth_TxErrorCounterValuesType        TxErrorCounterValues;
#endif /* ( ETH_GET_TX_ERROR_COUNTER_VALUES_API == STD_ON ) */
#if ( ETH_GET_ETHER_STATS_API == STD_ON )
    uint32                              etherStats[18U] = { 0U };
#endif /* ( ETH_GET_ETHER_STATS_API == STD_ON ) */
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
#if ( ETH_GET_RX_STATS_API == STD_ON )
    Eth_RxStatsType                     RxStats;

    RxStats.RxStatsPkts             = (uint32)0U;
    RxStats.RxStatsBroadcastPkts    = (uint32)0U;
    RxStats.RxStatsMulticastPkts    = (uint32)0U;
    RxStats.RxStatsCrcAlignErrors   = (uint32)0U;
    RxStats.RxStatsUndersizePkts    = (uint32)0U;
    RxStats.RxStatsOversizePkts     = (uint32)0U;
#endif /* ( ETH_GET_RX_STATS_API == STD_ON ) */
#if ( ETH_GET_TX_STATS_API == STD_ON )
    Eth_TxStatsType                     TxStats;
#endif /* ( ETH_GET_TX_STATS_API == STD_ON ) */
    uint32                              LulInc;
#if ( ETH_CTRL_ENABLE_MII == STD_ON )
    uint16                              LusRegValue;
#endif /* ( ETH_CTRL_ENABLE_MII == STD_ON ) */
    uint8                               LucRetrieveMacAddr[ETH_TOTAL_CTRL_CONFIG][SRC_MACADDR_LEN];

    for ( LucCtrlCnt = (uint8)0U; LucCtrlCnt < (uint8)ETH_TOTAL_CTRL_CONFIG; LucCtrlCnt++ )
    {
#if ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON )
        LenEthCtrlMode[LucCtrlCnt]  = (Eth_ModeType)ETH_MODE_DOWN;
        LenEthRxStatus[LucCtrlCnt]  = (Eth_RxStatusType)ETH_NOT_RECEIVED;
#endif /* ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON ) */

        GusRxFrameCnt[LucCtrlCnt]   = (uint16)0U;
        GusMsgLength[LucCtrlCnt]    = (uint16)0U;
    }


#if ( ETH_GLOBAL_TIME_SUPPORT == STD_ON )
    LstCurrentTime.nanoseconds      = (uint32)0U;
    LstCurrentTime.seconds          = (uint32)0U;
    LstCurrentTime.secondsHi        = (uint16)0U;
#endif /* ( ETH_GLOBAL_TIME_SUPPORT == STD_ON ) */
    GucTxFrameSentCnt               = (uint8)0U;
#if ( ETH_CTRL_ENABLE_MII == STD_ON )
    LulDelayCounter                 = (uint32)0U;
#endif /* ( ETH_CTRL_ENABLE_MII == STD_ON ) */
#if ( ETH_GET_DROP_COUNT_API == STD_ON )
    CountValues                     = (uint8)15U;
#endif /* ( ETH_GET_DROP_COUNT_API == STD_ON ) */
    /********************** Check version info **********************/

#if ( ETH_VERSION_INFO_API == STD_ON )
    /* Invoke Eth_GetVersionInfo to get version info */
    Eth_GetVersionInfo( &GddVersionInfo );
    if ( ( (uint16)ETH_VENDOR_ID == GddVersionInfo.vendorID )
      && ( (uint16)ETH_MODULE_ID == GddVersionInfo.moduleID )
      && ( (uint8)ETH_SW_MAJOR_VERSION == GddVersionInfo.sw_major_version )
      && ( (uint8)ETH_SW_MINOR_VERSION == GddVersionInfo.sw_minor_version )
      && ( (uint8)ETH_SW_PATCH_VERSION == GddVersionInfo.sw_patch_version ) )
    {
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
        GucVerCheckStatus               = (uint8)ETH_TRUE;
        EthPassedCount[EthCheckCount]   = (uint8)ETH_PASSED;
    }
    else
    {
        /* If the version information is incorrect */
        GucVerCheckStatus               = (uint8)ETH_FALSE;
    }
    EthCheckCount++;
#endif /* ( ETH_VERSION_INFO_API == STD_ON ) */

    /************************** Initilazation  *****************************/
    /* Initialize Clock */
    Clock_Init();

#if 0   /* 1: Disable ECC function (for CPU processing time test) */
    ECC_DISABLE
#endif /* 0 */

    /* Initialize MCU */
    Mcu_Init();         /* Enable interrupts */

    /* Initialize WDG */
    Wdg_Init();

    /* Initialize PORT for Ethernet */
    Port_Init();

    /* Invoke Eth_Init to initialize the Ethernet Driver */
    Eth_Init( Eth_Config );

    /************************** Set up transceiver *****************************/

#if ( ETH_CTRL_ENABLE_MII == STD_ON )
    for ( LucCtrlCnt = (uint8)0U; LucCtrlCnt < (uint8)ETH_TOTAL_CTRL_CONFIG; LucCtrlCnt++ )
    {
        switch ( Eth_Config->pCtrlConfig->pEthConfig[LucCtrlCnt].enEthPHYInterface )
        {
 #if ( ETH_MACRO_ETNF == STD_ON || ETH_MACRO_ETNE == STD_ON )
        case ETH_T1S:
            /* Configure the Eth transceiver */
            Eth_InitT1s( LucCtrlCnt );
            do
            {
                /* Read the state register of the PHY(CT25205). */
                Eth_ReadMii( LucCtrlCnt, PHY_TRCV_IDX, PHY_TRCV_STATUS_REG, &LusRegValue );

                LulDelayCounter++;
            } /* Check the link up */
            while ( ( ( LusRegValue & (uint16)PHY_LINKUP_BIT ) != (uint16)PHY_LINKUP_BIT )
                 && ( LulDelayCounter < (uint32)ETH_TIMEOUT ) );

            if ( LulDelayCounter < (uint32)ETH_TIMEOUT )
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
            }
            EthCheckCount++;

            break;
 #endif /* ( ETH_MACRO_ETNF == STD_ON || ETH_MACRO_ETNE == STD_ON ) */

        default:
            /* Do nothing */
            break;
        }
    }
#endif /* ( ETH_CTRL_ENABLE_MII == STD_ON ) */

    for ( LucCtrlCnt = (uint8)0U; LucCtrlCnt < (uint8)ETH_TOTAL_CTRL_CONFIG; LucCtrlCnt++ )
    {
        /********************** Set controller MAC address **********************/

        /* Invoke Eth_SetPhysAddr to change physical address */
        Eth_SetPhysAddr( LucCtrlCnt, mac_board_addr[LucCtrlCnt] );

        /* Invoke Eth_GetPhysAddr to get physical address */
        Eth_GetPhysAddr( LucCtrlCnt, (uint8 *)LucRetrieveMacAddr[LucCtrlCnt] );

        if ( memcmp( &LucRetrieveMacAddr[LucCtrlCnt][0U], &mac_board_addr[LucCtrlCnt][0U],
SRC_MACADDR_LEN ) == 0 )
        {
            /* If the MAC address is correct */
            EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
        }
        EthCheckCount++;

        /********************** Configure frame filter ************************/
#if ( ETH_UPDATE_PHYS_ADDR_FILTER == STD_ON )
        /* Add addition multicast address for this controller to listen to */
        Eth_UpdatePhysAddrFilter( LucCtrlCnt, mac_tgt_addr0, ETH_ADD_TO_FILTER );
#endif /* ( ETH_UPDATE_PHYS_ADDR_FILTER == STD_ON ) */

        /************************** Enable controller ***************************/

        /* Set the Controller mode to ACTIVE */
        do
        {
            LucReturnValue = Eth_SetControllerMode( LucCtrlCnt, ETH_MODE_ACTIVE );
        }
        while ( (Std_ReturnType)E_NOT_OK == LucReturnValue );

        /* Get controller mode */
        Eth_GetControllerMode( LucCtrlCnt, &LenEthCtrlMode[LucCtrlCnt] );
        if ( (Eth_ModeType)ETH_MODE_ACTIVE == LenEthCtrlMode[LucCtrlCnt] )
        {
            EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
        }
        EthCheckCount++;
    }

    LoopCount = (uint32)0UL;
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
    while ( LoopCount < (uint32)0x10000000UL ) /* Waiting for other nodes to start */
    {
        LoopCount++;
    }

