

RZ/T2M, RZ/N2L, RZ/T2N Group

TSN Sample application Package

要旨

本書は RZ/T2M, RZ/N2L 上で動作する TSN(Time-Sensitive Networking)を実現するサンプルアプリケーションについて、システム概要、対応機能、および TSN を評価するための基本的な使用方法を説明します。

動作確認デバイス

RZ/T2M, RZ/N2L, RZ/T2N

目次

1.	概説	3
1.1	略語/定義	4
1.2	参照	5
1.3	制限事項 / 既知の問題	5
2.	特徴	6
2.1	IEEE802.1AS (gPTP)	6
2.2	IEEE802.1Qbv (TDMA)	7
2.3	IEEE802.1Qbu (Frame Preemption)	7
2.4	IEEE802.1Qci (PSFP)	7
2.5	IEEE802.1Qch (CQF).....	7
2.6	IEEE802.1AB(LLDP)	7
3.	開発環境	8
3.1	開発時の環境	8
3.2	ハードウェア要件.....	9
3.2.1	RZ/T2M-RSK,RZ/N2L-RSK のハードウェア設定	10
3.2.2	RZ/T2M-RSK,RZ/N2L-RSK と接続する	24
3.3	ソフトウェア要件.....	29
3.3.1	サンプルアプリケーションのダウンロードに必要なソフトウェア	29
3.3.2	サードパーティーソフトウェア	33
4.	サンプルアプリケーションの実行.....	34
4.1	ビルドとダウンロード	34
4.1.1	プロジェクトを開く	34
4.1.2	デバイスの事前設定	38
4.1.3	ビルドと RZ/T2M-RSK,RZ/N2L-RSK へのダウンロード.....	48
4.2	サンプルアプリケーションの起動	52
4.2.1	EWARM 使用時	52
4.2.2	e ² studio 使用時	52
4.3	サンプルアプリケーションの動作確認の例	53

4.4	サンプルアプリケーションの動作確認の例(Qch 3 台構成)	54
5.	TSN サンプルアプリケーション	55
5.1	アプリケーション動作概要	55
5.1.1	メニュー	57
5.1.2	gPTP 動作仕様	59
5.1.3	TDMA 動作仕様	60
5.1.4	RT フレーム・NRT フレーム動作仕様	61
5.1.5	Frame Preemption 動作仕様	61
5.1.6	PSFP 動作仕様	64
5.1.7	CQF 動作仕様	66
5.1.8	LLDP 動作仕様	69
5.2	アプリケーション動作確認内容	70
5.2.1	ログ出力内容	70
5.2.2	ピン出力内容	77
5.3	アプリケーション構成	90
5.3.1	モジュール構成	90
5.3.2	注意事項 – サンプルアプリケーションのタスク優先度	94
	改訂記録	95

1. 概説

本書では、RZ/T2M, RZ/N2L, RZ/T2N 向けの TSN サンプルアプリケーションの概要を示します。

TSN(Time-Sensitive Networking)とは、リアルタイムトラフィックを実装しつつ同一ネットワーク上で他のベストエフォート型トラフィックを同時に運用させることを可能とするネットワークシステムのことを指します。

時刻同期をはじめとする仕組みを利用することで、TSN は標準的な Ethernet ベースのネットワーク上で決定論的な通信動作を実現可能とします。

本サンプルアプリケーションは、ターゲットデバイス上で TSN 関連機能をどのように利用できるかを示し、Ethernet スイッチ機能および関連するソフトウェアコンポーネントを用いた TSN 通信の基本設定や動作および評価手順の理解を助けることを目的としています。

本サンプルアプリケーションには、時刻同期、スケジュール型トラフィック送信、フレームプリエンブション、トラフィック制御、周期キューイング、およびデバイス情報通知の機能が含まれています。

Ethernet スイッチモジュールなどのハードウェア機能と IEEE802.1 AS ソフトウェアスタックおよび LLDP 関連処理といったソフトウェアコンポーネントを組み合わせることで提供されるこれらの機能が、決定論的な通信にどのように寄与するかを確認できます。

