RH850/U2C Group, U2B-E Group

Ethernet 10BASE-T1S Interface

要旨

本アプリケーションノートでは、RH850/U2C , U2B-E に搭載される Ethernet の PHY-LSI への接続処理および Ethernet 10BASE-T1S Interface (ETNF)モジュールの使用方法について説明します。

本資料およびプログラムは、RH850/U2C, U2B-E 搭載機能の理解促進を意図するものであり、量産設計を対象とするものではありません。

また、最新のマニュアル、正誤表、テクニカルアップデートや、開発環境の更新を反映しておりません。該当機能を使用される場合には、本プログラムは参考として扱い、最新のドキュメントや開発環境にて、お客様の責任において行ってください。

対象デバイス

- RH850/U2C Group, U2B-E Group
  動作確認済みマイコン型名：R7F702613FABB

対象統合開発環境

- IDE：CS+ (ルネサスエレクトロニクス製) Ver. E8.11.00c3 [10 May 2024] (RH850/U2C)
  デバイスファイル：DR7F702606.DVF
- コンパイラ：Green Hills Compiler (アドバンスド データ コントロールズ社製) V2023.5.4
- ビルド：GNU make Ver3.81 ※コマンドプロンプトで実行

使用 BSW

- MCAL：RH850/U2Cx AUTOSAR R22-11 MCAL ※Eth ドライバを使用

評価ボード　※本サンプルソフトの動作環境

- Main Board : Y-COMMON-MB-T1-V1 (ルネサスエレクトロニクス製)
- Piggyback Board : Y-RH850-U2C-292PIN-PB-T1-V1 (ルネサスエレクトロニクス製)
- Ethernet Extension Board : LAN8680 Ethernet PHY Transceiver (Microchip Technology 社製)
- 10BASE-T1S Ethernet PHY Transceiver Evaluation Board: EVB-LAN8670-USB (Microchip Technology 社製)
- E2 エミュレータ：RTE0T00020KCE00000R、変換アダプタ：RTE0T00020KCA00000R

参照文書

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

本アプリケーションノートは上記のマニュアルを参照し作成しております。

またデバイスの機能詳細及び電気的特性に関してはユーザーズマニュアル ハードウェア編に記載します。

目次

1. 概要

このドキュメントでは、RH850/U2C に搭載している Ethernet 10BASE-T1S Interface (ETNF)モジュールの使用方法について説明します。

1.1 用語と頭字語

表 1-1 用語と頭字語(1/2)

用語/頭字語	意味
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	Ethernet の基本的なアーキテクチャと動作原則を定義
Clause 22	MDIO インタフェースに関する仕様を定義
Clause 45	MDIO インタフェースの拡張仕様を定義
Clause 90	Ethernet の時間同期プロトコルをサポートするための仕様を定義
Clause 148	PLCA に関する仕様を定義
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	イーサネットの略称
ETNF	RH850/U2C 搭載 Ethernet 10BASE-T1S Interface モジュール
FIFO	First-In-First-Out
FSM	Finite State Machine
IFG	Inter-Frame Gap
IP	Intellectual Property
MAC	Media Access Control ([REF05]の Clause 4 参照)
MDIO	Management Data Input Output ([REF05]の Clause 22 参照)
MII	Media Independent Interface ([REF05]の Clause 22 参照)
NRZ	Non-Return to Zero

表 1-2 用語と頭字語(2/2)

用語/頭字語	意味
OA	Open Alliance
OS	Operating System
PCS	Physical Coding Sublayer
PHY	Physical Layer Entity
PLCA	Physical Layer Collision Avoidance ([REF03]の Clause 148 参照)
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 ([REF05]の Clause 90 参照)
TO	Transmit Opportunity
TSN	Time-Sensitive Networking
TXC	Open Alliance TC14 Transceiver ([REF01] 参照)
HS IntOSC	High Speed Internal Oscillator
Main OSC	Main Oscillator
PLL	Phase Locked Loop
SSCG	Spread Spectrum Clock Generator
arxml	AUTOSAR で定義されている XML 形式ファイルの拡張子
GHS	Green Hills

1.2 リファレンス

表 1-3 リファレンスドキュメント

リファレンス	ドキュメント名	ドキュメント番号
[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 10BASE-T1S の基本情報

10BASE-T1S は、IEEE 802.3cg 規格に基づくシングルペアイーサネット（Single Pair Ethernet, SPE）技術であり、産業用ネットワークや車載ネットワークにおける効率的で低コスト通信を実現するために設計されています。この技術は、従来の Ethernet の利点を活かしつつ、特定の用途に最適化されており、特に低速ながら信頼性の高い通信が求められる環境での使用に適しています。

技術的特徴

- 通信速度とケーブル構成:
  - 最大通信速度は 10Mbps で、低速ながら信頼性の高い通信を実現。
  - 単一対ツイストペアケーブルを使用し、配線の簡素化とコスト削減を可能にします。
- マルチドロップ接続:
  - ポイントツーポイント以外に最大 8 ノードまでのマルチドロップ接続をサポート。
  - スター型トポロジではなく、バス型トポロジを採用することで、配線コストを削減。
- PLCA（Physical Layer Collision Avoidance）:
  - 衝突回避機能を備え、効率的な半二重通信を実現。
  - 各ノードに送信機会を順番に割り当てることで、データ衝突を防ぎます。
- 低消費電力:
  - IoT や車載環境に適した省電力設計。
- 互換性:
  - Ethernet MAC やプロトコルスタックをそのまま使用可能。
  - 従来の Ethernet 資産を活用し、設計の効率化を図ります。
- PoDL（Power over Data Line）対応:
  - ポイントツーポイントの場合、信号線を通じて電力供給を行い、配線の軽量化と省スペース化を実現。

1.4 RH850/U2C 搭載 ETNF モジュールの機能概要

RH850/U2C に搭載されている ETNF モジュールは、イーサネット AVB コントローラと共に RMII（全二重モード）および T1S（半二重モード）のインタフェースに対応しています。また、内蔵の IP コアにより、10BASE-T1S イーサネット物理層の PMA、PCS、および PLCA の構成要素を備えています。本アプリケーションノートでは、ETNF モジュールの 10BASE-T1S（半二重モード）インタフェースの機能に特化して説明します。

ETNF における 10BASE-T1S 対応機能

- プロトコル
  - IEEE 802.3x 規格に準拠したフロー制御
- 転送速度
  - 10 [Mbps]
- 転送モード
  - T1S モード（半二重モード）
- 機能
  - OATC14 10BASE-T1S トランシーバ・インタフェースに準拠
  - OATC14 10BASE-T1S PLCA 管理レジスタに準拠
  - IEEE802.3cg-2019 に準拠
  - ディスクリプタ管理システム
- 送信/受信 FIFO
  - 送信時: 16 [KB]
  - 受信時: 8 [KB]
- 通信インタフェース
  - Open Alliance 10BASE-T1S トランシーバ・インタフェース

ETNF のイーサネット AVB コントローラ部

- AVB-DMAC（DMA 転送コントローラ）
  - 役割：URAM 内のデータストレージ領域と、受信・送信 FIFO バッファ間のデータ転送を DMA（ダイレクト・メモリ・アクセス）機能を利用して処理。
  - 動作： FIFO バッファからの直接の読み書きは不可。ディスクリプタと呼ばれる情報を利用して、送受信データの保存アドレスを管理。複数のディスクリプタをディスクリプタリストに配置することで、複数のイーサネットフレームを連続送受信可能。

- E-MAC（MAC コントローラ）
  - 役割：受信・送信 FIFO バッファと T1S トランシーバ間の転送を処理。
  - 動作： T1S トランシーバをサポートし、PHY とのインタフェースを提供。送信 FIFO 内のデータをもとにイーサネットフレームを構築し、T1S トランシーバへ送信。T1S トランシーバから受信したフレームに対して CRC チェックを実施し、受信 FIFO に書き込み。

ETNF のイーサネット PHY 部

- IEEE 802.3cg™ 10BASE-T1S 準拠のイーサネット物理層
  - PMA（Physical Medium Attachment）
  - PCS（Physical Coding Sublayer）
  - PLCA（PHY-Level Collision Avoidance）調整サブレイヤ
- MAC との連携
  - MII（Media Independent Interface）を使用し、標準 IEEE CSMA/CD イーサネット MAC と動作
  - PLCA RS の統合により、新しい PLCA MII 拡張を持たない MAC デバイスでも PLCA の機能を活用可能
- トランシーバ接続
  - PMA は、標準の OPEN Alliance 10BASE-T1S トランシーバと接続
- 管理機能
  - オプションのレジスタファイルを搭載
  - Clause 22/Clause 45 のレジスタを介した管理機能を提供
  - MDIO（Management Data Input/Output）スレーブモジュールを PHY トップレベルモジュールに接続

2. 10BASE-T1S の機能概要とソフトウェア開発

本章では、10BASE-T1S インタフェースの技術的な機能概要を示し、これを活用するためのソフトウェア開発環境と構築方法について説明します。

2.1 機能説明

10BASE-T1S の MAC 層はデータリンク層に属し、ポイントツーポイント接続と最大 8 ノードのマルチドロップ接続をサポートします。PLCA を採用することで、CSMA/CD の代わりに送信機会を管理し、衝突を回避しながら効率的な通信を実現します。MAC 層は、イーサネットフレームの送受信管理、フロー制御、およびエラー検出を担い、リアルタイム性が求められるアプリケーションに適しています。また、MDIO 経由で PHY 層と接続することで、既存のイーサネットインフラとの統合が可能ですが、適切な PHY を使用する必要があります。

本アプリケーションノートでは、MAC 層と PHY 間のデータリンクインタフェースとしては MDIO のシリアル通信とイーサネットの送信・受信フレーム通信をシェア(*1)した、3 ピン接続を採用しています。

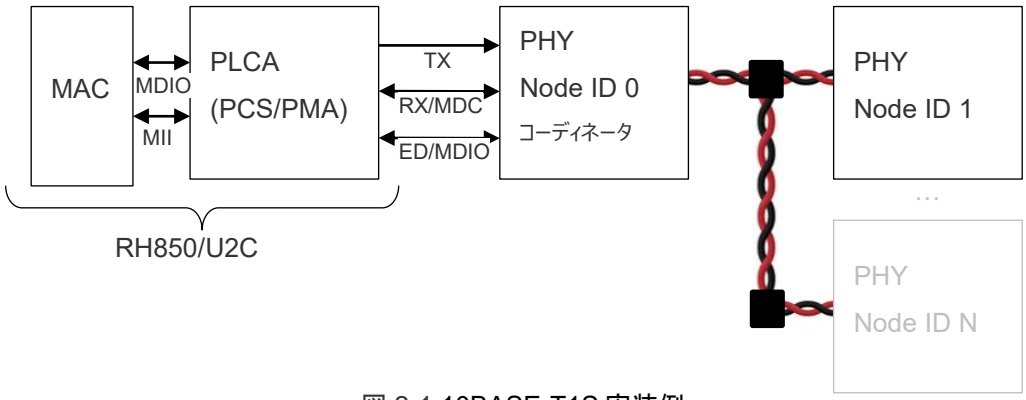


図 2-1 10BASE-T1S 実装例

表 2-1 機能ブロック説明

機能ブロック	説明
MAC 層	イーサネットのデータ転送を管理し、データフレームの構築や送受信フレームの解析を担当。フロー制御やエラー検出を行う。
PLCA 制御モジュール	ノード間の送信機会をスケジュールし、効率的なデータ転送を確保。ノード ID をビーコン信号により動作管理する。
PHY 層	電気信号を利用して通信を実現。10Mbps 通信速度に対応した単一対ツイストペアケーブルを使用し、PLCA により衝突を回避。

(*1) ETNF モジュールでは、3 ピン接続とは別に MDIO とイーサネット通信を分離した 5 ピン接続も可能です。

2.2 開発環境の整備

10BASE-T1S の実装には、ハードウェアの準備、ネットワーク設定、ソフトウェア構成の各要素を適切に設定・統合し、動作環境を構築する必要があります。これらの準備が完了することで、安定した動作を保証できます。以下に、開発環境の整備手順を詳しく説明します。

- ハードウェアの準備
  - Main Board、Piggyback Board、および Ethernet Extension Board の各ボードでジャンパ設定を行います。本サンプルソフトでは、Main OSC の発振周波数として 20MHz を想定しています。

表 2-2 各ボードのジャンパ設定(例)