    /********************** Configure gPTP timer ***************************/

    /************************ Transmit and Receive a frame ******************/

    for ( LucCtrlCnt = (uint8)0U; LucCtrlCnt < (uint8)ETH_TOTAL_CTRL_CONFIG; LucCtrlCnt++ )
    {
        /* Send broadcast frame */
        SendData( LucCtrlCnt, &TxEthFrame[0U][0U], FRAME_LEN, ETH_IPV4_TYPE, 0U, mac_tgt_addr );

#if 0   /* 1: For testing Eth transmission start timing of node ID 0 when PLCA is enabled
(oscilloscope observation) */
        while ( 1U )
        {
            volatile uint32                     test_count;

            for ( test_count = 0UL; test_count < 10000UL; test_count++ );
            /* Send 1722 frames with reserved resource in this station */
            SendData( LucCtrlCnt, &TxEthFrame[1U][0U], FRAME_LEN, ETH_AVBTP_TYPE, 0U, mac_tgt_addr0 );
        }
#endif /* 0 */

#if ( ETH_GLOBAL_TIME_SUPPORT == STD_ON )
        /* Get the current time */
        Eth_GetCurrentTime( LucCtrlCnt, &LenEthTimeQualPtr, &LstCurrentTime );

        if ( ( (Eth_TimeStampQualType)ETH_VALID == LenEthTimeQualPtr )
          && ( ( LstCurrentTime.secondsHi > TxTimeStamp[LucCtrlCnt].secondsHi )
            || ( ( LstCurrentTime.secondsHi == TxTimeStamp[LucCtrlCnt].secondsHi )
              && ( LstCurrentTime.seconds > TxTimeStamp[LucCtrlCnt].seconds ) )
            || ( ( LstCurrentTime.secondsHi == TxTimeStamp[LucCtrlCnt].secondsHi )
              && ( LstCurrentTime.seconds == TxTimeStamp[LucCtrlCnt].seconds )
              && ( LstCurrentTime.nanoseconds > TxTimeStamp[LucCtrlCnt].nanoseconds ) ) ) )
        {
            EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
        }
        EthCheckCount++;
#endif /* ( ETH_GLOBAL_TIME_SUPPORT == STD_ON ) */

#if ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON )
        /* Wait to receive message */
        do
        {
            /* Check for received message */
            Eth_Receive( LucCtrlCnt,
 #if ( ETH_AR_VERSION >= ETH_AR_431_VERSION )
                         0U,
 #endif /* ( ETH_AR_VERSION >= ETH_AR_431_VERSION ) */
                         &LenEthRxStatus[LucCtrlCnt] );
        }
        while ( (Eth_RxStatusType)ETH_RECEIVED != LenEthRxStatus[LucCtrlCnt] );
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
        /*************** Transmit and Receive multiple frames ******************/

        /* Send 1722 frames with reserved resource in this station */
        SendData( LucCtrlCnt, &TxEthFrame[1U][0U], FRAME_LEN, ETH_AVBTP_TYPE, 0U, mac_tgt_addr0 );
        SendData( LucCtrlCnt, &TxEthFrame[2U][0U], FRAME_LEN, ETH_AVBTP_TYPE, 0U, mac_tgt_addr1 );

#if ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON )
        while ( GusRxFrameCnt[LucCtrlCnt] != (uint16)ETH_TX_SAMPLE_TOTAL )
        {
            Eth_MainFunction();

            /* Wait to receive message for controller ETNB0 */
 #if ( ETH_AR_VERSION >= ETH_AR_431_VERSION )
            do
            {
                /* Check for received message */
                Eth_Receive( LucCtrlCnt, RX_QUEUE_0, &LenEthStatusRxQueue0[LucCtrlCnt] );
  #if ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) )
                Eth_Receive( LucCtrlCnt, RX_QUEUE_1, &LenEthStatusRxQueue1[LucCtrlCnt] );
                Eth_Receive( LucCtrlCnt, RX_QUEUE_2, &LenEthStatusRxQueue2[LucCtrlCnt] );
                Eth_Receive( LucCtrlCnt, RX_QUEUE_3, &LenEthStatusRxQueue3[LucCtrlCnt] );
  #endif /* ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) ) */
            }
            while ( ( (Eth_RxStatusType)ETH_NOT_RECEIVED == LenEthStatusRxQueue0[LucCtrlCnt] )
  #if ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) )
                 && ( (Eth_RxStatusType)ETH_NOT_RECEIVED == LenEthStatusRxQueue1[LucCtrlCnt] )
                 && ( (Eth_RxStatusType)ETH_NOT_RECEIVED == LenEthStatusRxQueue2[LucCtrlCnt] )
                 && ( (Eth_RxStatusType)ETH_NOT_RECEIVED == LenEthStatusRxQueue3[LucCtrlCnt] )
  #endif /* ( ( ETH_MACRO_ETNB == STD_ON ) || ( ETH_MACRO_ETNF == STD_ON ) ) */
                 && ( GusRxFrameCnt[LucCtrlCnt] != (uint16)ETH_TX_SAMPLE_TOTAL ) );
 #else /* !( ETH_AR_VERSION >= ETH_AR_431_VERSION ) */
            do
            {
                /* Check for received message */
                Eth_Receive( LucCtrlCnt, &LenEthRxStatus[LucCtrlCnt] );
            }
            while ( (Eth_RxStatusType)ETH_NOT_RECEIVED == LenEthRxStatus[LucCtrlCnt] );
 #endif /* ( ETH_AR_VERSION >= ETH_AR_431_VERSION ) */
        }
#endif /* ( ETH_CTRL_ENABLE_RX_POLLING == STD_ON ) */

        /************************** Frame statistics ***************************/

#if ( ETH_GET_DROP_COUNT_API == STD_ON )
        /* Get drop packets statistic */
        do
        {
            Eth_GetDropCount( LucCtrlCnt, CountValues, DropCount );

            if ( ( (uint32)0U == DropCount[CRC_ERROR] )
              && ( (uint32)0U == DropCount[UNDERSIZE_PACKET_ERROR] )
              && ( (uint32)0U == DropCount[OVERSIZE_PACKET_ERROR] )
              && ( (uint32)0U == DropCount[ALIGNMENT_ERROR] )
              && ( (uint32)0U == DropCount[LATE_COLLISION] ) )
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
                break;
            }
        }
        while ( 1U );
        EthCheckCount++;
#endif /* ( ETH_GET_DROP_COUNT_API == STD_ON ) */

#if ( ETH_GET_COUNTER_VALUES_API == STD_ON )
        /* Get drop packets statistic */
        do
        {
            Eth_GetCounterValues( LucCtrlCnt, &CounterPtr );

            if ( ( (uint32)0U == CounterPtr.DropPktBufOverrun )
              && ( (uint32)0U == CounterPtr.DropPktCrc )
              && ( (uint32)0U == CounterPtr.UndersizePkt )
              && ( (uint32)0U == CounterPtr.OversizePkt )
              && ( (uint32)0U == CounterPtr.AlgnmtErr ) )
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
                break;
            }
        }
        while ( 1U );
        EthCheckCount++;
#endif /* ( ETH_GET_COUNTER_VALUES_API == STD_ON ) */

#if ( ETH_GET_TX_ERROR_COUNTER_VALUES_API == STD_ON )
        /* This feature is not supported */
        Eth_GetTxErrorCounterValues( LucCtrlCnt, &TxErrorCounterValues );
#endif /* ( ETH_GET_TX_ERROR_COUNTER_VALUES_API == STD_ON ) */

#if ( ETH_GET_ETHER_STATS_API == STD_ON )
        /* Get statistic */
        do
        {
            Eth_GetEtherStats( LucCtrlCnt, etherStats );

            /* Multicast packets dropped due to Eth_UpdatePhysAddrFilter is INCLUDED in statistic */
            if ( ( (uint32)0U == etherStats[DROP_EVENTS] )
              && ( (uint32)0U == etherStats[BROADCAST_PACKETS] )
              && ( (uint32)0U == etherStats[CRC_ALIGN_ERRORS] )
              && ( (uint32)0U == etherStats[UNDERSIZE_ERRORS] )
              && ( (uint32)0U == etherStats[OVERSIZE_ERRORS] )
              && ( (uint32)0U == etherStats[LATE_COLLISIONS] ) )
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
                break;
            }
        }
        while ( 1U );
        EthCheckCount++;
#endif /* ( ETH_GET_ETHER_STATS_API == STD_ON ) */
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
#if ( ETH_GET_RX_STATS_API == STD_ON )
        /* Get statistic */
        do
        {
            Eth_GetRxStats( LucCtrlCnt, &RxStats );

            if ( ( (uint32)ETH_TX_SAMPLE_TOTAL <= RxStats.RxStatsPkts )
              && ( (uint32)0U == RxStats.RxStatsCrcAlignErrors )
              && ( (uint32)0U == RxStats.RxStatsUndersizePkts )
              && ( (uint32)0U == RxStats.RxStatsOversizePkts ) )
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
                break;
            }
        }
        while ( 1U );
        EthCheckCount++;
#endif /* ( ETH_GET_RX_STATS_API == STD_ON ) */