Main Board		
ジャンパ名	設定	機能
JP1	3-2 接続	UART0(2-1) または LIN0(3-2) interface. (注)右が 1 番ピン
Piggyback Board		
ジャンパ名	設定	機能
JP1 GETH0BVCC	OPEN	SGMII 用電源
JP1 GETH0PVCC	OPEN	SGMII 用電源
JP1 (その他)	-	3.3V 側または 5V 側 または OPEN に設定してください
JP2	OPEN	コア電圧の選択
JP3	CLOSED	デバイスの選択 ·JP3[OPEN]: U2C8 ·JP3[CLOSED]: U2C4
JP4	OPEN	ポート P00_7 を GND にプルダウンする
JP6	2-3 接続	TRST#信号の信号源を選択する ·JP6[1-2]: TRST#信号を E0VCC に固定する ·JP6[2-3]: TRST#信号は E2 デバッグコネクタ（コネクタ CN9 のピン 3）からの TRST_TOOL#信号である
JP7	OPEN	FLMD0 信号を「H」に変更する
JP8	OPEN	FLMD1 信号を「GND」に変更する
JP9	2-5 接続	+3.3 V 電源ソースを選択する ·JP9[1-4]: オンボード電圧レギュレータから+3.3 V を取得する ·JP9[2-5]: メインボードから+3.3 V を取得する ·JP9[3-6]: CN7 の外部+3.3 V 電源から+3.3 V を取得する
JP10	CLOSED	電流測定ブリッジ +3.3 V
JP11	CLOSED	メインボードからの+5.0 V 電源を有効にする
JP12	CLOSED	電流測定ブリッジ +5.0 V
JP13	OPEN	デバッガ用イベントトリガ出力 EVTO0#
JP14	OPEN	プルアップ/プルダウンコネクタ CN12 のプルアップ電圧を選択 ·JP14[1-2]: ピン 1/3/5/7 を 5.0V プルアップ ·JP14[2-3]: ピン 1/3/5/7 を 3.3V プルアップ
JP15	OPEN	プルアップ/プルダウンコネクタ CN12 のプルアップ電圧を選択 ·JP15[1-2]: ピン 9/11/13/15 を 5.0 V プルアップ ·JP15[2-3]: ピン 9/11/13/15 を 3.3 V プルアップ
JP17	OPEN	ポート P20_0 をメインボードに接続して選択 ·JP17[OPEN]：P20_0 を B1_P20_0（ETH0RXD0）に接続 ·JP17[CLOSED]：P20_0 を B2_P20_0（I2SMCLK）に接続
JP18	OPEN	Ethernet1 MDIO/MDC ポート選択 ·JP18[OPEN]: ETH1_MDIO = P21_0, ETH1_MDC = P20_11 ·JP18[CLOSED]: ETH1_MDIO = P04_9, ETH1_MDC = P04_8
JP19	OPEN	基板製造テスト用ジャンパ
JP20	OPEN	LED 出力を有効にする
Ethernet Extension Board		
ジャンパ名	設定	機能
終端抵抗ジャンパ	CLOSED	終端抵抗を有効にする（ツイストペアケーブルコネクタ近辺）

- Main Board、Piggyback Board および Ethernet Extension Board を接続します。接続時は、確実にコネクタを奥まで差し込み、接触不良が発生しないよう注意してください。ツイストペアケーブルをネジ止め式端子台に接続する際は、芯線の飛び出しや締め付け不足によるショートを防ぐため、適切な接続を行ってください。特に、1 つのネジ止め式端子台に 2 系統のツイストペアケーブルを接続する場合、締め付け不足やケーブルのずれによる接触不良や信号劣化を防ぐため、端子の固定状態を十分に確認してください。※ノード数が 3 以上の場合（例：ボード 2 セット+モニタ）

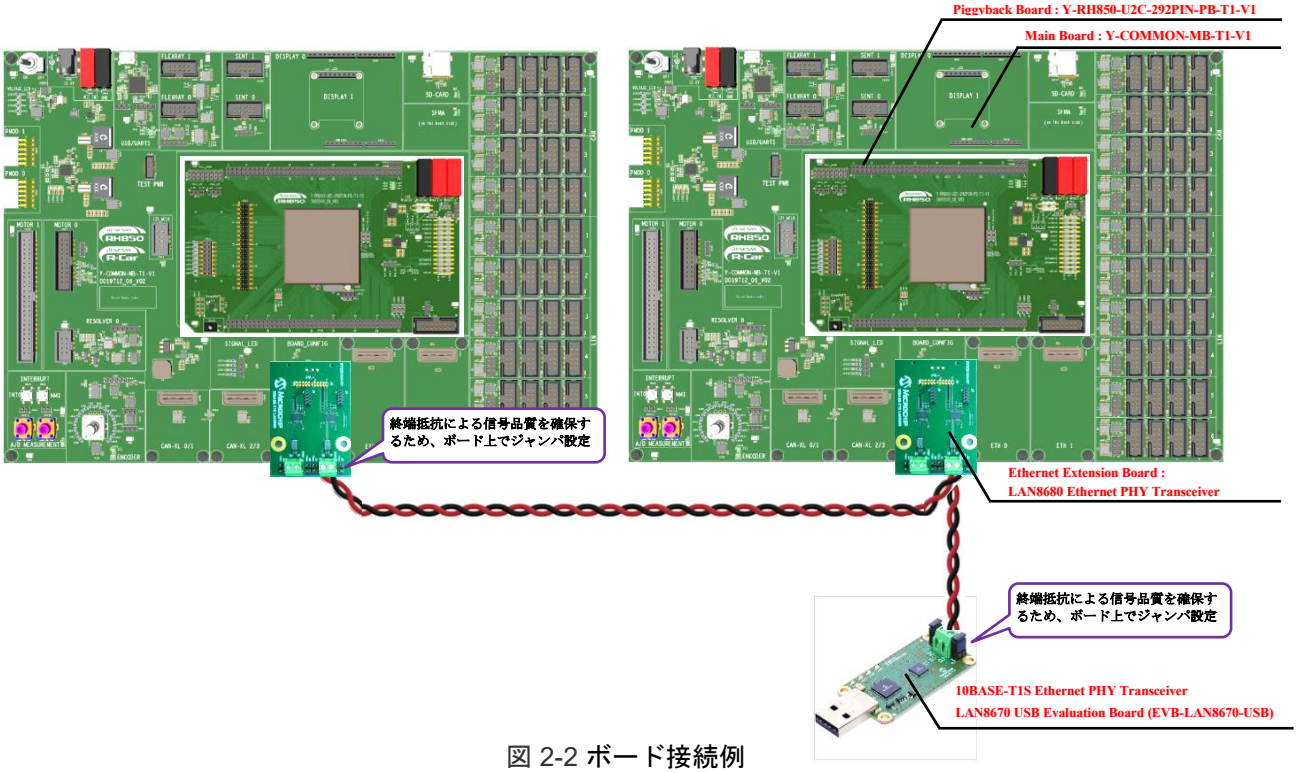


図 2-2 ボード接続例

- Piggyback Board に変換アダプタを介して E2 エミュレータを接続してください（変換アダプタの SW1 は'1'側に設定）。Main Board にボードコンフィギュレーション用の USB ケーブルを接続後、12V の電源を供給します（電源アダプタまたは安定化電源を使用）。ボードコンフィギュレーションツールでは、本サンプルソフトがデフォルト設定（Set default ボタン）で正常に動作することが確認されています。

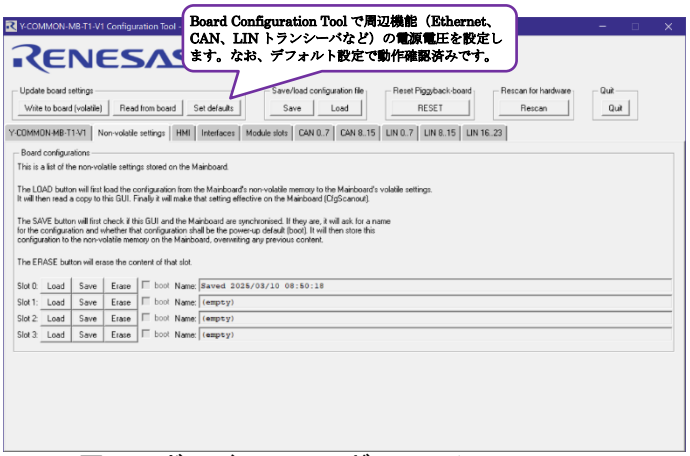


図 2-3 ボードコンフィギュレーションツール

- Ethernet 通信のモニタとしては、10BASE-T1S Ethernet PHY Transceiver Evaluation Board の EVB-LAN8670-USB を使用しています。Windows PC では、USB 2.0 インタフェースを 10BASE-T1S ネットワークインタフェースに接続するためのネットワークアダプタとして動作します。また、プロパティの詳細設定で物理層のパラメータを調整します。（次図の例を参照）

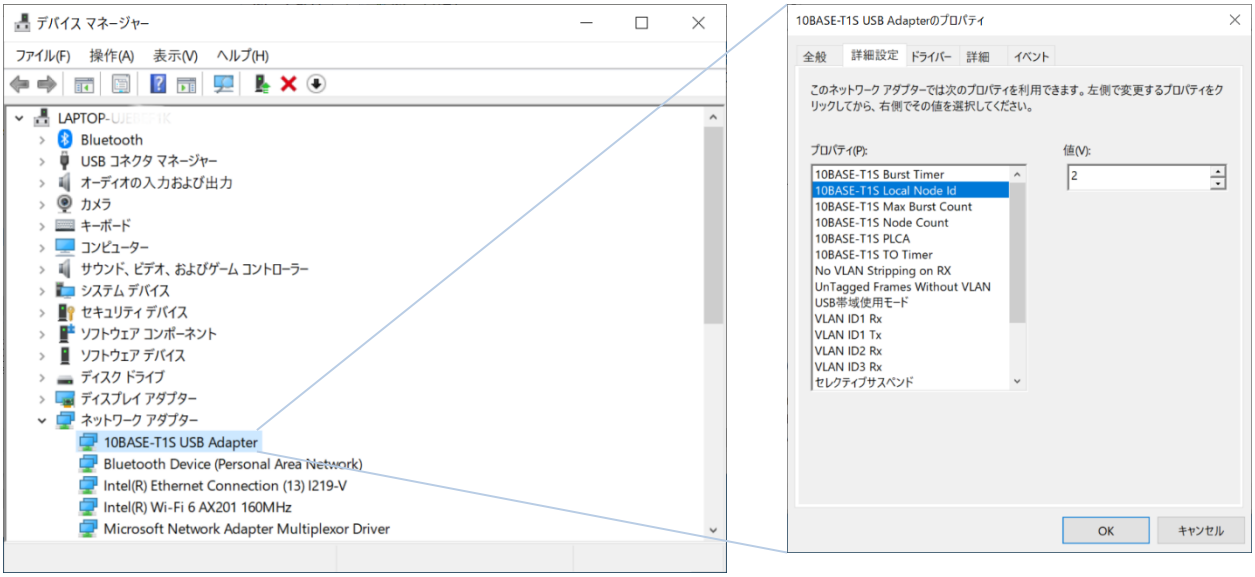


図 2-4 ネットワークアダプタ

10BASE-T1S USB Adapter のプロパティ「詳細設定」を一覧表示します。

表 2-3 10BASE-T1S USB Adapter のプロパティ(例)

プロパティ	値
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	有効
10BASE-T1S TO Timer	32
No VLAN Stripping on RX	Enabled
UnTagged Frames Without VLAN	Enabled
USB 帯域使用モード	自動
VLAN ID1 Rx	0
VLAN ID1 Tx	0
VLAN ID2 Rx	0
VLAN ID3 Rx	0
セレクティブサスペンド	無効
セレクティブサスペンドアイドルタイムアウト	10
ネットワークアドレス	存在しない
フロー制御	無効
速度と半二重/全二重	10Mbps/半二重
優先 VLAN	優先 VLAN 無効

- ビルド環境の準備
  - 統合開発環境と C コンパイラをインストールします。CS+統合開発環境や GHS コンパイラについては、対応するマニュアルを参照してください。
  - MCAL を準備してください。本サンプルソフトでは Eth ドライバを使用します。
  - ビルドには GNU Make 3.81 を使用します。ダウンロードサイトから取得し、インストールしてください。
  - Windows PC のシステムの詳細設定で環境変数(GNUMAKE)を設定します。Make コマンドのインストール先フォルダを指定します。

図 2-5 GNU Make の環境変数設定

  - 環境変数の設定として、GNU Make のパス設定はビルド用バッチファイル ***T1S_SampleApp.bat*** 内で指定されています。

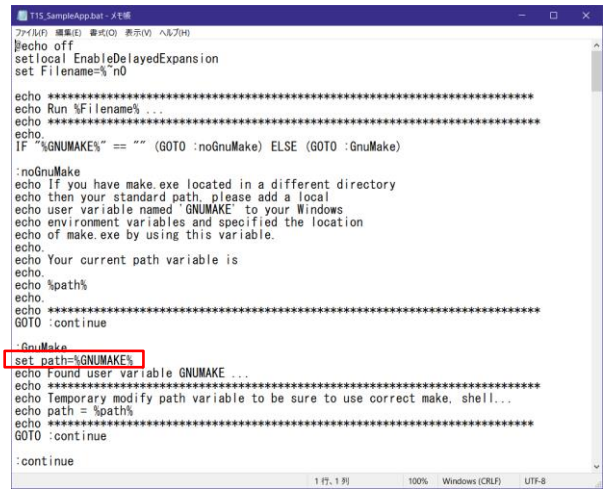


図 2-6 GNU Make のパス設定

- サンプルソフトの動作確認手順

  ➢ ***command.bat*** を実行して、コマンドプロンプトを起動します。本サンプルソフトをビルドするには、コマンドプロンプト上で対象のプロジェクトへ移動した後、ビルド用バッチファイル ***T1S_SampleApp.bat*** を実行します。

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

※ノード ID0 プロジェクトの場合

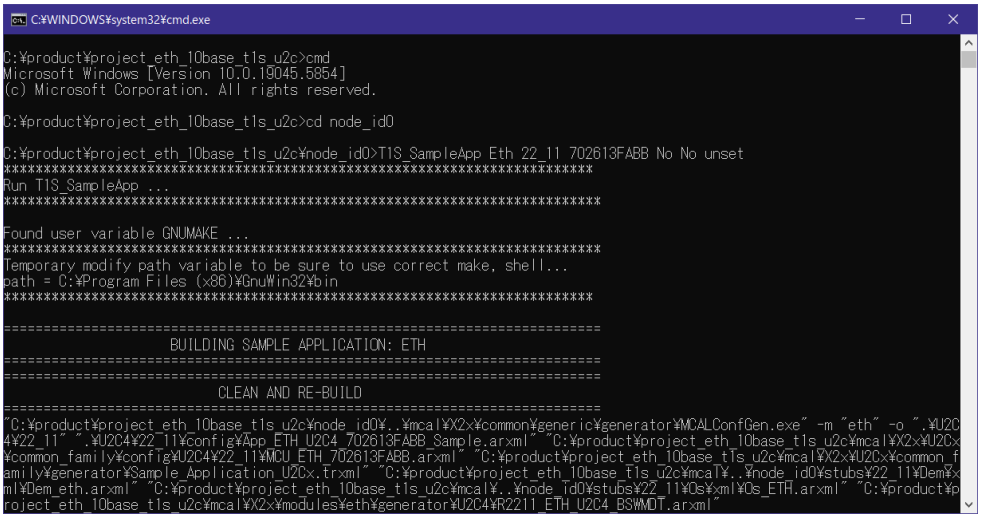
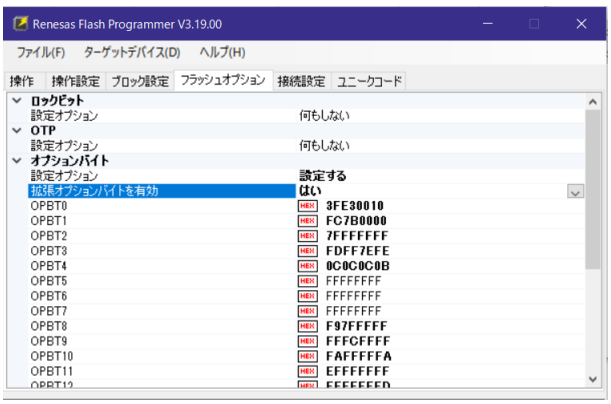