#if ( ETH_GET_TX_STATS_API == STD_ON )
        /* Get statistic */
        do
        {
            Eth_GetTxStats( LucCtrlCnt, &TxStats );
            if ( ( (uint32)OCTET_TOTAL              == TxStats.TxNumberOfOctets )
              && ( (uint32)RX_UNICAST_FRAME_TOTAL   == TxStats.TxUniCastPkts ) )
            {
                EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
                break;
            }
        }
        while ( 1U );
        EthCheckCount++;
#endif /* ( ETH_GET_TX_STATS_API == STD_ON ) */
    }

    /* Invoke Eth_DeInit to reset process needed for the initialization of the Ethernet Driver*/
#if ( ETH_DEINIT_API == STD_ON )
    Eth_DeInit( 0U );
#endif /* ( ETH_DEINIT_API == STD_ON ) */

    /************************** Checkpoint validate ****************************/

    for ( LulInc = (uint32)0U; LulInc < EthCheckCount; LulInc++ )
    {
        if ( EthPassedCount[LulInc] != (uint8)ETH_PASSED )
        {
            sample_NG_end();
        }
    }

    sample_end();

    return (int)0;
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
} /* End of main() function */

/*****************************************************************************/
/* Send message                                                              */
/*****************************************************************************/
static void SendData
(
    uint8                               CtrlIdx,
    uint8                              *Buf,
    uint16                              LenByte,
    Eth_FrameType                       FrameType,
    uint8                               Priority,
    const uint8                        *PhysAddrPtr
)
{
    Std_ReturnType                      LucReturnValue;
    Eth_DataType                       *BufPtr;
    Eth_BufIdxType                      BufIdx;
    BufReq_ReturnType                   LenRequestBuffer;
    uint16                              LusLength;

    LusLength               = LenByte;
    GucTxConfirmed[CtrlIdx] = (uint8)0U;

    /* Invoke Eth_ProvideTxBuffer to get available buffer */
    LenRequestBuffer = Eth_ProvideTxBuffer( CtrlIdx,
#if ( ETH_AR_VERSION >= ETH_AR_431_VERSION )
                                            Priority,
#endif /* ( ETH_AR_VERSION >= ETH_AR_431_VERSION ) */
                                            &BufIdx,
                                            &BufPtr,
                                            &LusLength );

    if ( ( (BufReq_ReturnType)BUFREQ_OK == LenRequestBuffer )
      && ( Buf != NULL_PTR ) )
    {
        EthPassedCount[EthCheckCount] = (uint8)ETH_PASSED;
    }
    else
    {
#if ( ETH_CTRL_ENABLE_TX_POLLING == STD_ON )
        Eth_TxConfirmation( CtrlIdx );
#endif /* ( ETH_CTRL_ENABLE_TX_POLLING == STD_ON ) */
        return;
    }
    EthCheckCount++;

#if ( ETH_GLOBAL_TIME_SUPPORT == STD_ON )
    /* Invoke Eth_EnableEgressTimeStamp to activate egress time stamping */
    Eth_EnableEgressTimeStamp( CtrlIdx, BufIdx );
#endif /* ( ETH_GLOBAL_TIME_SUPPORT == STD_ON ) */
    /* Copy Transmit data to the buffer to transmit */
    memcpy( BufPtr, Buf, LenByte );

    do
```

File name: \node_id0\src\App_ETH_Common_Sample.c

```
    {
        LucReturnValue = Eth_Transmit( CtrlIdx, BufIdx, FrameType, ETH_TRUE, LenByte, PhysAddrPtr );
    }
    while ( (Std_ReturnType)E_NOT_OK == LucReturnValue );
#if ( ETH_CTRL_ENABLE_TX_POLLING == STD_ON )
    do
    {
        /* Polling for Tx confirmation */
        Eth_TxConfirmation( CtrlIdx );
    }
    while ( (uint8)OU == GucTxConfirmed[CtrlIdx] );
#endif /* ( ETH_CTRL_ENABLE_TX_POLLING == STD_ON ) */
}

/**********************************************************************************
**                          Notification Functions                               **
**********************************************************************************/

/**********************************************************************************
**                          End of File                                          **
**********************************************************************************/
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
/*==================================================================================*/
/* Project      = AUTOSAR Renesas U2C MCAL Components                               */
/* Module       = App_ETH_U2C4_Sample.c                                             */
/* SW-VERSION   = 1.0.0                                                             */
/*==================================================================================*/
/*                                   COPYRIGHT                                      */
/*==================================================================================*/
/* (c) 2025 Renesas Electronics Corporation. All rights reserved.                   */
/*==================================================================================*/
/* Purpose:                                                                         */
/* This file contains sample application for ETH Driver Component                   */
/*                                                                                  */
/*==================================================================================*/
/*                                                                                  */
/* Unless otherwise agreed upon in writing between your company and                 */
/* Renesas Electronics Corporation the following shall apply!                       */
/*                                                                                  */
/* Warranty Disclaimer                                                              */
/*                                                                                  */
/* There is no warranty of any kind whatsoever granted by Renesas. Any              */
/* warranty is expressly disclaimed and excluded by Renesas, either expressed       */
/* or implied, including but not limited to those for non-infringement of           */
/* intellectual property, merchantability and/or fitness for the particular         */
/* purpose.                                                                         */
/*                                                                                  */
/* Renesas shall not have any obligation to maintain, service or provide bug        */
/* fixes for the supplied Product(s) and/or the Application.                        */
/*                                                                                  */
/* Each User is solely responsible for determining the appropriateness of           */
/* using the Product(s) and assumes all risks associated with its exercise          */
/* of rights under this Agreement, including, but not limited to the risks          */
/* and costs of program errors, compliance with applicable laws, damage to          */
/* or loss of data, programs or equipment, and unavailability or                    */
/* interruption of operations.                                                      */
/*                                                                                  */
/* Limitation of Liability                                                          */
/*                                                                                  */
/* In no event shall Renesas be liable to the User for any incidental,              */
/* consequential, indirect, or punitive damage (including but not limited           */
/* to lost profits) regardless of whether such liability is based on breach         */
/* of contract, tort, strict liability, breach of warranties, failure of            */
/* essential purpose or otherwise and even if advised of the possibility of         */
/* such damages. Renesas shall not be liable for any services or products           */
/* provided by third party vendors, developers or consultants identified or         */
/* referred to the User by Renesas in connection with the Product(s) and/or         */
/* the Application.                                                                 */
/*                                                                                  */
/*==================================================================================*/
/* Environment:                                                                     */
/*              Devices:        U2C                                                 */
/*==================================================================================*/

/************************************************************************************
**                      Revision Control History                                   **
************************************************************************************/
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
/***********************************************************************************/

/***********************************************************************************
**                      Include Section                                           **
***********************************************************************************/
#include "App_ETH_Device_Sample.h"
/***********************************************************************************
**                            Macros                                              **
***********************************************************************************/
#define DISABLE_WRITE_KEY_CODE              (0xA5A5A500UL)
#define ENABLE_WRITE_KEY_CODE               (0xA5A5A501UL)

/***********************************************************************************
**                       Phy Initialization                                       **
***********************************************************************************/
#ifdef ETH_CPUCLK_MHZ
#undef ETH_CPUCLK_MHZ
#endif /* ETH_CPUCLK_MHZ */
#define ETH_CPUCLK_MHZ                      (320UL)
#define ETH_WAIT_NS( t )                                                            \
do                                                                                  \
{                                                                                   \
    volatile uint32 cnt;                                                            \
    for ( cnt = (uint32)0U;                                                         \
          cnt < ((((uint32)ETH_CPUCLK_MHZ * ((uint32)t)) / (uint32)1000U) + (uint32)1U);    \
          cnt++ );                                                                  \
}                                                                                   \
while ( 0U )