図 2-7 ビルド実行例

バッチファイルの引数は MCAL のサンプルソフトと同じであるため、詳細は MCAL のマニュアルを参照してください。また、GHS コンパイラを実行する際は、ドングルやライセンスファイルが正しく認識されていることを事前に確認してください。なお、本サンプルソフトのファイル構成については、**表 4-1** を参照ください。

➢ E2 エミュレータを使用してオプションバイトを確認してください。Renesas Flash Programmer を使用する場合、Piggyback Board の FLMD1 ピンレベルを JP8[CLOSED]: FLMD1 = GND に設定してください。また、オプションバイトを設定した後、P06_7（LIN0RX）を使用する場合は、JP8[OPEN]: FLMD1 = OPEN に戻してください。

```
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
```



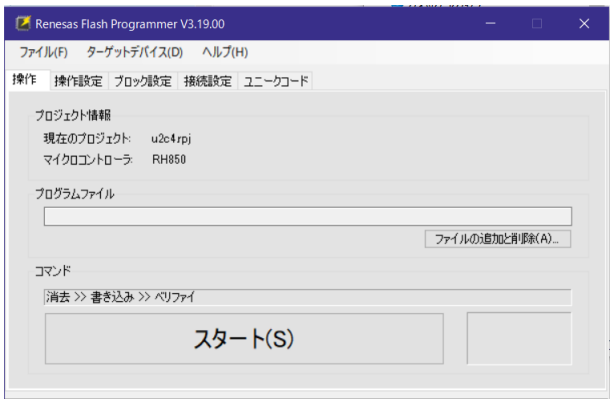


図 2-8 フラッシュオプションタブ非表示の場合

Renesas Flash Programmer でフラッシュオプションタブが表示されないなどの理由で拡張オプションバイトを直接確認または設定できない場合は、対象エリアを一時的にモトローラ・S タイプ・ファイルに変換することで、確認・設定を行う方法があります。モトローラ・S タイプ・ファイルの値を変更する際は、バイトエンディアンとチェックサム値に注意してください。

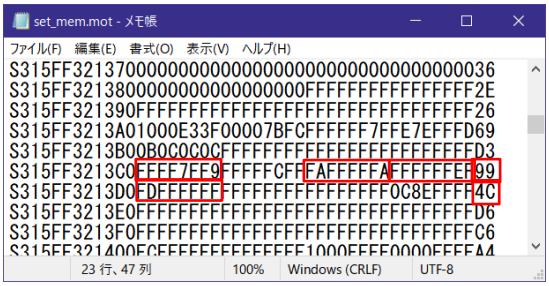


図 2-9 オプションバイト設定例

- RH850/U2C 対応の CS+を 2 つ起動して、サンプルソフトのプロジェクト（ノード ID0 のプロジェクト：***run_node_id0.mtpj*** 、または、ノード ID1 のプロジェクト：***run_node_id1.mtpj***）をそれぞれ開きます。プロジェクトファイルをダブルクリックすると、RH850/U2C に対応していない CS+が誤って起動する可能性があるため、注意してください。

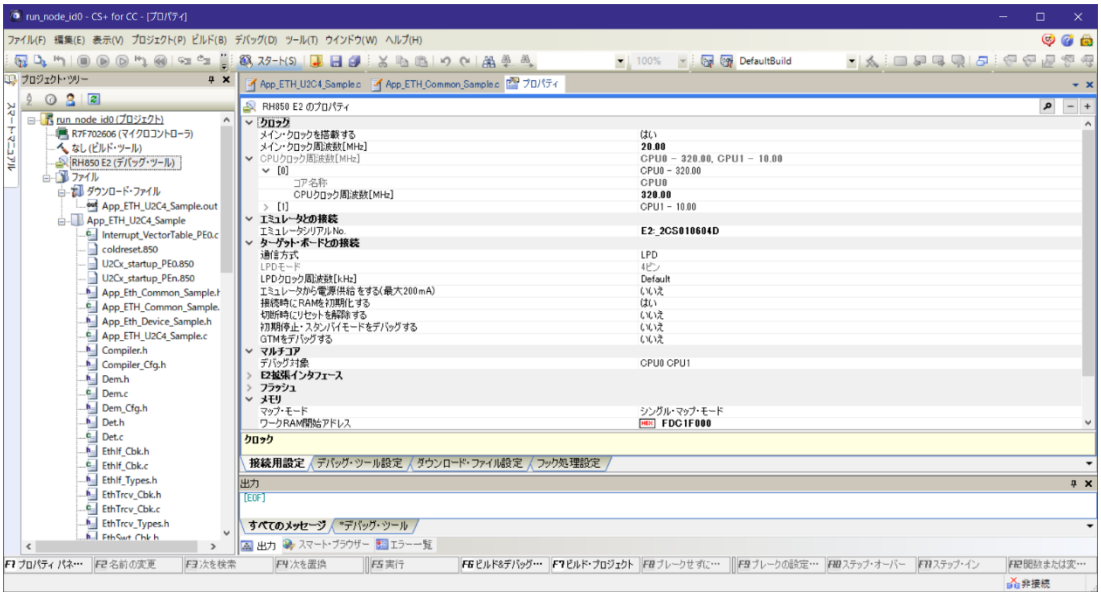


図 2-10 CS+起動

➢ ビルドで生成されたロードモジュールファイルをそれぞれダウンロードして実行します。

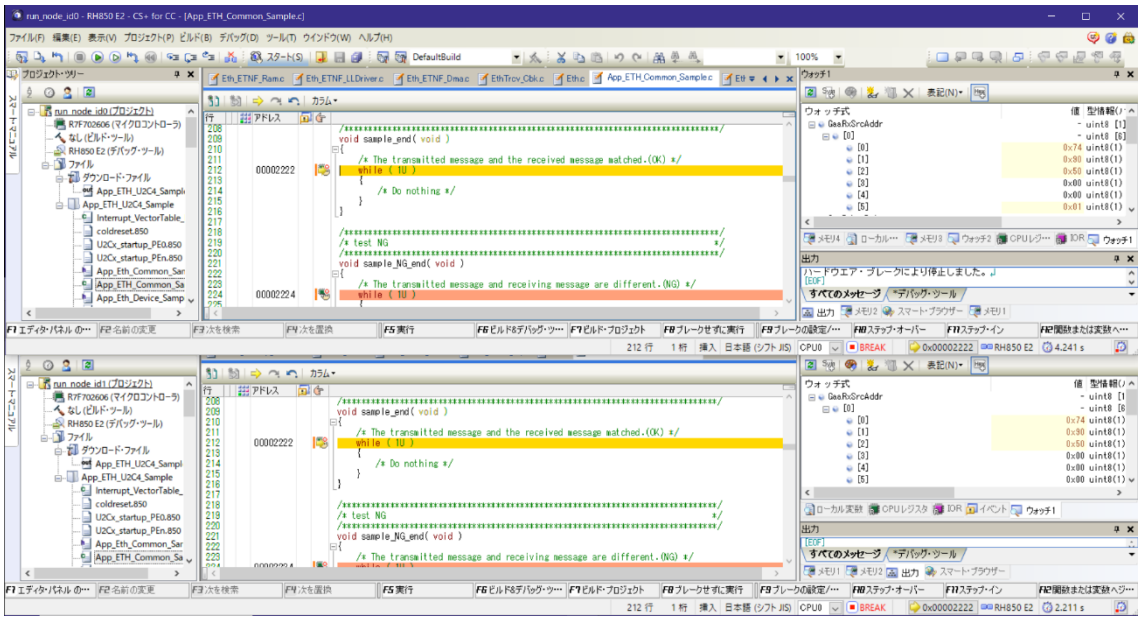


図 2-11 サンプルソフトの実行（ボード 2 セット時）

- 10BASE-T1S ネットワークの通信確認

➢ Wireshark を使用し、EVB-LAN8670-USB を介してボード 2 セット間のイーサネット通信のパケットをキャプチャ・解析する。

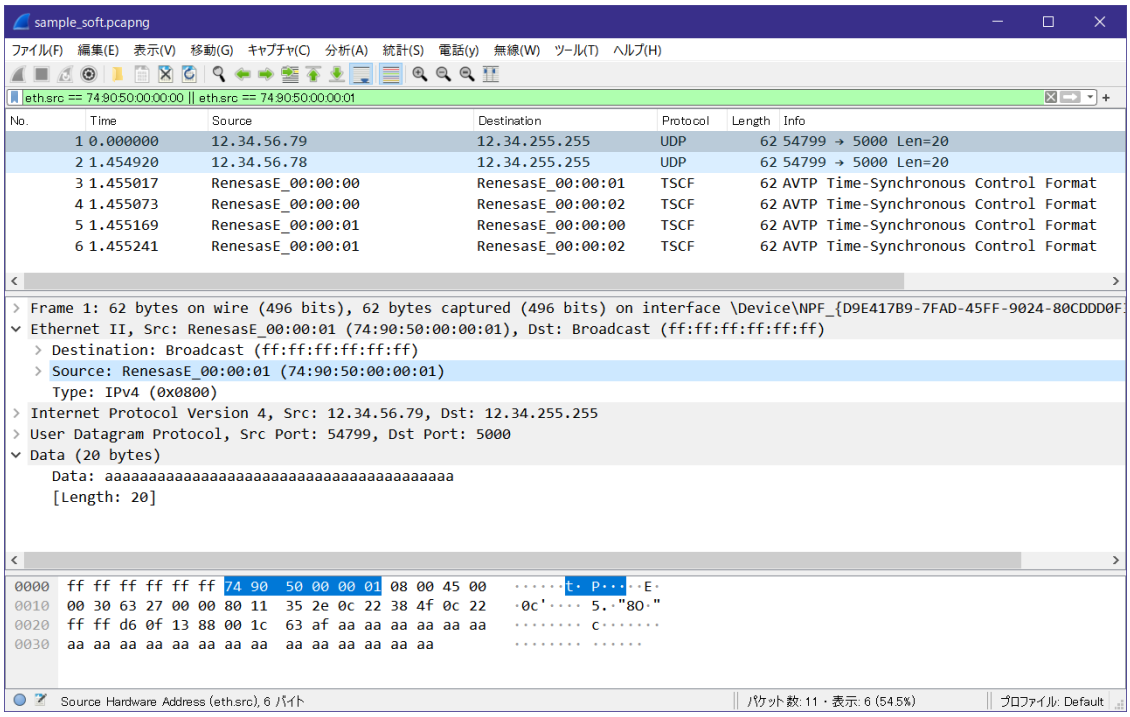


図 2-12 イーサネット通信のモニタ（ボード 2 セット間の通信キャプチャ例）

3. 10BASE-T1S の使用方法

10BASE-T1S を活用するためのソフトウェア処理について、MCAL の Eth ドライバを使用し、ETNF モジュールの活用方法を解説します。本アプリケーションノートでは、10BASE-T1S の通信プロトコル（Ethernet フレーム構造、MAC アドレス管理）の詳細な説明は省略し、ソフトウェア処理の活用に焦点を当てます。

以下に、Ethernet トランシーバのコンフィグ・パラメータを一覧表示します。これにより、各設定項目の概要を把握し、適切なパラメータ選定が可能になります。

表 3-1 Ethernet トランシーバのコンフィグ・パラメータ

定義名	内容
T1S_PHY_ADDR	トランシーバの PHY アドレスを設定する。
ETH_PHY_ADDR	マイコン内蔵 PHY アドレスを設定する。
T1S_PHY_CONTROL_CFG	CONTROL レジスタを設定する。
T1S_PHY_T1SPMACTRL_CFG	T1SPMACTRL レジスタを設定する。
T1S_PHY_T1SPMATSTM_CFG	T1SPMATSTM レジスタを設定する。
T1S_PHY_T1SPCSCTRL_CFG	T1SPCSCTRL レジスタを設定する。
T1S_PHY_T1STWEAKS_CFG	T1STWEAKS レジスタを設定する。
T1S_PHY_T1SPLCAEXT_CFG	T1SPLCAEXT レジスタを設定する。
T1S_PHY_T1SPLCAEXT_CFG	T1SPLCAEXT レジスタを設定する。
T1S_PHY_T1SPMATUNE0_CFG	T1SPMATUNE0 レジスタを設定する。
T1S_PHY_T1SPMATUNE1_CFG	T1SPMATUNE1 レジスタを設定する。
T1S_PHY_PLCACTRL0_CFG	PLCACTRL0 レジスタを設定する。
T1S_PHY_PLCACTRL1_CFG	PLCACTRL1 レジスタを設定する。
T1S_PHY_PLCATOTMR_CFG	PLCATOTMR レジスタを設定する。
T1S_PHY_PLCABURST_CFG	PLCABURST レジスタを設定する。
ビット定義名	**内容**
T1S_PHY_ADDR1_REG18H_SET_CFG	XCVR_ANALOG_SET_2 のセットビット値を設定する。
T1S_PHY_ADDR1_REG18H_CLR_CFG	XCVR_ANALOG_SET_2 のクリアビット値を設定する。
T1S_PHY_ADDR1_REG1FH_SET_CFG	XCVR_ANALOG_SET_9 のセットビット値を設定する。
T1S_PHY_ADDR1_REG1FH_CLR_CFG	XCVR_ANALOG_SET_9 のクリアビット値を設定する。
T1S_PHY_REG000000_RESET	Soft Reset を設定する。
T1S_PHY_REG000000_LOOP_OFF	ループバックしない設定とする。
T1S_PHY_REG000000_LCTL_ON	PHY リンク制御を有効設定とする。
T1S_PHY_REG000000_LPWR_OFF	低電力モードに入らない設定とする。
T1S_PHY_REG000000_ISOM_OFF	分離モードに入らない設定とする。
T1S_PHY_REG000000_CTEST_OFF	衝突実行モード無効設定とする。
T1S_PHY_REG2108F9_LPWR_OFF	低電力モードにしない設定とする。
T1S_PHY_REG2108F9_LOOP_OFF	ループバックモードにしない設定とする。
T1S_PHY_REG2108FB_NORMAL	Normal operation 設定とする。
T1S_PHY_REG2308F3_LOOP_OFF	ループバックモードにしない設定とする。
T1S_PHY_REG3F8001_PKTLOOP_OFF	TX フレームが MII RX に反映されない設定とする。
T1S_PHY_REG3F8001_ENIE_OFF	拡張ノイズ耐性モードにしない設定とする。
T1S_PHY_REG3F8001_UNJT_OFF	自動回復しない設定とする。
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設定とする。
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設定とする。
T1S_PHY_REG3F8001_NCOLM_ON	衝突検出はマスクされない設定とする。
T1S_PHY_REG3F8001_RXDLY_ON	同時通信を回避するために MII RX 信号が遅延される設定とする。
T1S_PHY_REG3F8002_PREN_OFF	PLCA RS は標準モードで動作設定とする。
T1S_PHY_REG3F8002_LDEN_OFF	PLCA リーダはノード ID によって選択設定とする。
T1S_PHY_REG3F8002_LDR_OFF	LDEN=1 であれば、PLCA スレーブとする設定とする。
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
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
T1S_PHY_REG3FCA01_EN_ON	PLCA RS 機能が有効設定とする。
T1S_PHY_REG3FCA02_NCNT	総ノード数 NCNT を設定する。
T1S_PHY_REG3FCA02_ID	ノード ID を設定する。
T1S_PHY_REG3FCA04_TOTMR	PLCA 送信機会タイマを設定
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

本サンプルソフトのパラメータ設定値は 4 章を参照し、MCAL（Eth ドライバ）のコンフィグ・パラメータの詳細はマニュアル（[REF07]）を参照してください。本章では、ソフトウェア処理の実践的な活用として、対象 Eth ドライバの機能概要と、Eth トランシーバを含む処理手順について詳しく説明します。

前提条件

- AUTOSAR MCAL の Eth ドライバを使用し、Ethernet 通信の制御を行います。

Eth ドライバの概要

- 基本仕様
  - AUTOSAR R22_11 に準拠
  - Det、Dem、EthIf 等のモジュールはスタブ（**表 4-1 ファイル構成（抜粋）** 参照）で実装
    - 動作要件に従い、スタブ実装済みのモジュールに対して必要な調整や処理変更を行ってください。
- スレッド管理
  - マルチスレッド非対応
    - AUTOSAR 規定のシングルスレッドのみ使用可能
- 初期化管理
  - MCU 固有の初期化は Eth ドライバで未実装
    - 本サンプルソフトでは Eth ドライバ以外の処理として実装済み
  - 初期化前に使用できる API
    - `Eth_Init` の前に呼び出し可能なのは `Eth_GetVersionInfo` と `Eth_MainFunction`のみ

Eth ドライバのネットワーク管理

- アドレスフィルタリング
  - Eth_UpdatePhysAddrFilter API
    - コントローラがアクティブまたはダウン状態のときに呼び出し可能
    - ブロードキャストアドレス (`FF:FF:FF:FF:FF:FF`) はダウン状態のみ許可
    - プロミスキャスモード
      - `FF:FF:FF:FF:FF:FF` を指定すると有効化
      - `00:00:00:00:00:00` を指定すると解除
      - プロミスキャスモード中は、すべてのマルチキャストアドレスを無視
    - アドレス登録の制限
      - マルチキャストアドレスは最大 32 個まで登録可能

- ユニキャストアドレスは登録不可
- `00:00:00:00:00:00` を渡すとすべての登録済みマルチキャストアドレスを削除

➢ Eth_SetPhysAddr API

✧ ユニキャストアドレスのみ指定可能

✧ マルチキャスト・ブロードキャスト不可

Eth ドライバのデータ送受信管理

- 受信バッファ管理

➢ EthIf_RxIndication API

✧ `LenByte`パラメータにはイーサネットフレームヘッダを除いたペイロードサイズを設定

✧ 受信バッファは処理後すぐに解放され、バッファリング 1 周後に上書き

- 送信バッファ管理

➢ Eth_Transmit API

✧ 送信完了後、送信バッファを解放

✧ 内部リソースが不足すると`E_NOT_OK` を返すため、アプリケーション側で再試行が必要

✧ ポーリングモード

- 送信バッファは `Eth_TxConfirmation` によって解放される
- `EthIf_TxConfirmation` が不要でも、定期的に `Eth_TxConfirmation` の呼び出しが必要

Eth ドライバのコントローラ管理

- コントローラの状態管理

➢ Eth_SetControllerMode API

✧ `ETH_MODE_DOWN` が渡されると、進行中の送信完了までブロック

✧ 最大ブロック時間は `EthTimeout` で設定可能

➢ コントローラのアクティブ・ダウン状態

✧ `Eth_ProvideTxBuffer`, `Eth_Transmit`, `Eth_Receive`, `Eth_TxConfirmation` はアクティブ状態でのみ使用可能

✧ `Eth_SetPhysAddr` はダウン状態のみ許可

Eth ドライバの QoS と優先度制御

- 送信キュー管理

➢ AVB（Audio Video Bridging）

✧ 最大 4 個の送信キューを設定可能

➢ 送信キューの優先度設定

- ✧ `EthTxQueueIdx` パラメータで制御
  - `0:` ベストエフォート
  - `1:` Precision Time Protocol のネットワーク制御
  - `2:` CBS アルゴリズム（クラス B）
  - `3:` CBS アルゴリズム（クラス A）
- ➢ QoS サポート
  - ✧ `EthQosSupport` が無効の場合、すべての送信キューはベストエフォート扱い
  - ✧ AVB 標準のみを適用

Eth ドライバの帯域幅管理

- CBS（クレジットベースシェーパー）
  - ➢ `EthTxQueuePolicy` を `ETH_CBS` に設定すると CBS アルゴリズム が適用
  - ➢ `EthCtrlTxQueueBwFraction` を利用し、送信キューの帯域幅割合 を設定可能
  - ➢ CBS を有効化すると、レイテンシ問題の解決に貢献

Eth ドライバの制御フレーム

- ブロードキャスト/マルチキャストストーム検出
  - ➢ `EthBroadcastStormThreshold` でブロードキャストストーム検出の閾値を設定
  - ➢ `EthMulticastStormThreshold` でマルチキャストストーム検出の閾値を設定
- フロー制御
  - ➢ EthPauseFrame パラメータ
    - ✧ true に設定すると送受信フロー制御が有効
    - ✧ `EthPauseTime` で ポーズ時間 を設定
    - ✧ `EthPauseFrameRetransmissionTime` で 再送時間 を設定

Eth ドライバのタイムスタンプ管理

- 受信フレームのタイムスタンプ
  - ➢ `Eth_GetIngressTimeStamp` で受信フレームのタイムスタンプを取得可能
  - ➢ `EthBeTimeStampStore` を true に設定すると、ベストエフォートの受信ディスクリプタにタイムスタンプ情報を付加
  - ➢ `EthStreamTimeStampStore` を true に設定すると、ストリームの受信ディスクリプタにタイムスタンプ情報を付加

3.1 初期化処理

10BASE-T1S の通信を行うために、ETNF モジュールの Ethernet MAC 機能と PHY インタフェースを活用した初期化手順を、フローチャートを用いて説明します。なお、本フローチャートでは、AUTOSAR MCAL の Eth ドライバを使用することを前提としています。