/*-------------------------------------------------------------------------------------------------------------
-------------------------*/
#define T1S_PHY_CONTROL_REG                 (0x000000U)     /* PHY Control Register[Config] */
#define T1S_PHY_STATUS_REG                  (0x000001U)     /* PHY Status Register */
#define T1S_PHY_PHYID0_REG                  (0x000002U)     /* PHY Identifier register #2 */
#define T1S_PHY_PHYID1_REG                  (0x000003U)     /* PHY Identifier register #3 */
#define T1S_PHY_DEVINPKG1_REG               (0x210005U)     /* Device in package functionality */
#define T1S_PHY_DEVINPKG2_REG               (0x210006U)     /* Device in package functionality */
#define T1S_PHY_BASET1EXT_REG               (0x210012U)     /* BASE-T1 Capability Advertisement */
#define T1S_PHY_T1SPMACTRL_REG              (0x2108F9U)     /* PMA control register[Config] */
#define T1S_PHY_T1SPMAST_REG                (0x2108FAU)     /* PMA status register */
#define T1S_PHY_T1SPMATSTM_REG              (0x2108FBU)     /* PMA test-mode register[Config] */
#define T1S_PHY_DEVINPKG1_REG3              (0x230005U)     /* Device in package functionality */
#define T1S_PHY_DEVINPKG2_REG3              (0x230006U)     /* Device in package functionality */
#define T1S_PHY_T1SPCSCTRL_REG              (0x2308F3U)     /* PCS control register[Config] */
#define T1S_PHY_T1SPCSST_REG                (0x2308F4U)     /* PCS status register */
#define T1S_PHY_T1SPCSDIAG1_REG             (0x2308F5U)     /* PCS remote jabber counter */
#define T1S_PHY_T1SPCSDIAG2_REG             (0x2308F6U)     /* PCS physical collisions counter */
#define T1S_PHY_CTIPVER_REG                 (0x3F8000U)     /* CT25205-RTL IP version */
#define T1S_PHY_T1STWEAKS_REG               (0x3F8001U)     /* Proprietary extensions register[Config]
*/
#define T1S_PHY_T1SPLCAEXT_REG              (0x3F8002U)     /* PLCA proprietary extensions
register[Config] */
#define T1S_PHY_T1SPMATUNE0_REG             (0x3F8003U)     /* PMA proprietary extensions
register[Config] */
#define T1S_PHY_T1SPMATUNE1_REG             (0x3F8004U)     /* PMA proprietary extensions
register[Config] */
#define T1S_PHY_T1STXCCTL_REG               (0x3F8005U)     /* PMA interface configuration */
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
#define T1S_PHY_T1STXCST_REG                (0x3F8006U)     /* PMA interface status */
#define T1S_PHY_PLCAIDVER_REG               (0x3FCA00U)     /* PLCA ID and version register */
#define T1S_PHY_PLCACTRL0_REG               (0x3FCA01U)     /* PLCA control register #0[Config] */
#define T1S_PHY_PLCACTRL1_REG               (0x3FCA02U)     /* PLCA node configuration
register[Config] */
#define T1S_PHY_PLCAST_REG                  (0x3FCA03U)     /* PLCA status register */
#define T1S_PHY_PLCATOTMR_REG               (0x3FCA04U)     /* PLCA TO_TIMER register[Config] */
#define T1S_PHY_PLCABURST_REG               (0x3FCA05U)     /* PLCA burst mode register [Config] */
#define T1S_PHY_PLCADIAG_REG                (0x3FCA06U)     /* PLCA Diagnostic */

#define T1S_PHY_REG000000_RESET             (0x8000U)
#define T1S_PHY_REG000000_LOOP_ON           (0x4000U)
#define T1S_PHY_REG000000_LOOP_OFF          (0x0000U)       /* (~0x4000U) */
#define T1S_PHY_REG000000_LCTL_ON           (0x1000U)
#define T1S_PHY_REG000000_LCTL_OFF          (0x0000U)       /* (~0x1000U) */
#define T1S_PHY_REG000000_LPWR_ON           (0x0800U)
#define T1S_PHY_REG000000_LPWR_OFF          (0x0000U)       /* (~0x0800U) */
#define T1S_PHY_REG000000_ISOM_ON           (0x0400U)
#define T1S_PHY_REG000000_ISOM_OFF          (0x0000U)       /* (~0x0400U) */
#define T1S_PHY_REG000000_CTEST_ON          (0x0080U)
#define T1S_PHY_REG000000_CTEST_OFF         (0x0000U)       /* (~0x0080U) */
#define T1S_PHY_REG2108F9_PMARST            (0x8000U)
#define T1S_PHY_REG2108F9_TXDIS             (0x4000U)
#define T1S_PHY_REG2108F9_LPWR_ON           (0x0800U)
#define T1S_PHY_REG2108F9_LPWR_OFF          (0x0000U)       /* (~0x0800U) */
#define T1S_PHY_REG2108F9_LOOP_ON           (0x0001U)
#define T1S_PHY_REG2108F9_LOOP_OFF          (0x0000U)       /* (~0x0001U) */
#define T1S_PHY_REG2108FB_NORMAL            (0U<<13U)
#define T1S_PHY_REG2108FB_TESTM1            (1U<<13U)
#define T1S_PHY_REG2108FB_TESTM2            (2U<<13U)
#define T1S_PHY_REG2108FB_TESTM3            (3U<<13U)
#define T1S_PHY_REG2108FB_TESTM4            (4U<<13U)
#define T1S_PHY_REG2308F3_PCSRST            (0x8000U)
#define T1S_PHY_REG2308F3_LOOP_ON           (0x4000U)
#define T1S_PHY_REG2308F3_LOOP_OFF          (0x0000U)       /* (~0x4000U) */
#define T1S_PHY_REG3F8001_PKTLOOP_ON        (0x8000U)
#define T1S_PHY_REG3F8001_PKTLOOP_OFF       (0x0000U)       /* (~0x8000U) */
#define T1S_PHY_REG3F8001_ENIE_ON           (0x0080U)
#define T1S_PHY_REG3F8001_ENIE_OFF          (0x0000U)       /* (~0x0080U) */
#define T1S_PHY_REG3F8001_UNJT_ON           (0x0040U)
#define T1S_PHY_REG3F8001_UNJT_OFF          (0x0000U)       /* (~0x0040U) */
#define T1S_PHY_REG3F8001_RXNRZ_ON          (0x0008U)
#define T1S_PHY_REG3F8001_RXNRZ_OFF         (0x0000U)       /* (~0x0008U) */
#define T1S_PHY_REG3F8001_SCRD_ON           (0x0004U)
#define T1S_PHY_REG3F8001_SCRD_OFF          (0x0000U)       /* (~0x0004U) */
#define T1S_PHY_REG3F8001_NCOLM_ON          (0x0002U)
#define T1S_PHY_REG3F8001_NCOLM_OFF         (0x0000U)       /* (~0x0002U) */
#define T1S_PHY_REG3F8001_RXDLY_ON          (0x0001U)
#define T1S_PHY_REG3F8001_RXDLY_OFF         (0x0000U)       /* (~0x0001U) */
#define T1S_PHY_REG3F8002_PREN_ON           (0x8000U)
#define T1S_PHY_REG3F8002_PREN_OFF          (0x0000U)       /* (~0x8000U) */
#define T1S_PHY_REG3F8002_LDEN_ON           (0x0002U)
#define T1S_PHY_REG3F8002_LDEN_OFF          (0x0000U)       /* (~0x0002U) */
#define T1S_PHY_REG3F8002_LDR_ON            (0x0001U)
#define T1S_PHY_REG3F8002_LDR_OFF           (0x0000U)       /* (~0x0001U) */
#define T1S_PHY_REG3F8003_NNTHR             (0x2000U)
#define T1S_PHY_REG3F8003_DRFTW             (4U)            /* 2～4 */
#define T1S_PHY_REG3F8004_JJHHTHR           (0x3300U)
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
#define T1S_PHY_REG3F8004_JJTHR             (0x20U)
#define T1S_PHY_REG3FCA01_EN_ON             (0x8000U)       /* #define EthTrcvEnablePLCA */
#define T1S_PHY_REG3FCA01_EN_OFF            (0x0000U)       /* (~0x8000U) #define EthTrcvEnablePLCA */
#define T1S_PHY_REG3FCA01_RST               (0x4000U)
#define T1S_PHY_REG3FCA02_NCNT              (0x0800U)       /* #define EthTrcvPhysLayerPlcaLocalNodeId
*/
#define T1S_PHY_REG3FCA02_ID                (0U)            /* #define EthTrcvPhysLayerPlcaLocalNodeId
*/
#define T1S_PHY_REG3FCA04_TOTMR             (0x20U)
#define T1S_PHY_REG3FCA05_MAXBC             (0x00U<<8U)
#define T1S_PHY_REG3FCA05_BTMR              (0x80U)