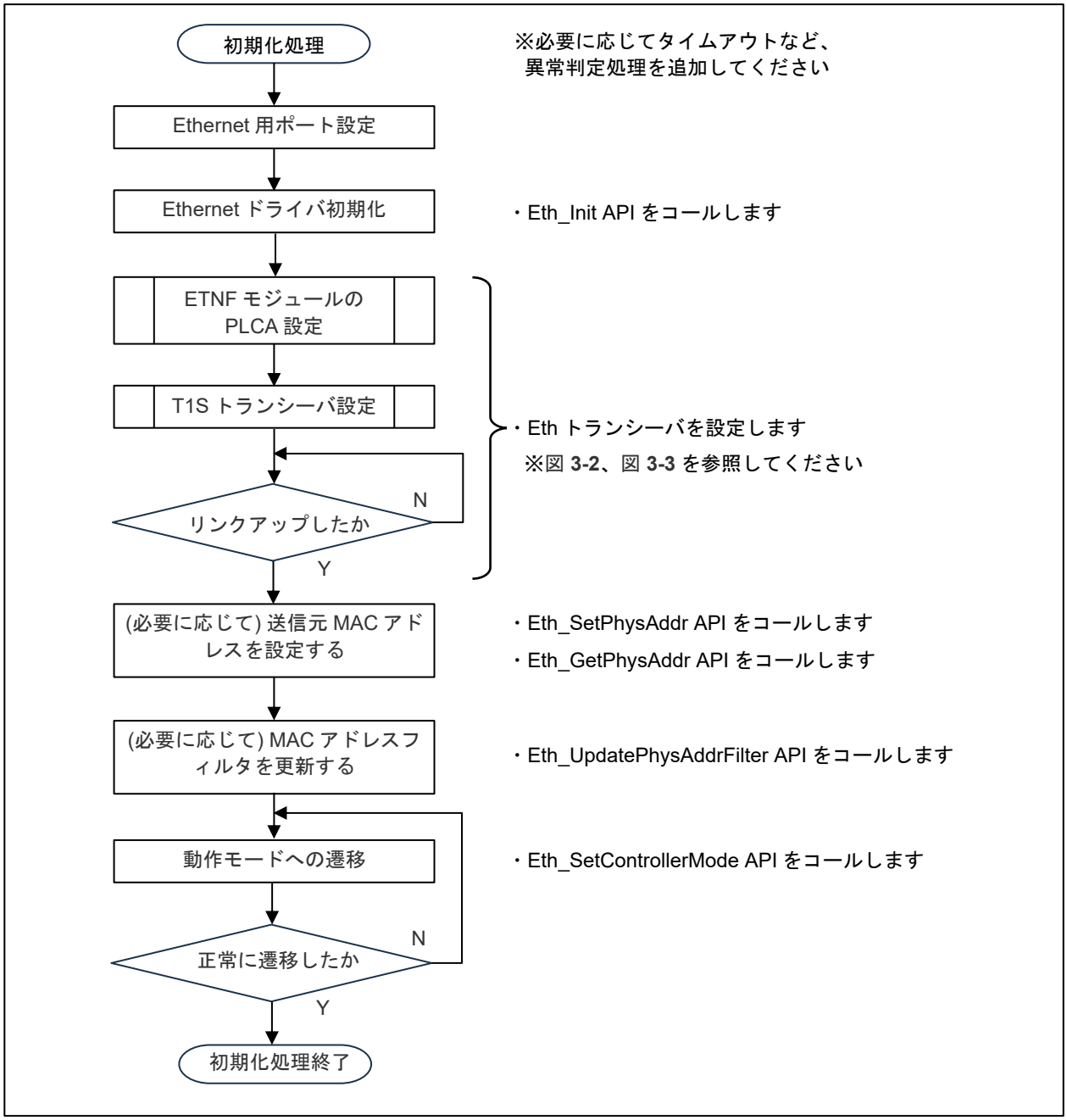


図 3-1 T1S 初期化処理例

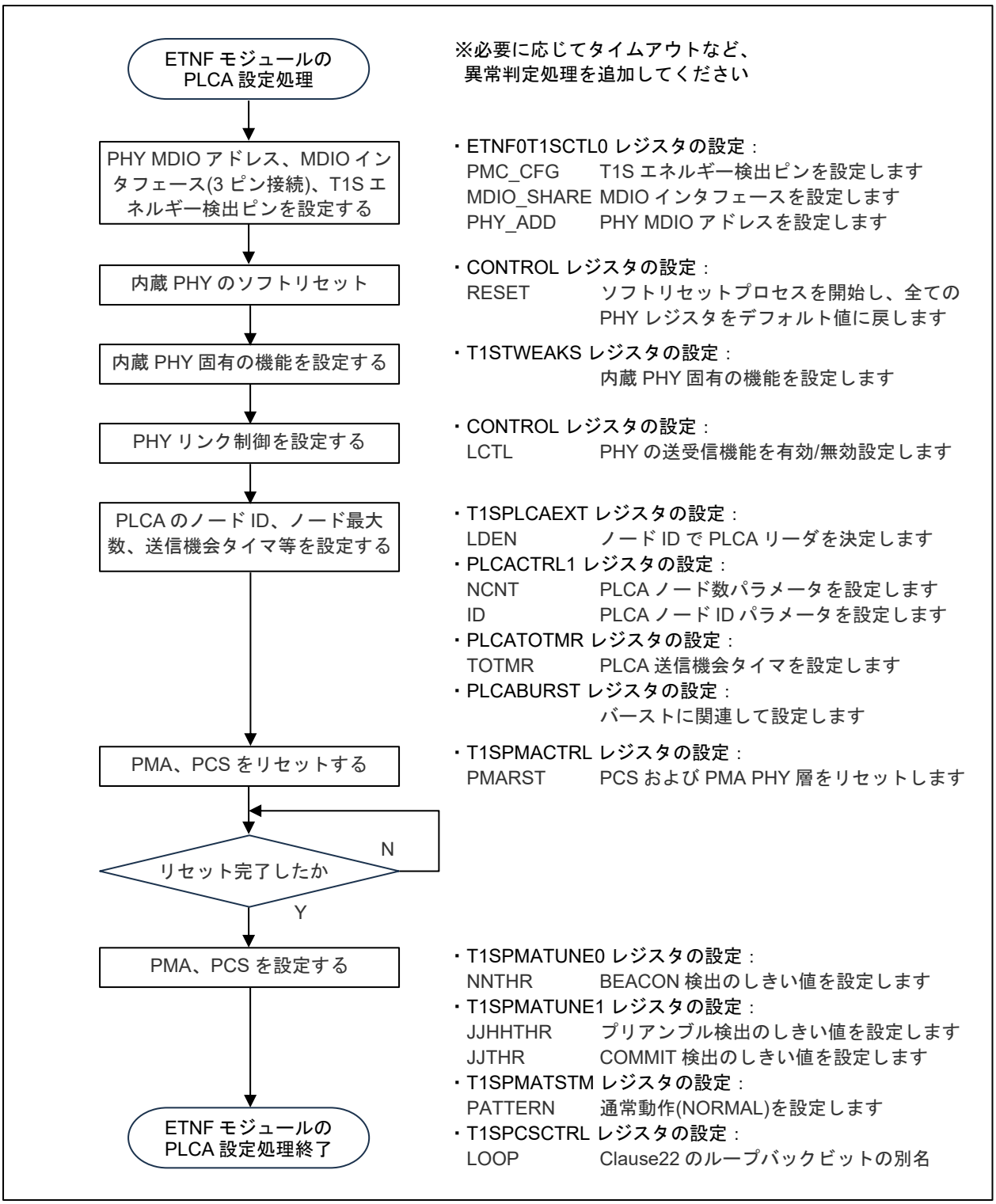


図 3-2 PLCA 設定処理フロー例

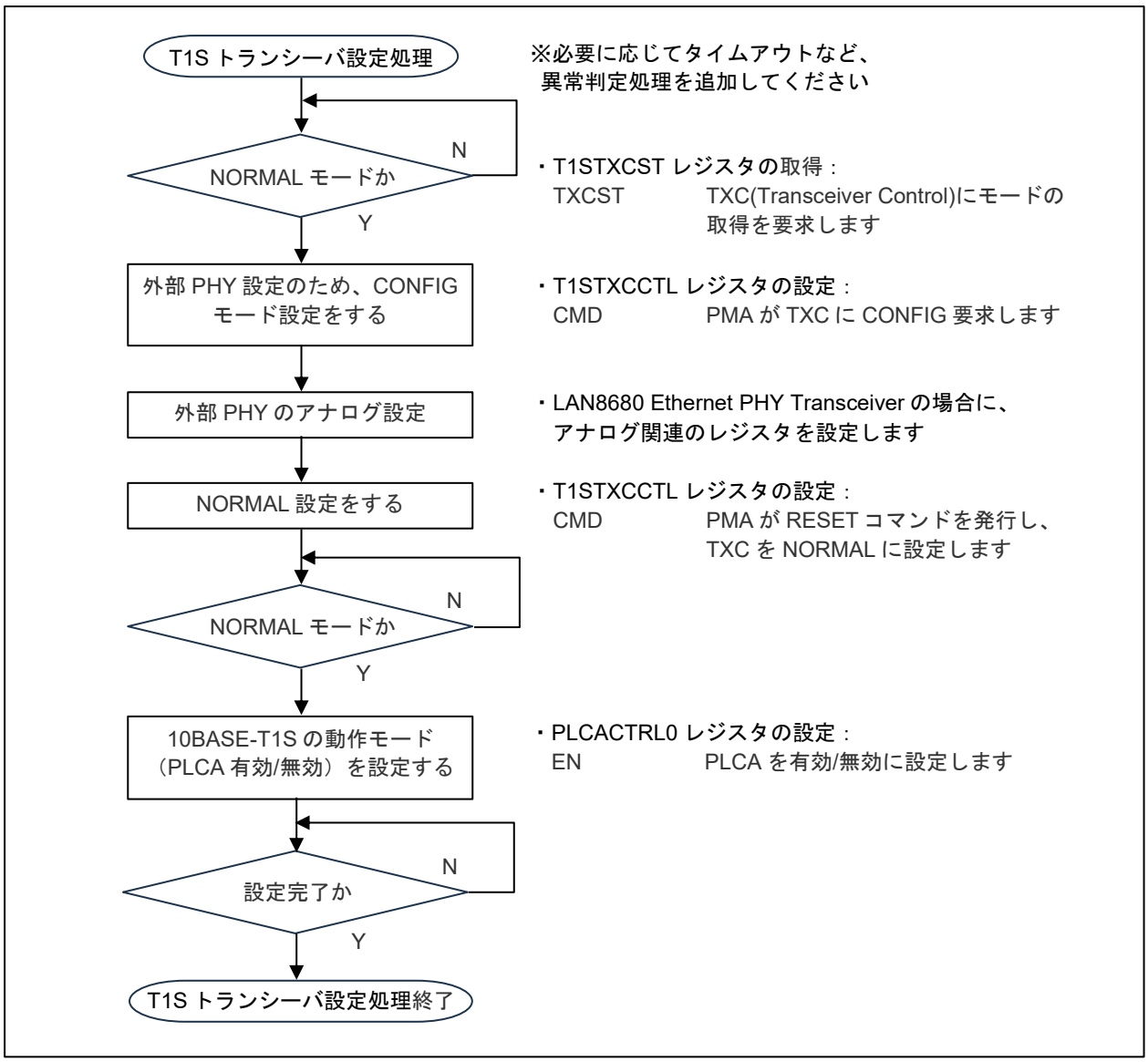


図 3-3 T1S トランシーバ設定処理フロー例

3.2 データ送信処理

AUTOSAR MCAL の Eth ドライバの使用を前提に、10BASE-T1S のデータ送信処理をフローチャートで説明します。

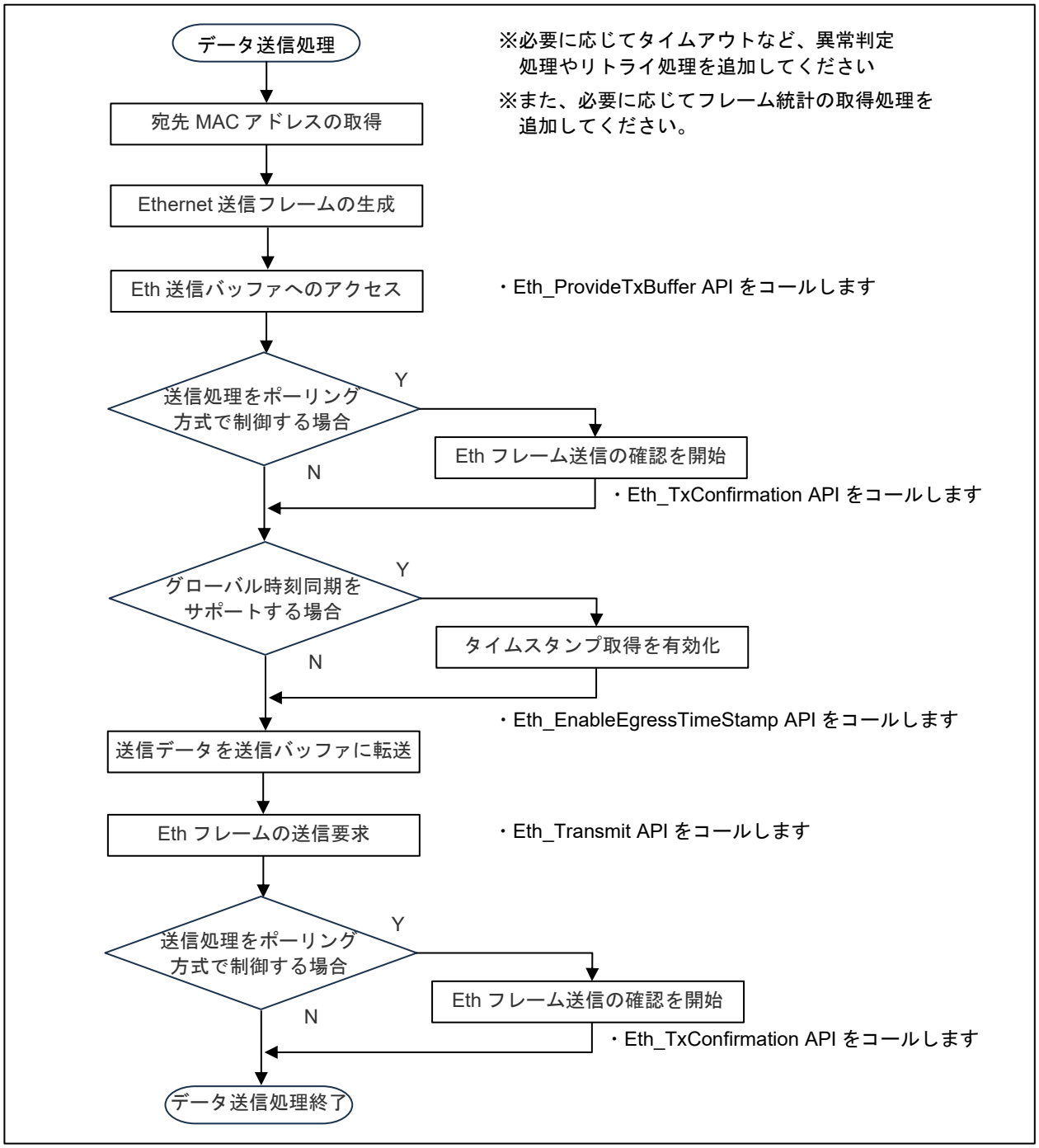


図 3-4 データ送信処理フロー例

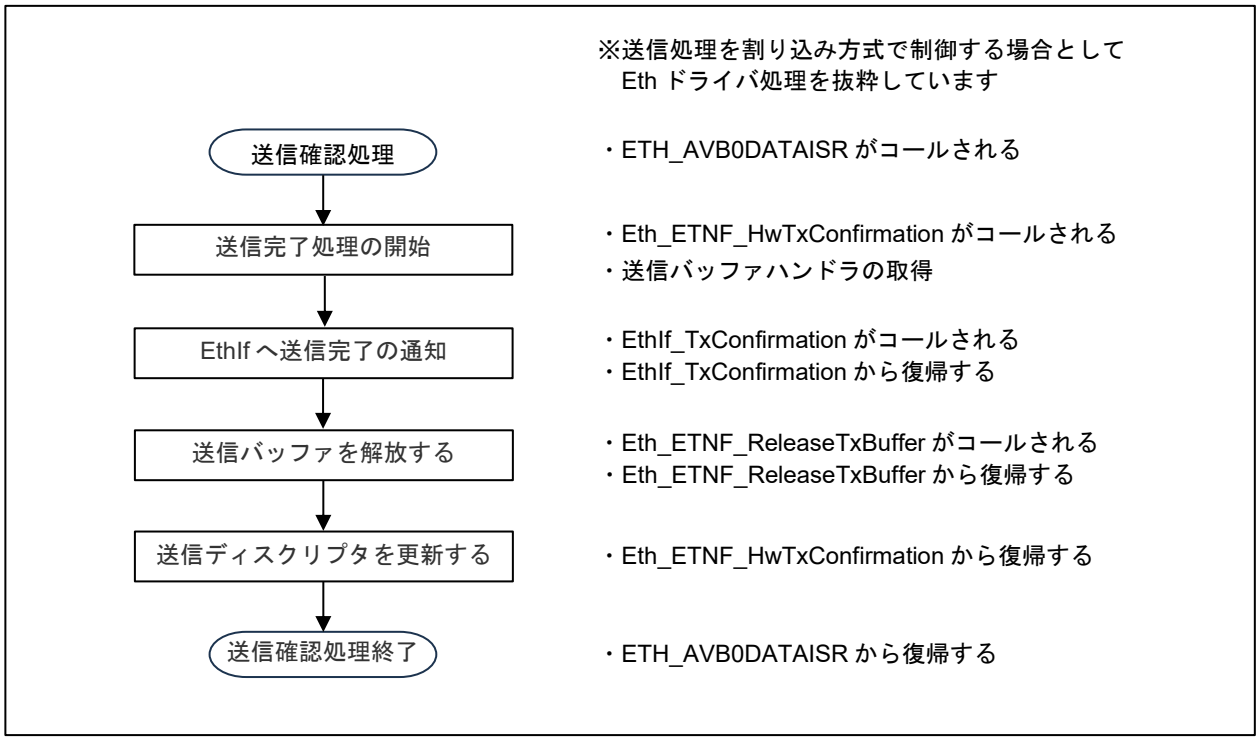


図 3-5 送信割り込み処理フロー

3.3 データ受信処理

AUTOSAR MCAL の Eth ドライバの使用を前提に、10BASE-T1S のデータ受信処理をフローチャートで説明します。

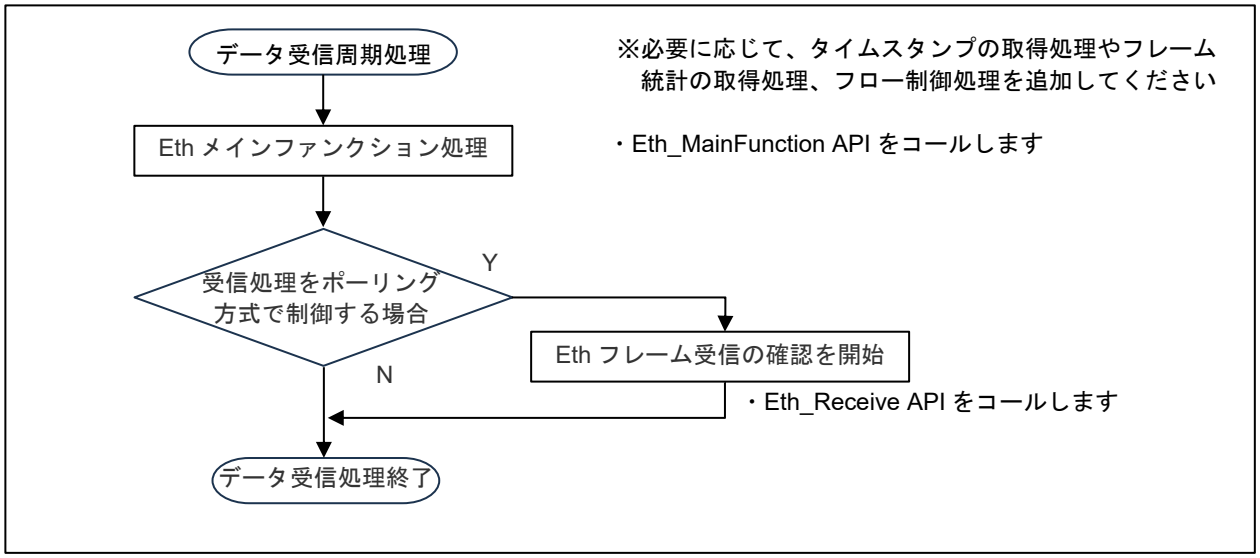


図 3-6 データ受信処理フロー例

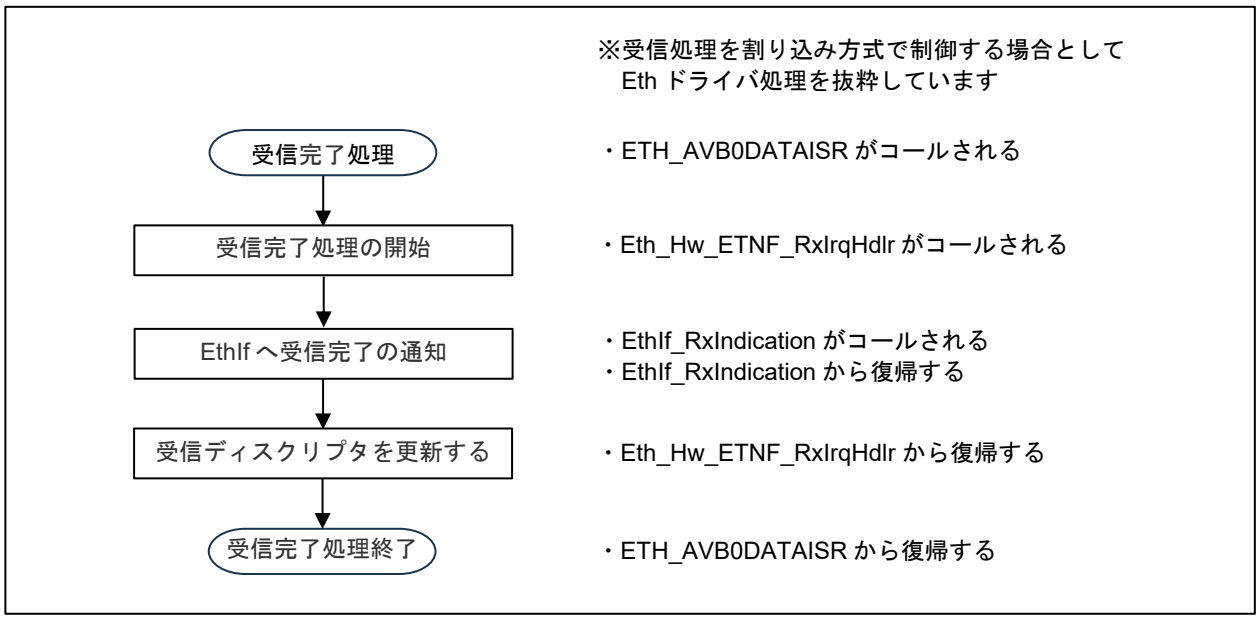


図 3-7 受信完了割り込み処理フロー

3.4 エラー検出処理

AUTOSAR MCAL の Eth ドライバの使用を前提に、10BASE-T1S のエラー検出処理をフローチャートで説明します。

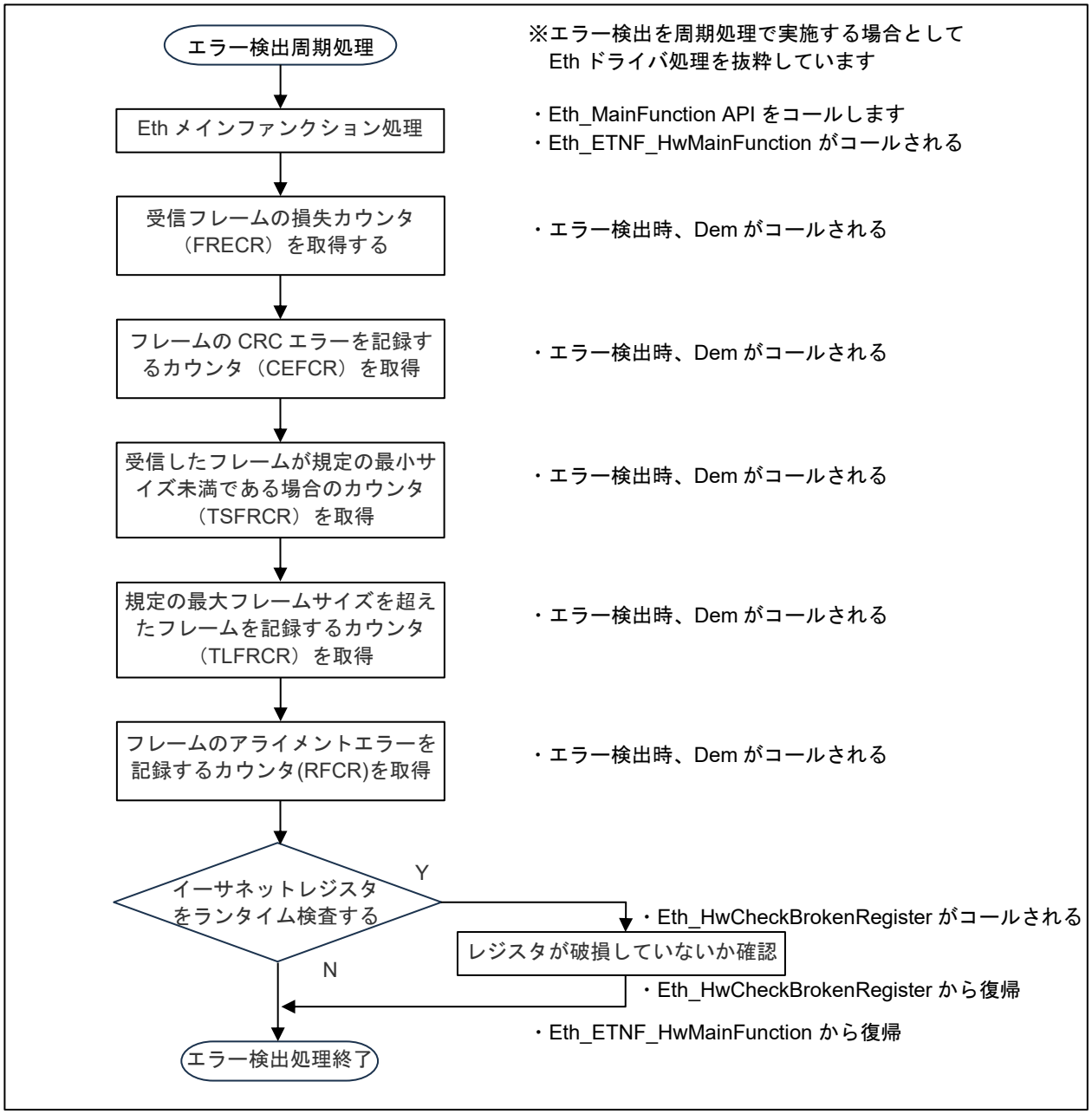


図 3-8 エラー検出処理フロー

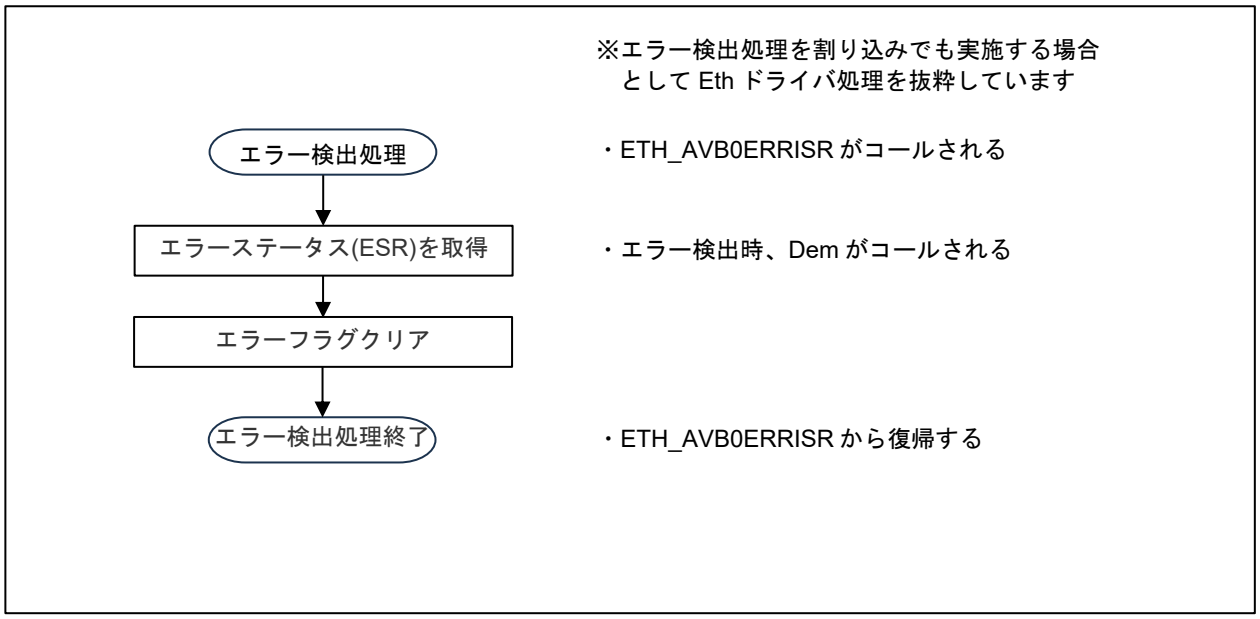


図 3-9 エラー割り込み処理フロー

4. サンプルソフト

ここでは、サンプルソフトの処理について説明します。

4.1 アプリケーション処理概要

サンプルアプリケーション処理の概要について説明します。

- Node ID0 のサンプルアプリ処理
  - 初期化処理
  - 一定期間を待機した後に Eth 送信処理開始（複数フレーム送信）※Node ID1 の Eth 受信可能待ち
  - Node ID1 からの Eth 受信完了確認処理（複数フレーム受信）
  - アプリ処理終了（自番地の無限ループ処理）
- Node ID1 のサンプルアプリ処理
  - 初期化処理
  - Eth 送信処理開始（1 フレーム送信）
  - Eth 受信完了待ち処理（複数フレーム受信）
  - Node ID0 からの Eth 受信確認後に Eth 送信処理開始（複数フレーム送信）
  - アプリ処理終了（自番地の無限ループ処理）