#define ETH_CHECKPOINT_TOTAL                (40U)
#define ETH_FAILED                          (0)
#define ETH_PASSED                          (1)
#define ETH_NC_TYPE                         (0x88F7U)   /* Precision Time Protocol (PTP) over Ethernet
(IEEE 1588) */
#define PHY_TRCV_CTRL_REG                   (0U)
#define PHY_TRCV_STATUS_REG                 (1U)
#define PHY_TRCV_ID1_REG                    (2U)        /* PHY Identifier 1 (0x0141 is set for
88E1112) */
#define FRAME_LEN                           (62U)       /* Without header and FCS.    */
#define FRAME_LEN_MTU                       (1500U)     /* Maximum payload size */
#define APP_ETH_MACADDR_LEN                 (6U)
#define STREAMID_LEN                        (8U)
#define ETHTYPE_LEN                         (2U)
#define PHY_LINKUP_BIT                      (0x002CU)

#define RX_QUEUE_0                          (0U)
#define RX_QUEUE_1                          (1U)
#define RX_QUEUE_2                          (2U)
#define RX_QUEUE_3                          (3U)
/* Time-out */
#define ETH_TIMEOUT                         ((uint32)0x200000)

#define BUFFER_OVERRUN_ERROR                (1U)
#define CRC_ERROR                           (2U)
#define UNDERSIZE_PACKET_ERROR              (3U)
#define OVERSIZE_PACKET_ERROR               (4U)
#define ALIGNMENT_ERROR                     (5U)

#define ALL_PACKETS                         (3U)
#define BROADCAST_PACKETS                   (4U)
#define MULTICAST_PACKETS                   (5U)
#define CRC_ALIGN_ERRORS                    (6U)
#define UNDERSIZE_ERRORS                    (7U)
#define OVERSIZE_ERRORS                     (8U)
/*-------------------------------------------------------------------------------------------------
-------------------------*/

/*-------------------------------------------------------------------------------------------------
-------------------------*/
/* User configuration
*/
/*-------------------------------------------------------------------------------------------------
-------------------------*/
#define T1S_PHY_ADDR                        (0x01U) /* トランシーバの PHY アドレスを指定 */
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
#define ETH_PHY_ADDR                        (0x1FU) /* 内蔵 PHY アドレスを指定 */

#define T1S_PHY_CONTROL_CFG                 /* CONTROL レジスタのコンフィグ設定 */              \
(                                                                                               \
        T1S_PHY_REG000000_LOOP_OFF        | /* ループバックしない */                           \
        T1S_PHY_REG000000_LCTL_ON         | /* PHY リンク制御を有効: 通常動作開始 */           \
        T1S_PHY_REG000000_LPWR_OFF        | /* 低電力モードに入らない */                       \
        T1S_PHY_REG000000_ISOM_OFF        | /* 分離モードに入らない */                         \
        T1S_PHY_REG000000_CTEST_OFF         /* 衝突実行モード無効: 通常動作 */                 \
)
#define T1S_PHY_T1SPMACTRL_CFG              /* T1SPMACTRL レジスタのコンフィグ設定 */           \
(                                                                                               \
        T1S_PHY_REG2108F9_LPWR_OFF        | /* 低電力モードにしない */                         \
        T1S_PHY_REG2108F9_LOOP_OFF          /* ループバックモードにしない */                   \
)
#define T1S_PHY_T1SPMATSTM_CFG              /* T1SPMATSTM レジスタのコンフィグ設定 */           \
(                                                                                               \
        T1S_PHY_REG2108FB_NORMAL            /* Normal operation: 通常動作 */                   \
)
#define T1S_PHY_T1SPCSCTRL_CFG              /* T1SPCSCTRL レジスタのコンフィグ設定 */           \
(                                                                                               \
        T1S_PHY_REG2308F3_LOOP_OFF          /* ループバックにしない */                         \
)
#define T1S_PHY_T1STWEAKS_CFG               /* T1STWEAKS レジスタのコンフィグ設定 */            \
(                                                                                               \
        T1S_PHY_REG3F8001_PKTLOOP_OFF     | /* TX フレームが MII RX に反映されない */          \
        T1S_PHY_REG3F8001_ENIE_OFF        | /* 拡張ノイズ耐性モードにしない */                 \
        T1S_PHY_REG3F8001_UNJT_OFF        | /* 自動回復しない */                               \
        T1S_PHY_REG3F8001_RXNRZ_OFF       | /* AFE RX 信号は RZ エンコードされていると見なされる */\
        T1S_PHY_REG3F8001_SCRD_OFF        | /* PCS スクランブルおよびデスクランブル機能が無効 */ \
        T1S_PHY_REG3F8001_NCOLM_ON        | /* 衝突検出はマスクされない */                     \
        T1S_PHY_REG3F8001_RXDLY_ON          /* 同時通信を回避するために MII RX 信号が遅延される */\
)
#define T1S_PHY_T1SPLCAEXT_CFG              /* T1SPLCAEXT レジスタのコンフィグ設定 */           \
(                                                                                               \
        T1S_PHY_REG3F8002_PREN_OFF        | /* PLCA RS は標準モードで動作 */                   \
        T1S_PHY_REG3F8002_LDEN_OFF        | /* PLCA リーダはノード ID によって選択 */           \
        T1S_PHY_REG3F8002_LDR_OFF           /* LDEN=1 であれば、PLCA スレーブとする */         \
)
#define T1S_PHY_T1SPMATUNE0_CFG             /* T1SPMATUNE0 レジスタのコンフィグ設定 */          \
(                                                                                               \
        T1S_PHY_REG3F8003_NNTHR           | /* NN コンパレータのしきい値を設定 */              \
        T1S_PHY_REG3F8003_DRFTW             /* ドリフト補償器の時間ウィンドウを設定 */         \
)
#define T1S_PHY_T1SPMATUNE1_CFG             /* T1SPMATUNE1 レジスタのコンフィグ設定 */          \
(                                                                                               \
        T1S_PHY_REG3F8004_JJHHTHR         | /* JJHH コンパレータのしきい値を設定 */            \
        T1S_PHY_REG3F8004_JJTHR             /* JJ コンパレータのしきい値を設定 */              \
)
#define T1S_PHY_PLCACTRL0_CFG               /* PLCACTRL0 レジスタのコンフィグ設定 */            \
(                                                                                               \
        T1S_PHY_REG3FCA01_EN_ON             /* PLCA RS 機能が有効 */                           \
)
#define T1S_PHY_PLCACTRL1_CFG               /* PLCACTRL1 レジスタのコンフィグ設定 */            \
(                                                                                               \
        T1S_PHY_REG3FCA02_NCNT            | /* ノードの最大数を設定 */                         \
        T1S_PHY_REG3FCA02_ID                /* PLCA ノード ID を設定 */                        \
)
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
#define T1S_PHY_PLCATOTMR_CFG               /* PLCATOTMR レジスタのコンフィグ設定 */            \
(                                                                                           \
        T1S_PHY_REG3FCA04_TOTMR             /* PLCA 送信機会タイマを設定 */                    \
)
#define T1S_PHY_PLCABURST_CFG               /* PLCABURST レジスタのコンフィグ設定 */            \
(                                                                                           \
        T1S_PHY_REG3FCA05_MAXBC           | /* 送信機会内に送信できる追加のパケット数を設定 */  \
        T1S_PHY_REG3FCA05_BTMR              /* バーストに連結するのを待機する時間を設定 */      \
)

#define T1S_PHY_ADDR1_REG18H_SET_CFG        (0x6000U)   /* XCVR_ANALOG_SET_2, Set [14:13] -> 11B */
#define T1S_PHY_ADDR1_REG18H_CLR_CFG        (~0x0000U)
#define T1S_PHY_ADDR1_REG1FH_SET_CFG        (0x0018U)   /* XCVR_ANALOG_SET_9, Set [4:3] -> 11B */
#define T1S_PHY_ADDR1_REG1FH_CLR_CFG        (~0x0000U)

/*****************************************************************************
**                          Global Data                                     **
*****************************************************************************/

/*****************************************************************************
**                       Global Symbols                                     **
*****************************************************************************/

/*****************************************************************************
**                        Global variables                                  **
*****************************************************************************/

static const uint32 GaaPcrRegAddr[] =
{
    REG_PCR( 20,  4 ),          /* P20_4  (ALT_OUT3         - ETSO_TX)      */
    REG_PCR( 20,  3 ),          /* P20_3  (ALT_BI3          - ETSO_RX_MDC)  */
    REG_PCR( 20, 13 ),          /* P20_13 (ALT_IN4/ALT_OUT4 - ETSO_ED_MDIO) */

    (uint32)NULL_PTR
};

static const uint32 GaaPcrRegValue[] =
{
    PCR_AF3_OUT_SOFTIOCNT,      /* P20_4  (ALT_OUT3         - ETSO_TX)      */
    PCR_AF3_BI_SOFTIOCNT,       /* P20_3  (ALT_BI3          - ETSO_RX_MDC)  */
    PCR_AF4_PIPC_SOFTIOCNT,     /* P20_13 (ALT_IN4/ALT_OUT4 - ETSO_ED_MDIO) */

    (uint32)NULL_PTR
};

/*****************************************************************************
**                       ISR Defines                                        **
*****************************************************************************/

/*****************************************************************************
**                       Mcu Initialization                                 **
*****************************************************************************/
void Mcu_Init ( void )
{
    /*-----------------------------------------------------------*/
    /* Interrupt Controller                                      */
    /*-----------------------------------------------------------*/
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    /* Following settings are for AVBT1S. */
    EIC536 |= (uint16)( 1U << 6U );
    EIC537 |= (uint16)( 1U << 6U );
    EIC538 |= (uint16)( 1U << 6U );
    EIC539 |= (uint16)( 1U << 6U );

#ifdef RUN_OTHER_PE
    EIBD708 &= (uint32)0xFFFFFFF8UL;
    EIBD708 |= (uint32)0x00000002UL;
    EIBD709 &= (uint32)0xFFFFFFF8UL;
    EIBD709 |= (uint32)0x00000002UL;
    EIBD710 &= (uint32)0xFFFFFFF8UL;
    EIBD710 |= (uint32)0x00000002UL;
    EIBD711 &= (uint32)0xFFFFFFF8UL;
    EIBD711 |= (uint32)0x00000002UL;
    EIBD712 &= (uint32)0xFFFFFFF8UL;
    EIBD712 |= (uint32)0x00000002UL;
    EIBD713 &= (uint32)0xFFFFFFF8UL;
    EIBD713 |= (uint32)0x00000002UL;
    EIBD714 &= (uint32)0xFFFFFFF8UL;
    EIBD714 |= (uint32)0x00000002UL;
    EIBD715 &= (uint32)0xFFFFFFF8UL;
    EIBD715 |= (uint32)0x00000002UL;
#endif /* RUN_OTHER_PE */
    /*****************************************************************/
    /* Standby Controller                                            */
    /*****************************************************************/
    /* Release the write protection of Standby controller register.*/
    MSRKCPROT = (uint32)ENABLE_WRITE_KEY_CODE;

    /* Enable clocks supplied for ES0, ES1 */
    MSR_ETN_ETNF &= (uint32)~0x00000003UL;
    while ( MSR_ETN_ETNF != (uint32)0U )
    {
        /* Do nothing */
    }

    /* Enable clocks supplied for ES0 */
    MSR_ETN_ETND &= (uint32)~0x00000003UL;
    while ( MSR_ETN_ETND != (uint32)0U )
    {
        /* Do nothing */
    }

    /* Set the write protection of Standby controller registers.*/
    MSRKCPROT = (uint32)DISABLE_WRITE_KEY_CODE;

    /* Enable interrupts */
    ENABLE_INTERRUPT();
}

/*****************************************************************************
**                      System Initialization                               **
*****************************************************************************/
void Clock_Init ( void )
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
{
    /**********************/
    /* LSIntOSC(240[KHz]) */
    /**********************/
    /* (After power supply the LSIntOSC starts operation. It cannot be stopped.) */

    /**********************/
    /* HSIntOSC(200[MHz]) */
    /**********************/
    /* (After Power On Reset or System Reset1 release the HSIntOSC starts operation.) */

    /* Wait to HSIntOSC clock is stable( HSOSCS.HSOSCSTAB = 1 ). */
    while ( ( HSOSCS & (uint32)0x00000002UL ) != (uint32)0x00000002UL )
    {
        /* Do nothing */
    }

    /* HSIntOSC stops operation in stand-by mode( HSOSCSTPM.MOSCSTPMSK = 0 ). */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;
    HSOSCSTPM   = (uint32)0x00000000UL;
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

    /***********/
    /* MainOSC */
    /***********/
    /* Start the MainOSC ( MOSCE.MOSCENTRG = 1 ). */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;
    MOSCE       = (uint32)0x00000001UL;
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

    /* Confirm that the MainOSC has been started ( MOSCS.MOSCSTAB = 1 ). */
    while ( ( MOSCS & (uint32)0x00000002UL ) != (uint32)0x00000002UL )
    {
        /* Do nothing */
    }

    /* MainOSC stops operation in stand-by mode ( MOSCSTPM.MOSCSTPMSK = 0 ). */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;
    MOSCSTPM    = (uint32)0x00000000UL;
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

    /************/
    /* PLL/SSCG */
    /************/
    /* Start the PLL ( PLLE.PLLENTRG = 1 ). */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;
    PLLE        = (uint32)0x00000001UL;
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

    /* Confirm that the PLL has been started ( PLLS.PLLCLKSTAB = 1 ). */
    while ( ( PLLS & (uint32)0x00000002UL ) != (uint32)0x00000002UL )
    {
        /* Do nothing */
    }
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    /* PLL stops operation in stand-by mode ( PLLSTPM.PLLSTPMSK = 0 ) */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;
    PLLSTPM     = (uint32)0x00000000UL;
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

    /* Release the write protection of Clock controller register */
    CLKKCPROT1  = (uint32)ENABLE_WRITE_KEY_CODE;

    /* Division ratio of clock source PLL is changed from 1 to 3/8 */
    CKD_PLLC    = (uint32)0x6U; /* Write 0110B in CKD_PLLC.PLLCLKDCSID */
    (void)CKD_PLLC;
    CKD_SSCGC   = (uint32)0x6U; /* Write 0110B in CKD_SSCGC.SSCGCLKDCSID */
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();

    /* -- Divider clock synchronized for PLL -- */
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )   /* Read CKD_PLLS and verify that the value
of PLLCLKDSYNC is 1B */
    {
        /* Do nothing */
    }

    /* The clock source for the System clock is changed from CLK_IOSC to CLK_PLL0 */
    CKS_CLEANC  = (uint32)0x0U;
    (void)CKS_CLEANC;
    EXECUTE_SYNCP();

    /* Read CKS_CLEANS and verify that the value of SYSCLKSACT is 0B */
    while ( ( CKS_CLEANS & (uint32)0x1U ) != (uint32)0x0U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */

    /* Start of repetitions: 5 repetitions (800MHz) */
    /* Division ratio of clock source PLL is changed from 3/8 to 4/8 */
    CKD_PLLC    = (uint32)0x8U;
    (void)CKD_PLLC;
    EXECUTE_SYNCP();
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source PLL is changed from 4/8 to 5/8 */
    CKD_PLLC    = (uint32)0xAU;
    (void)CKD_PLLC;
    EXECUTE_SYNCP();
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source PLL is changed from 5/8 to 6/8 */
    CKD_PLLC    = (uint32)0xCU;
    (void)CKD_PLLC;
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    EXECUTE_SYNCP();
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source PLL is changed from 6/8 to 7/8 */
    CKD_PLLC    = (uint32)0xEU;
    (void)CKD_PLLC;
    EXECUTE_SYNCP();
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source PLL is changed from 7/8 to 1 */
    CKD_PLLC    = (uint32)0x0U;
    (void)CKD_PLLC;
    EXECUTE_SYNCP();
    while ( ( CKD_PLLS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* End of repetitions for PLL */

    /* -- Divider clock synchronized for SSCG -- */
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )  /* Read CKD_SSCGS and verify that the
value of SSCGCLKDSYNC is 1B */
    {
        /* Do nothing */
    }

    /* The clock source for the System clock is changed from CLK_IOSC to CLK_PLL0 */
    CKS_SSCGC   = (uint32)0x0U;
    (void)CKS_SSCGC;
    EXECUTE_SYNCP();

    /* Read CKS_SSCGS and verify that the value of SYSCLKSACT is 0B */
    while ( ( CKS_SSCGS & (uint32)0x1U ) != (uint32)0x0U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */