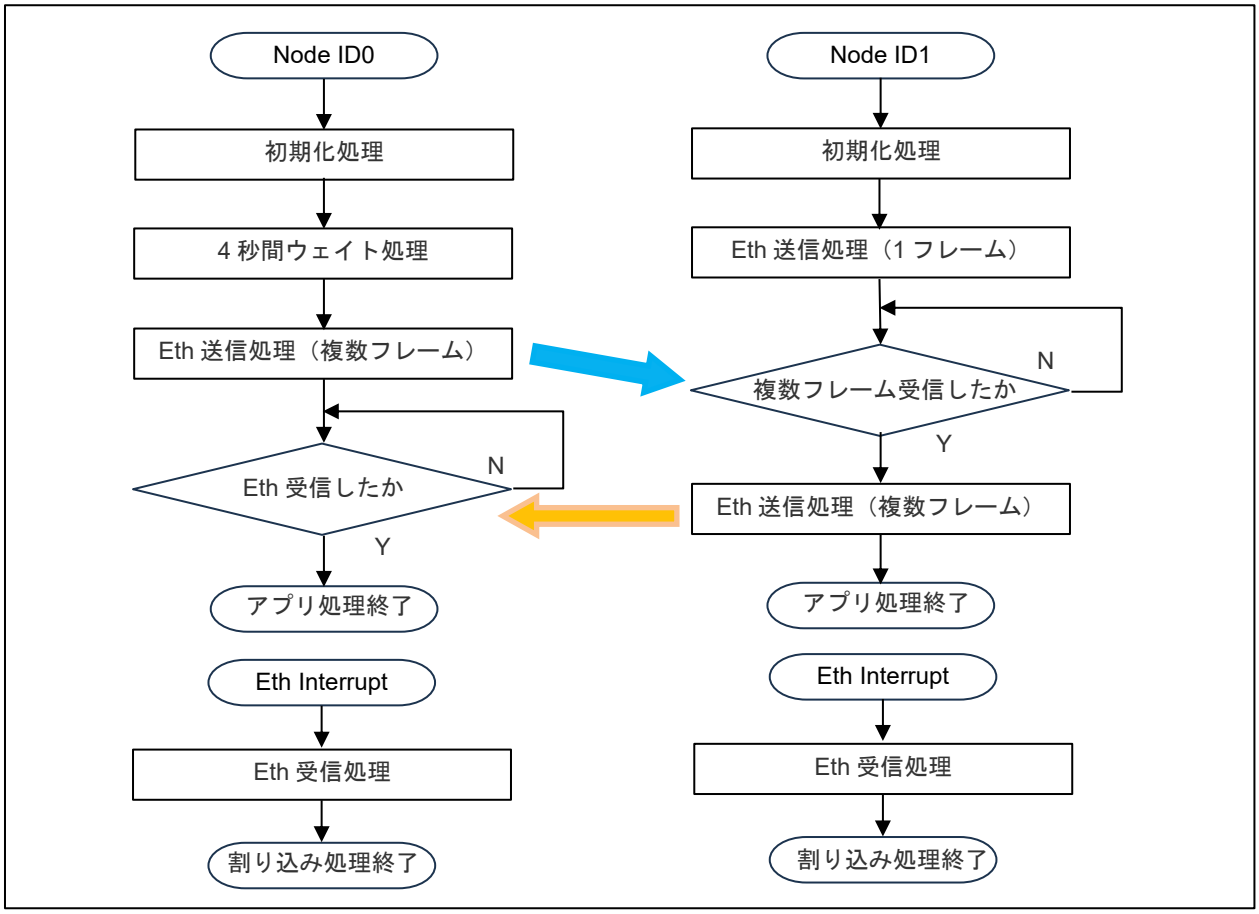


図 4-1 サンプルアプリケーション処理フロー

4.2 サンプルソフト処理内容説明

本サンプルソフトのファイル構成（抜粋）を一覧表示します。

表 4-1 ファイル構成（抜粋）

フォルダ名／ファイル名	コメント
\project_eth_10base_t1s_u2c\	カレントディレクトリ
├command.bat	cmd ウィンドウ起動用バッチファイル
├\Common\	共通ディレクトリ
├\mcal\	MCAL ディレクトリ ※Eth ドライバを使用する
├\node_id0\	Node ID0 格納ディレクトリ
│ ├run_node_id0.mtpj	Node ID0 の CS+プロジェクトファイル
│ ├T1S_SampleApp.bat	Node ID0 のビルド用バッチファイル
│ ├\include\	インクルードディレクトリ
│ │ └App_ETH_Common_Sample.h	サンプルアプリ共通 C インクルードファイル
│ ├\src\	C ソース格納ディレクトリ
│ │ └App_ETH_Common_Sample.c	サンプルアプリ共通 C ソースファイル
│ ├\stubs\	スタブ格納ディレクトリ※Dem, Det, EthIf など
│ └\U2C4\	RH850/U2U4 用 MCAL コンフィグ格納ディレクトリ
│ ├\22_11\	AUTOSAR R22-11 用 MCAL コンフィグ格納ディレクトリ
│ │ ├\config\	MCAL ETH コンフィグファイル格納ディレクトリ
│ │ │ └App_ETH_U2C4_702613FABB_Sample.arxml	R7F702613FABB 用 Eth サンプル arxml ファイル
│ │ ├\include\	MCAL ETH コンフィグヘッダ格納ディレクトリ
│ │ │ └Eth_Cfg.h	サンプルアプリ Eth コンフィグヘッダファイル (*1)
│ │ └\src\	MCAL ETH コンフィグ C ソース格納ディレクトリ
│ │ └Eth_PBcfg.c	サンプルアプリ Eth コンフィグ C ファイル (*1)
│ ├\obj\	サンプルアプリオブジェクト格納ディレクトリ
│ │ └\ghs\	サンプルアプリ ghs オブジェクト格納ディレクトリ
│ │ ├App_ETH_U2C4_Sample.map	サンプルアプリマップファイル (*1)
│ │ └App_ETH_U2C4_Sample.out	サンプルアプリ実行ファイル (*1)
│ ├\include\	サンプルアプリ C インクルードディレクトリ
│ │ └App_ETH_U2C4_Sample.h	サンプルアプリ C インクルードファイル
│ └\src\	サンプルアプリ C ソース格納ディレクトリ
│ └App_ETH_U2C4_Sample.c	サンプルアプリ C ソースファイル
└\node_id1\	Node ID1 のディレクトリ※以降は ID0 と同様

(*1) ビルドで生成されるファイル

次に、サンプルソフトの処理（Ethernet 以外の処理含む）について説明します。

- スタートアップ処理の手順は以下の通りです。

  (1) RAM の初期化

  (2) スタックの設定

  (3) 割り込みベクタの設定

  (4) 標準ライブラリのセットアップ

- クロック初期化設定の手順は以下の通りです。

  (1) HS IntOSC の発振が安定するまで待機します。

  (2-1) Main OSC を有効化します。

  (2-2) Main OSC の発振が安定するまで待機します。

(3-1) PLL を有効化します。

(3-2) PLL の発振が安定するまで待機します。

(3-3) システムクロックのギアアップシーケンス（[REF02] 13.6.4 章）に基づき、PLL の周波数を 800MHz、SSCG の周波数を 640MHz まで引き上げます。

- 割り込み初期化設定の手順は以下の通りです。

(1) INTETNF0DATA、INTETNF0ERR、INTETNF0MNG、INTETNF0MAC の割り込みを有効化します。

(2) ETNF モジュールのクロック供給を有効化します。

(3) 全体割り込みを有効化します。

- ポート初期化設定の手順は以下の通りです。

(1) Eth 関連の PCR20_4、PCR20_3、PCR20_13 のポート制御レジスタを設定します。

(2) P20_3、P20_4、P20_13 のポートレジスタで 'H'レベルに設定します。

- Ethernet 初期化設定の手順は以下の通りです。

(1) Eth_Init を呼び出して、Ethernet ドライバを初期化します（ETNF モジュールの初期化）。

(2) Eth_InitT1s を呼び出して、Ethernet トランシーバを初期化します（ETNF 内蔵の PMA/PCS/PLCA と外付け PHY の初期化）。

(3) Eth_ReadMii を呼び出して、PHY デバイスの状態をチェックします。

- アプリケーション処理として Eth 送信の手順は以下の通りです。

(1-1) Eth_SetPhysAddr を呼び出して、MAC アドレスを設定します。

(1-2) Eth_GetPhysAddr を呼び出して、MAC アドレスを検証します。

(1-3) Eth_UpdatePhysAddrFilter を呼び出して、アドレスフィルタを更新します。

(2-1) Eth_SetControllerMode を呼び出して、コントローラ モードを ACTIVE に設定（ETNF で動作モードへ遷移）します。

(2-2) Eth_GetControllerMode を呼び出して、現在の動作モードの状態をチェックします。

(3) PHY ノード ID0 の場合、一定時間待機処理を実行します。(PHY ノード ID1 の起動待ち)

(4) UDP フレームをブロードキャスト送信します。

・Eth_ProvideTxBuffer を呼び出して、使用可能なバッファを取得します。

・送信するデータをバッファにコピーします。

・Eth_Transmit を呼び出して、フレームを送信します。

(5) PHY ノード ID1 の場合、Ethernet 受信完了待ち（2 フレーム）します。

(6) IEEE1722（ユーザ定義 ACF メッセージ）フレームをユニキャスト送信します。

・Eth_ProvideTxBuffer を呼び出して、使用可能なバッファを取得します。

・送信するデータをバッファにコピーします。

・Eth_Transmit を呼び出して、フレームを送信します。

(7) Eth メッセージを受信したら、Eth_GetCounterValues、Eth_GetTxErrorCounterValues、Eth_GetRxStats、Eth_GetTxStats を呼び出してフレーム統計情報を収集します。

(8-1) フレーム統計が期待通りであれば、アプリケーションを正常終了する。

(8-2) フレーム統計が期待通りでなければ、アプリケーションを異常終了する。

- Eth 割り込み処理にて Eth 受信の手順は以下の通りです。

(1-1) MAC アドレスを取得します。

(1-2) ブロードキャストかユニキャストか判定し、フレーム統計としてカウントします。

(2) フレームタイプを取得します。

(3) レングスを取得します。

(4) EthIf_RxIndication を呼び出して、フレーム統計情報を更新します。

以下に、本サンプルソフトのノード ID と MAC アドレスの紐付けを一覧表示します。

表 4-2 サンプルソフトの MAC アドレス

Ethernet ノード ID	MAC アドレス
0	74:90:50:00:00:00
1	74:90:50:00:00:01

本サンプルソフトで使用する Ethernet トランシーバ、および Eth ドライバのコンフィグ・パラメータ設定値を一覧表示します。

表 4-3 Ethernet トランシーバの設定(1/2)

定義名	設定値	ソースコメント
T1S_PHY_ADDR	トランシーバの PHY アドレスを設定する。 設定値 : 0x01U	/* トランシーバの PHY アドレスを指定 */
ETH_PHY_ADDR	マイコン内蔵 PHY アドレスを設定する。 設定値 : 0x1FU	/* 内蔵 PHY アドレスを指定 */
T1S_PHY_CONTROL_CFG	CONTROL レジスタを設定する。 設定値 : T1S_PHY_REG000000_LOOP_OFF T1S_PHY_REG000000_LCTL_ON T1S_PHY_REG000000_LPWR_OFF(固定) T1S_PHY_REG000000_ISOM_OFF(固定) T1S_PHY_REG000000_CTEST_OFF	/* CONTROL レジスタのコンフィグ設定 */ /* ループバックしない */ /* PHY リンク制御を有効: 通常動作開始 */ /* 低電力モードに入らない */ /* 分離モードに入らない */ /* 衝突実行モード無効: 通常動作 */
T1S_PHY_T1SPMACTRL_CFG	T1SPMACTRL レジスタを設定する。 設定値 : T1S_PHY_REG2108F9_LPWR_OFF(固定) T1S_PHY_REG2108F9_LOOP_OFF	/* T1SPMACTRL レジスタのコンフィグ設定 */ /* 低電力モードにしない */ /* ループバックモードにしない */
T1S_PHY_T1SPMATSTM_CFG	T1SPMATSTM レジスタを設定する。 設定値 : T1S_PHY_REG2108FB_NORMAL	/* T1SPMATSTM レジスタのコンフィグ設定 */ /* Normal operation: 通常動作 */
T1S_PHY_T1SPCSCTRL_CFG	T1SPCSCTRL レジスタを設定する。 設定値 : T1S_PHY_REG2308F3_LOOP_OFF	/* T1SPCSCTRL レジスタのコンフィグ設定 */ /* ループバックにしない */
T1S_PHY_T1STWEAKS_CFG	T1STWEAKS レジスタを設定する。 設定値 : 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	/* T1STWEAKS レジスタのコンフィグ設定 */ /* TX フレームが MII RX に反映されない */ /* 拡張ノイズ耐性モードにしない */ /* 自動回復しない */ /* RZ エンコードされていると見なされる */ /* PCS スクランブル機能が無効 */ /* 衝突検出はマスクされない */ /* MII RX 信号が遅延される */
T1S_PHY_T1SPLCAEXT_CFG	T1SPLCAEXT レジスタを設定する。 設定値 : T1S_PHY_REG3F8002_PREN_OFF T1S_PHY_REG3F8002_LDEN_OFF T1S_PHY_REG3F8002_LDR_OFF	/* T1SPLCAEXT レジスタのコンフィグ設定 */ /* PLCA RS は標準モードで動作 */ /* PLCA リーダはノード ID によって選択 */ /* LDEN=1 であれば、PLCA スレーブとする */
T1S_PHY_T1SPLCAEXT_CFG	T1SPLCAEXT レジスタを設定する。 設定値 : T1S_PHY_REG3F8002_PREN_OFF T1S_PHY_REG3F8002_LDEN_OFF T1S_PHY_REG3F8002_LDR_OFF	/* T1SPLCAEXT レジスタのコンフィグ設定 */ /* PLCA RS は標準モードで動作 */ /* PLCA リーダはノード ID によって選択 */ * LDEN=1 であれば、PLCA スレーブとする */
T1S_PHY_T1SPMATUNE0_CFG	T1SPMATUNE0 レジスタを設定する。 設定値 : T1S_PHY_REG3F8003_NNTHR T1S_PHY_REG3F8003_DRFTW	/* T1SPMATUNE0 レジスタのコンフィグ設定 */ /* NN コンパレータのしきい値を設定 */ /* ドリフト補償器の時間ウィンドウを設定 */
T1S_PHY_T1SPMATUNE1_CFG	T1SPMATUNE1 レジスタを設定する。 設定値 : T1S_PHY_REG3F8004_JJHHTHR T1S_PHY_REG3F8004_JJTHR	/* T1SPMATUNE1 レジスタのコンフィグ設定 */ /* JJHH コンパレータのしきい値を設定 */ /* JJ コンパレータのしきい値を設定 */

表 4-4 Ethernet トランシーバの設定(2/2)

定義名	設定値	ソースコメント
T1S_PHY_PLCACTRL0_CFG	PLCACTRL0 レジスタを設定する。 設定値 : T1S_PHY_REG3FCA01_EN_ON	/* PLCACTRL0 レジスタのコンフィグ設定 */ /* PLCA RS 機能が有効 */
T1S_PHY_PLCACTRL1_CFG	PLCACTRL1 レジスタを設定する。 設定値 : T1S_PHY_REG3FCA02_NCNT T1S_PHY_REG3FCA02_ID	/* PLCACTRL1 レジスタのコンフィグ設定 */ /* ノードの最大数を設定 */ /* PLCA ノード ID を設定 */
T1S_PHY_REG3FCA02_NCNT	NCNT を設定する。 設定値 : 8	-
T1S_PHY_REG3FCA02_ID	ID を設定する。 設定値 : 0 or 1 ※プロジェクトにより異なる。	-
T1S_PHY_PLCATOTMR_CFG	PLCATOTMR レジスタを設定する。 設定値 : T1S_PHY_REG3FCA04_TOTMR	/* PLCATOTMR レジスタのコンフィグ設定 */ /* PLCA 送信機会タイマを設定 */
T1S_PHY_PLCABURST_CFG	PLCABURST レジスタを設定する。 設定値 : T1S_PHY_REG3FCA05_MAXBC T1S_PHY_REG3FCA05_BTMR	/* PLCABURST レジスタのコンフィグ設定 */ /* 送信できる追加のパケット数を設定 */ /* バースト連結する待機時間を設定 */
T1S_PHY_ADDR1_REG18H_SET_CFG	XCVR_ANALOG_SET_2 のセットビット値を設定する。 設定値 : 0x6000U	/* XCVR_ANALOG_SET_2 設定 */
T1S_PHY_ADDR1_REG18H_CLR_CFG	XCVR_ANALOG_SET_2 のクリアビット値を設定する。 設定値 : ~0x0000U	-
T1S_PHY_ADDR1_REG1FH_SET_CFG	XCVR_ANALOG_SET_9 のセットビット値を設定する。 設定値 : 0x0018U	/* XCVR_ANALOG_SET_9 設定 */
T1S_PHY_ADDR1_REG1FH_CLR_CFG	XCVR_ANALOG_SET_9 のクリアビット値を設定する。 設定値 : ~0x0000U	-

表 4-5 Eth ドライバの設定(1/3)

パラメータ名（ツリー表記）	設定値	概要
/Renesas/	-	-
└/EcucDefs_Eth/	-	-
└/Eth/	-	-
├/EthGeneral/	-	-
│ ├ EthDevErrorDetect	false	開発エラーの検出と通知の有効/無効
│ ├ EthMultiCoreSupport	false	マルチコア サポート モードの有効/無効
│ ├ EthGlobalTimeSupport	false	GlobalTime API の有効/無効
│ ├ EthIndex	0	インスタンス ID の指定
│ ├ EthMainFunctionPeriod	0.001	Eth_MainFunction 周期を秒単位で指定
│ ├ EthMaxCtrlsSupported	2	コントローラの合計数
│ ├ EthVersionInfoApi	false	バージョン情報 API の有効/無効
│ ├ EthVersionCheckExternalModules	false	AUTOSAR バージョン チェック有効/無効
│ ├ EthCriticalSectionProtection	true	ETH ドライバの CPU 負荷軽減の指定
│ ├ EthEnableInputClockRefImmediateValue	true	PBUS の直接入力クロック値の有効/無効
│ ├ EthInputClockRefImmediateValue	100000000	ETH マクロに設定される PBUS クロック
│ ├ EthTimeout	0.012	タイムアウト時間の指定
│ ├ EthGetRxStatsApi	true	Eth_GetRxStats API の有効/無効
│ ├ EthGetTxErrorCounterValuesApi	true	Eth_GetTxErrorCounterValues API の有効/無効
│ ├ EthGetCounterValuesApi	true	Eth_GetCounterValues API の有効/無効
│ ├ EthInterruptConsistencyCheck	false	ISR 割り込み一貫性チェックの有効/無効
│ ├ EthGetTxStatsApi	true	Eth_GetTxStats API の有効/無効
│ ├ EthSwitchManagementSupport	false	Ethswt 管理の有効/無効
│ ├ EthDeInitApi	true	Eth_DeInit API の有効/無効
│ ├ EthQosSupport	true	送信をサポートする API の有効/無効
│ ├ EthRegisterCheckInitTime	false	Eth_Init でのレジスタチェックの有効/無効
│ ├ EthRegisterCheckRunTime	false	Eth_MainFunction でのレジスタチェックの有効/無効
│ ├ EthUpdatePhysAddrFilter	true	Eth_UpdatePhysAddrFilter API 有効/無効
│ ├ EthGetDropCountApi	false	Eth_GetDropCount API の有効/無効
│ ├ EthGetEtherStatsApi	false	Eth_GetEtherStats API の有効/無効
│ ├ EthDeviceName	R7F702613FABB	デバイス名
│ ├ EthIsrCategory	CAT1	モジュールごとの ISR カテゴリ
│ └/EthCtrlOffloading/	-	-
│ ├ EthCtrlEnableOffloadChecksumIPv4	false	チェックサム不一致の IPv4 フレーム破棄
│ ├ EthCtrlEnableOffloadChecksumICMP	false	チェックサム不一致の ICMP フレーム破棄
│ ├ EthCtrlEnableOffloadChecksumTCP	false	チェックサム不一致の TCP フレーム破棄
│ └ EthCtrlEnableOffloadChecksumUDP	false	チェックサム不一致の UDP フレーム破棄
└/EthConfigSet/	-	-
└/EthCtrlConfig/	-	-
├ EthBeTimeStampStore	false	タイムスタンプ情報の組み込み有効/無効
├ EthCtrlConfigSwBufferHandling	false	SW バッファ管理の有効/無効
├ EthCtrlEnableMii	true	トランシーバ アクセスの MII 有効/無効
├ EthCtrlEnableRxInterrupt	true	受信割り込みの有効/無効
├ EthCtrlEnableSpiInterface	false	SPI インタフェースの有効/無効
├ EthCtrlEnableTxInterrupt	true	送信割り込みを有効/無効にし、Eth_TxConfirmation API の有効/無効
├ EthCtrlIdx	0	コントローラのインスタンス ID の指定
├ EthEnableCBS	true	クレジット ベース シェーピング有効/無効
├ EthInternalLoopBack	false	内部ループバック モード
├ EthNwCntrlFiltering	false	ネットワークフィルタリングの有効/無効
├ EthRamSize	102400	ユーザ RAM のサイズ
├ EthSRPTalkerFiltering	true	個別フィルタリング機能の有効 / 無効
├ EthStreamTimeStampStore	false	タイムスタンプ情報付加の有効/無効
├ EthEnableCRSCheck	false	CRS アサート チェックの有効/無効

表 4-6 Eth ドライバの設定(2/3)

パラメータ名（ツリー表記）	設定	概要
├ EthCtrlMacLayerSpeed	ETH_MAC_LAYER_SPEED_10M	MAC 層のボー レートの定義
├ EthCtrlMacLayerType	ETH_MAC_LAYER_TYPE_XMII	MAC 層タイプの定義
├ EthCtrlPhyAddress	74:90:50:00:00:00 (*2)	MAC アドレスのバイトオーダ指定
├ EthDuplexMode	ETH_T1S_MODE	デュプレックスモード設定を指定
├ EthStreamFiltering	ETH_AVBNMQUE0	ストリーム フィルタリング オプションの指定
├ EthTxQuePriority	ETH_AVBDEF	送信同期モード 0 (ベストエフォート) の指定
├ EthUnitSelection	ETH_ETNF0	イーサネット TSN およびイーサネット AVB ユニットの選択
├ EthCtrlMacLayerSubType	STANDARD	
├ EthPulseGenerationMode	ETH_PPS_GPTP_TIMER_VALUE	1 秒あたりのパルス生成モードの設定
├/EthCtrlConfigEgress/	-	-
│ ├/EthCtrlConfigEgressQueue/	-	-
│ │ ├ EthCtrlConfigEgressQueueBufLenByte	0	Egress FIFO 長。U2C 未サポート
│ │ ├ EthCtrlConfigEgressQueueBufTotal	0	Egress FIFO バッファ数。U2C 未サポート
│ │ └ EthCtrlConfigEgressQueueIdx	0	Egress FIFO インデックス。U2C 未サポート
│ ├/EthCtrlConfigScheduler/	-	-
│ │ └/EthCtrlConfigSchedulerPredecessor/	-	-
│ │ └ EthCtrlConfigSchedulerPredecessorOrder	0	スケジューラの順序を定義。U2C 未サポート
│ └/EthCtrlConfigShaper/	-	-
│ ├ EthCtrlConfigShaperMaxCredit	0	キューに蓄積できるクレジットの最大量。U2C 未サポート
│ └ EthCtrlConfigShaperMinCredit	0	キューに蓄積できるクレジットの最小量。U2C 未サポート
├/EthCtrlPriority/	-	-
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping）	-（多重化）
│ │ └ EthCtrlPriorityValue	0	特定トラフィック クラス優先順位
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_001）	-（多重化）
│ │ └ EthCtrlPriorityValue	1	特定トラフィック クラス優先順位
│ ├/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_002）	-（多重化）
│ │ └ EthCtrlPriorityValue	2	特定トラフィック クラス優先順位
│ └/EthCtrlPriorityMapping/	（EthCtrlPriorityMapping_003）	-（多重化）
│ └ EthCtrlPriorityValue	3	特定トラフィック クラス優先順位
├/EthMacConfig/	-	-
│ ├/EthFlowControlConfig/	-	-
│ │ ├ EthPauseFrame	false	ポーズフレーム制御
│ │ ├ EthPauseFrameRetransmissionTime	0	ポーズフレーム再送時間
│ │ ├ EthPauseTime	1	ポーズ時間
│ │ └ EthPauseTimeZero	false	TIME = 0 でポーズフレームの送信と受信の有効/無効
│ └ EthLinkVerificationTime	10	リンク検証タイマ。ミリ秒で定義
├/EthPhyConfig/	-	-
│ ├ EthMdcClockSelection	19	MDC のクロック サイクルの指定
│ ├ EthMdioCaptureTime	MDIO_CAPTURE_TIME_1_CLK_CYCLE	MDIO 通信のキャプチャ時間の指定
│ └ EthMdioHoldTime	MDIO_HOLD_TIME_1_CLK_CYCLE	MDIO 通信のホールド時間の指定
├/EthRxCommonConfig/	-	-
│ ├ EthRxMaxFrameSize	1518	受信の最大フレーム サイズの指定

表 4-7 Eth ドライバの設定(3/3)

パラメータ名（ツリー表記）	設定	概要
│ ├/EthTasCommonConfig/	-	-
│ │ ├ EthTasEnable	false	TAS ゲートの設定
│ │ └ EthTasTransmissionJitter	75	TAS 送信ジッタ [%]
│ └ EthFragmentSize	ETH_FRAGMENT_64_BYTE	送信フラグメントサイズ設定
├/EthTxQueueConfig/	（EthTxQueueConfig）	-（多重化）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	対応キューの優先度アルゴリズムの選択
│ ├ EthTxQueueBufs	8	送信ディスクリプタ数の設定
│ ├ EthTxQueueIdx	0	有効化する送信キューのインデックス
│ ├ EthTxQueueMaxFrameSize	1518	各送信キューの最大フレーム サイズ指定
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Tx キューのフレーム タイプの選択
├/EthTxQueueConfig/	（EthTxQueueConfig_001）	-（多重化）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	対応キューの優先度アルゴリズムの選択
│ ├ EthTxQueueBufs	8	送信ディスクリプタ数の設定
│ ├ EthTxQueueIdx	1	有効化する送信キューのインデックス
│ ├ EthTxQueueMaxFrameSize	1518	各送信キューの最大フレーム サイズ指定
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Tx キューのフレーム タイプの選択
├/EthTxQueueConfig/	（EthTxQueueConfig_002）	-（多重化）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ ├ EthCtrlTxQueueBwFraction	10	キューに割り当てられた帯域幅の割合[%]
│ │ └ EthTxQueuePolicy	ETH_CBS	対応キューの優先度アルゴリズムの選択
│ ├ EthTxQueueBufs	8	送信ディスクリプタ数の設定
│ ├ EthTxQueueIdx	2	有効化する送信キューのインデックス
│ ├ EthTxQueueMaxFrameSize	1518	各送信キューの最大フレーム サイズ指定
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Tx キューのフレーム タイプの選択
├/EthTxQueueConfig/	（EthTxQueueConfig_003）	-（多重化）
│ ├/EthCtrlTxQueueShaper/	-	-
│ │ └ EthTxQueuePolicy	ETH_NONE	対応キューの優先度アルゴリズムの選択
│ ├ EthTxQueueBufs	8	送信ディスクリプタ数の設定
│ ├ EthTxQueueIdx	3	有効化する送信キューのインデックス
│ ├ EthTxQueueMaxFrameSize	1518	各送信キューの最大フレーム サイズ指定
│ └ EthTxQueueFrameType	ETH_EXPRESS_FRAME	Tx キューのフレーム タイプの選択
├/EthRxQueueConfig	（EthRxQueueConfig）	-（多重化）
│ ├ EthRxQueueBufs	8	受信ディスクリプタ数の設定
│ ├ EthRxQueueIdx	0	有効化する受信キューのインデックス
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	フィルタリングに使用するパターン アドレスの指定
├/EthRxQueueConfig	（EthRxQueueConfig_001）	-（多重化）
│ ├ EthRxQueueBufs	8	受信ディスクリプタ数の設定
│ ├ EthRxQueueIdx	1	有効化する受信キューのインデックス
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	フィルタリングに使用するパターン アドレスの指定
├/EthRxQueueConfig	（EthRxQueueConfig_002）	-（多重化）
│ ├ EthRxQueueBufs	8	受信ディスクリプタ数の設定
│ ├ EthRxQueueIdx	2	有効化する受信キューのインデックス
│ └ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	フィルタリングに使用するパターン アドレスの指定
└/EthRxQueueConfig	（EthRxQueueConfig_003）	-（多重化）
├ EthRxQueueBufs	8	受信ディスクリプタ数の設定
├ EthRxQueueIdx	3	有効化する受信キューのインデックス
└ EthPatternStream	74:90:50:00:00:01:00:00 (*2)	フィルタリングに使用するパターン アドレスの指定

(*2) Node ID=0 のパラメータ値

付録.A サンプルアプリケーションコード

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)0U == 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_PLLO */
    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                                     **
*****************************************************************************/
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2025.5.27	-	初版
1.11	2025.11.11	-	RH850/U2C シリーズと U2B-E シリーズを統合

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/.