    /* Start of repetitions: 5 repetitions (640MHz) */
    /* Division ratio of clock source SSCG is changed from 3/8 to 4/8 */
    CKD_SSCGC   = (uint32)0x8U;
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source SSCG is changed from 4/8 to 5/8 */
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    CKD_SSCGC   = (uint32)0xAU;
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source SSCG is changed from 5/8 to 6/8 */
    CKD_SSCGC   = (uint32)0xCU;
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source SSCG is changed from 6/8 to 7/8 */
    CKD_SSCGC   = (uint32)0xEU;
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* Division ratio of clock source SSCG is changed from 7/8 to 1 */
    CKD_SSCGC   = (uint32)0x0U;
    (void)CKD_SSCGC;
    EXECUTE_SYNCP();
    while ( ( CKD_SSCGS & (uint32)0x2U ) != (uint32)0x2U )
    {
        /* Do nothing */
    }
    ETH_WAIT_NS( 100U * 1000U ); /* 100us */
    /* End of repetitions for SSCG */

    /* Set the write protection of Clock controller */
    CLKKCPROT1  = (uint32)DISABLE_WRITE_KEY_CODE;

#if 0   /* 1: For testing CPU clock frequency operation */
    while ( 1U ) /* Instruction cache effect from the second round onwards */
    {
        ASM_NOP320
    }
#endif /* 0 */
}

/*******************************************************************************
**                      Wdg Initialization                                    **
*******************************************************************************/
void Wdg_Init ( void )
{
    /* Do nothing */
}
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
/*****************************************************************************
**                      Port Initialization                                 **
*****************************************************************************/
void Port_Init ( void )
{
    volatile uint32                    LulCount;
    volatile uint32                   *LpPcrReg;
    volatile uint16                   *addr;
    volatile uint16                    val;

    addr        = (volatile uint16 *)0U;
    val         = (uint16)0U;
    LulCount    = (uint32)0UL;

    /* Register Protection Disable */
    REG_PKCPROT = (uint32)KCPROT_SET;

    /* Set Port Write Enable Register */
    REG_PWE     = (uint32)0x05F1FDDDUL;

    do
    {
        LpPcrReg    = (volatile uint32 *)( GaaPcrRegAddr[LulCount] );
        *LpPcrReg   = ( (uint32)*LpPcrReg & (uint32)( ~PCR_MASK ) ) | GaaPcrRegValue[LulCount];
        LulCount++;
    }
    while ( GaaPcrRegAddr[LulCount] != (uint32)NULL_PTR );

    /*=====================================================*/
    /* RESET of Ether PHY asserted and de-asserted to reset */
    /*=====================================================*/
    /* ETS0_RX_MDC(P20_3=HIGH) */
    addr    = (volatile uint16 *)REG_P( 20 );
    val     = *addr;
    val    |= (uint16)0x0008U;
    *addr   = val;

    /* ETS0_TX(P20_4=HIGH) */
    addr    = (volatile uint16 *)REG_P( 20 );
    val     = *addr;
    val    |= (uint16)0x0010U;
    *addr   = val;

    /* ETS0_ED_MDIO(P20_13=HIGH) */
    addr    = (volatile uint16 *)REG_P( 20 );
    val     = *addr;
    val    |= (uint16)0x2000U;
    *addr   = val;

    ETH_WAIT_NS( 100U * 1000U ); /* 100us */

    /* Register Protection Enable */
    REG_PWE     = (uint32)0x00000000UL;
    REG_PKCPROT = (uint32)KCPROT_CLR;
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
}

/*****************************************************************************
** Function:    R_ETNF_ConfigPLCA
** Description: Configure ETNF Module PLCA.
** Parameter:   Ethernet Controller Configure ID.
** Return:      None.
*****************************************************************************/
void R_ETNF_ConfigPLCA ( uint8 LucCtrlIdx )
{
    uint16                          vs;

    ETNF0T1SCTL0 =  ( 0x00FEFFE0UL) |   /* Reserved: write the value after reset */
                    ( 1UL << 24U )  |   /* Configures the PMA: Handle a filtered */
                    ( 1UL << 16U )  |   /* Configures the MDIO interface: use ETS0_RX_MDC,
ETS0_ED_MDIO */
                    ( ETH_PHY_ADDR );   /* PHY MDIO address */

    /* Soft reset of the built-in PHY */
    Eth_WriteMii( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_CONTROL_REG, T1S_PHY_REG000000_RESET );

    /*-------------------------------------------------------------------------------------------*/
    /* Check content of some T1S-PHY register                                      (content) */
    /*-------------------------------------------------------------------------------------------*/
    Eth_ReadMii ( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_CONTROL_REG,    &vs     );  /* -> 0x0000 */
    Eth_ReadMii ( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_STATUS_REG,     &vs     );  /* -> 0x0809 */
    Eth_ReadMii ( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_PHYID0_REG,     &vs     );  /* -> 0xB824 */
    Eth_ReadMii ( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_PHYID1_REG,     &vs     );  /* -> 0x2B01 */
    vs = regRead( LucCtrlIdx,               T1S_PHY_CTIPVER_REG             );  /* -> 0x20C2 */

    /*-------------------------------------------------------------------------------------------*/
    /* Setup T1S-PHY                                                                             */
    /*-------------------------------------------------------------------------------------------*/
    /* Configuration for CT25205-specific features */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1STWEAKS_REG,  T1S_PHY_T1STWEAKS_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1STWEAKS_REG           );

    /* Set Control register, as specified in Clause 22.2.4.1 of the IEEE standard for Ethernet. */
    Eth_WriteMii( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_CONTROL_REG,    T1S_PHY_CONTROL_CFG );
    Eth_ReadMii ( LucCtrlIdx, ETH_PHY_ADDR, T1S_PHY_CONTROL_REG,    &vs     );

    /*-------------------------------------------------------------------------------------------*/
    /* Setup PLCA                                                                  (content) */
    /*-------------------------------------------------------------------------------------------*/
    /* Configuration for CT25205-specific PLCA features */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPLCAEXT_REG, T1S_PHY_T1SPLCAEXT_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPLCAEXT_REG          );

    /* Set PLCA node configuration register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_PLCACTRL1_REG,  T1S_PHY_PLCACTRL1_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_PLCACTRL1_REG           );

    /* Set PLCA transmit timer configuration register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_PLCATOTMR_REG,  T1S_PHY_PLCATOTMR_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_PLCATOTMR_REG           );
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    /* Set PLCA burst mode register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_PLCABURST_REG,  T1S_PHY_PLCABURST_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_PLCABURST_REG           );

    vs = regRead( LucCtrlIdx,               T1S_PHY_PLCAST_REG              );  /* -> 0x0000 */

    /*-----------------------------------------------------------------------------------------*/
    /* Setup PMA/PCS                                                                           */
    /*-----------------------------------------------------------------------------------------*/
    /* Setting this bit causes the PCS and PMA PHY layers to reset. */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPMACTRL_REG, T1S_PHY_REG2108F9_PMARST );
    do
    {
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1SPMACTRL_REG          );
    }
    while ( ( vs & (uint16)T1S_PHY_REG2108F9_PMARST ) != (uint16)0x0000U );

    /* Setting this bit causes the PCS and PMA PHY layers to reset. */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPCSCTRL_REG,  T1S_PHY_REG2308F3_PCSRST );
    do
    {
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1SPCSCTRL_REG          );
    }
    while ( ( vs & (uint16)T1S_PHY_REG2308F3_PCSRST ) != (uint16)0x0000U );

    /* Configuration for CT25205-specific PMA features. */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPMATUNE0_REG, T1S_PHY_T1SPMATUNE0_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPMATUNE0_REG         );

    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPMATUNE1_REG, T1S_PHY_T1SPMATUNE1_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPMATUNE1_REG         );

    /* Set PMA Control register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPMACTRL_REG,  T1S_PHY_T1SPMACTRL_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPMACTRL_REG          );

    /* Set PMA test-mode register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPMATSTM_REG,  T1S_PHY_T1SPMATSTM_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPMATSTM_REG          );

    /* Set PCS Control register */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1SPCSCTRL_REG,  T1S_PHY_T1SPCSCTRL_CFG );
    vs = regRead( LucCtrlIdx,               T1S_PHY_T1SPCSCTRL_REG          );
}

/***********************************************************************************
** Function:    R_ETNF_Config_LAN8680_T1S
** Description: Configure LAN8680 T1S Transceiver (PHY).
** Parameter:   Ethernet Controller Configure ID.
** Return:      None.
***********************************************************************************/
void R_ETNF_Config_LAN8680_T1S ( uint8 LucCtrlIdx )
{
    uint16                          vs;
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
    /*-------------------------------------------------------------------------------------------*/
    /* Configure LAN8680 T1S Transceiver (PHY)                                    (content) */
    /*-------------------------------------------------------------------------------------------*/
    do
    {
        /* Checking status of the attached transceiver */
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1STXCST_REG            );
    }
    while ( ( vs & (uint16)0x0007U ) != (uint16)0x0000U );

    /* Set Config Mode (PHY CONFIG Request) */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1STXCCTL_REG,  0x0002U );

    /* Setup PHY on Address 1 */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, T1S_PHY_PHYID0_REG,     &vs     );  /* -> 0x0007 */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, T1S_PHY_PHYID1_REG,     &vs     );  /* -> 0xC200 */

    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x000018U,              &vs     );  /* XCVR_ANALOG_SET_2
*/
    vs = ( vs & (uint16)T1S_PHY_ADDR1_REG18H_CLR_CFG ) | (uint16)T1S_PHY_ADDR1_REG18H_SET_CFG;
    Eth_WriteMii( LucCtrlIdx, T1S_PHY_ADDR, 0x000018U,              vs      );

    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x00001FU,              &vs     );  /* XCVR_ANALOG_SET_9
*/
    vs = ( vs & (uint16)T1S_PHY_ADDR1_REG1FH_CLR_CFG ) | (uint16)T1S_PHY_ADDR1_REG1FH_SET_CFG;
    Eth_WriteMii( LucCtrlIdx, T1S_PHY_ADDR, 0x00001FU,              vs      );

    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x000018U,              &vs     );  /* -> [14:13] = 11B */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x00001FU,              &vs     );  /* -> [04:03] = 11B */

    /* Set Normal Mode (PHY RESET/NORMAL Request) */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1STXCCTL_REG,  0x0000U );
    /* Check TXCST status NORMAL */
    do
    {
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1STXCST_REG            );
    }
    while ( ( vs & (uint16)0x0007U ) != (uint16)0x0000U );

    /*-------------*/
    /* Set PLCA EN */
    /*-------------*/
    /* Setting PLCA control register #0 */
    regWrite    ( LucCtrlIdx,               T1S_PHY_PLCACTRL0_REG,  T1S_PHY_PLCACTRL0_CFG );
    do
    {
        vs = regRead ( LucCtrlIdx,          T1S_PHY_PLCACTRL0_REG           );  /* -> 0x0000/0x8000 */
    }
    while ( ( vs & (uint16)T1S_PHY_REG3FCA01_EN_ON ) != ( (uint16)T1S_PHY_PLCACTRL0_CFG &
(uint16)T1S_PHY_REG3FCA01_EN_ON ) );
}

/**************************************************************************
** Function:    R_ETNF_Config_NCV7310_T1S
** Description: Configure NCV7310 T1S Transceiver (PHY).
** Parameter:   Ethernet Controller Configure ID.
** Return:      None.
**************************************************************************/
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
void R_ETNF_Config_NCV7310_T1S ( uint8 LucCtrlIdx )
{
    uint16                              vs;

    /*-----------------------------------------------------------------------------------------*/
    /* Configure NCV7310 T1S Transceiver (PHY)                                    (content) */
    /*-----------------------------------------------------------------------------------------*/
    do
    {
        /* Checking status of the attached transceiver */
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1STXCST_REG            );
    }
    while ( ( vs & (uint16)0x0007U ) != (uint16)0x0000U );

    /* Set Config Mode (PHY CONFIG Request) */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1STXCCTL_REG,  0x0002U );

    /* Setup PHY on Address 1 */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, T1S_PHY_PHYID0_REG,     &vs     );  /* -> 0x180F */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, T1S_PHY_PHYID1_REG,     &vs     );  /* -> 0xF411 */

    Eth_WriteMii( LucCtrlIdx, T1S_PHY_ADDR, 0x000012U,              0x1830U );  /* TWEAKS1  , Disable
ED col detection algorithm */
    Eth_WriteMii( LucCtrlIdx, T1S_PHY_ADDR, 0x000011U,              0x2330U );  /* TWEAKS0  , Set fast
transmit edges */

    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x000012U,              &vs     );  /* -> 0x1830 */
    Eth_ReadMii ( LucCtrlIdx, T1S_PHY_ADDR, 0x000011U,              &vs     );  /* -> 0x2330 */

    /* Set Normal Mode (PHY RESET/NORMAL Request) */
    regWrite    ( LucCtrlIdx,               T1S_PHY_T1STXCCTL_REG,  0x0000U );
    /* Check TXCST status NORMAL */
    do
    {
        vs = regRead( LucCtrlIdx,           T1S_PHY_T1STXCST_REG            );
    }
    while ( ( vs & (uint16)0x0007U ) != (uint16)0x0000U );

    /*-------------*/
    /* Set PLCA EN */
    /*-------------*/
    /* Setting PLCA control register #0 */
    regWrite    ( LucCtrlIdx,               T1S_PHY_PLCACTRL0_REG,  T1S_PHY_PLCACTRL0_CFG );
    do
    {
        vs = regRead ( LucCtrlIdx,          T1S_PHY_PLCACTRL0_REG           );  /* -> 0x0000/0x8000 */
    }
    while ( ( vs & (uint16)T1S_PHY_REG3FCA01_EN_ON ) != ( (uint16)T1S_PHY_PLCACTRL0_CFG &
(uint16)T1S_PHY_REG3FCA01_EN_ON ) );

}

/***********************************************************************************
** Function:    Eth_InitT1s
** Description: Initialize ETNF Module and T1S Transceiver (PHY).
** Parameter:   Ethernet Controller Configure ID.
** Return:      None.
***********************************************************************************/
```

File name: \node_id0\U2C4\src\App_ETH_U2C4_Sample.c

```
void Eth_InitT1s ( uint8 LucCtrlIdx )
{
    /* Configure PLCA */
    R_ETNF_ConfigPLCA ( LucCtrlIdx );

    /* Configure LAN8680 T1S Transceiver (PHY) */
    R_ETNF_Config_LAN8680_T1S ( LucCtrlIdx );
}

/*******************************************************************************/
/* Write PHY's register follow clause 45.                                      */
/*******************************************************************************/
void regWrite ( uint8 LucCtrlIdx, uint32 regAddr45, uint16 reg_val )
{
    uint16                              mmd;
    uint16                              reg_addr;

    mmd      = ( (uint16)( regAddr45 >> 16U ) & (uint16)0x001FU ); /* MMD1 to MMD31, so 1FH */
    reg_addr = ( (uint16)( regAddr45 >>  0U ) & (uint16)0xFFFFU );
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 13U, 0x0000U | mmd ); /* MMD */
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 14U, reg_addr      );
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 13U, 0x4000U | mmd ); /* MMD + SEL_DATA */
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 14U, reg_val       );
}

/*******************************************************************************/
/* Read PHY's register follow clause 45.                                       */
/*******************************************************************************/
uint16 regRead ( uint8 LucCtrlIdx, uint32 regAddr45 )
{
    uint16                              mmd;
    uint16                              reg_addr;
    uint16                              reg_val;

    mmd      = ( (uint16)( regAddr45 >> 16U ) & (uint16)0x001FU ); /* MMD1 to MMD31, so 1FH */
    reg_addr = ( (uint16)( regAddr45 >>  0U ) & (uint16)0xFFFFU );
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 13U, 0x0000U | mmd ); /* MMD */
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 14U, reg_addr      );
    Eth_WriteMii ( LucCtrlIdx, ETH_PHY_ADDR, 13U, 0x4000U | mmd ); /* MMD + SEL_DATA */
    Eth_ReadMii  ( LucCtrlIdx, ETH_PHY_ADDR, 14U, &reg_val      );

    return reg_val;
}

/*******************************************************************************
**                          End of File                                       **
*******************************************************************************/
```

Revision Record

Rev.	Issue date	Revised contents	
		Page	Point
1.00	May 5, 2025	-	First edition
1.10	Oct 29, 2025	-	Add U2B-E Group to target devices
1.11	Nov 11, 2025	2	Corrected the typo of the table of contents

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant chapters of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

   No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
7. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
8. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
9. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
10. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
11. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
12. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
13. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products. .

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev. 4.0-2 April 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

www.renesas.com/contact/.