実際にサンプルアプリケーションをボード上で動作させる際は、本書に従ってボード及びサンプルアプリケーションの設定を行ってご使用ください。

1.1 略語/定義

表 1-1 略語/定義

Index	Abbreviations /Definitions	Description
1	IP	Internet Protocol
2	TCP	Transmission Control Protocol
3	USB	Universal Serial Bus
4	PC	Personal Computer
5	SW	Switch
6	GPIO	General Purpose Input/Output
7	EWARM	Embedded Workbench® for ARM
8	lwIP	lightweight IP
9	xSPI	Expanded Serial Peripheral Interface
10	QSPI	Quad Serial Peripheral Interface
11	フラッシュ	xSPI を使用するフラッシュデバイス
12	ETHSW	Ethernet Switch
13	GMAC	Ethernet MAC
14	TSN	Time-Sensitive Networking
15	IEEE	Institute of Electrical and Electronics Engineers
16	PTP / gPTP	Precision Time Protocol / Generalized Precision Time Protocol
17	TDMA	Time Division Multiple Access (IEEE802.1Qbv)
18	IDE	Integrated Development Environment
19	FSP	Flexible Software Package
20	NOR	Not OR
21	RAM	Random Access Memory
22	UART	Universal Asynchronous Receiver Transmitter
23	MAC	Media Access Control
24	RT	Real Time
25	NRT	Non-Real Time
26	UDP	User Datagram Protocol
27	CN	Connector
28	CH	Channel
29	JN	Jumper N
30	PSFP	Per-Stream Filtering and Policing (IEEE802.1Qci)
31	CQF	Cyclic Queuing and Forwarding (IEEE802.1Qch)
32	LLDP	Link Layer Discovery Protocol (IEEE 802.1AB)
33	MIB	Management Information Base
34	SNMP	Simple Network Management Protocol

1.2 参照

各ボードに関する技術情報は、ルネサスエレクトロニクスの Web ページから入手することができます。

表 1-2 RZ/T2M 技術情報

Index	Technical Inputs
1	Renesas Starter Kit+ for RZ/T2M, RZ/T2ME ユーザーズマニュアル r20ut4939jgxxx-rskplus-rzt2m-v1-um.pdf
2	RZ/T2M グループ ユーザーズマニュアル ハードウェア編 r01uh0916jjxxx-rzt2m.pdf
3	RZ/T2, RZ/N2 Getting Started with Flexible Software Package r01an6434ejxxx-rzt2-rzn2-fsp-getting-started.pdf

表 1-3 RZ/N2L 技術情報

Index	Technical Inputs
1	Renesas Starter Kit + for RZ/N2L ユーザーズマニュアル r20ut4984jgxxx-rskplus-rzn2l-v1-um.pdf
2	RZ/N2L グループ ユーザーズマニュアル ハードウェア編 r01uh0955jjxxx-rzn2l.pdf
3	RZ/T2, RZ/N2 Getting Started with Flexible Software Package r01an6434ejxxx-rzt2-rzn2-fsp-getting-started.pdf

表 1-4 RZ/T2N 技術情報

Index	Technical Inputs
1	RZ/T2N Evaluation Board ユーザーズマニュアル r20ut5573jjxxx-rzt2nevb(EVB1_537)_20251107.pdf
2	RZ/T2N Group User's Manual: Hardware r01uh1084ejxxx-rzt2n.pdf
3	RZ/T, RZ/N Series Getting Started with RZ Flexible Software Package r01an6434ej0420-rzt-rzn-series-fsp-getting-started.pdf

1.3 制限事項 / 既知の問題

- e²studio 使用時に、timeReceiver として動作しているデバイスの Ethernet ケーブルの挿抜を繰り返すと、"Sync-TimeOut"ログメッセージ出力直後にハングアップする場合があります。
- RZ/T2M-RSK は Qci 機能をサポートしていません。
そのため、Qci 関連のサンプルおよびそれらを用いたテストは実施不可能です。

2. 特徴

本アプリケーションにおける TSN は、以下の要素によって構成されます。

- IEEE802.1AS(gPTP)
 - A) ネットワーク上のデバイス間で時刻を同期します。本サンプルアプリケーションでは、時刻認識型 TSN 動作の基盤として使用されます。
- IEEE802.1Qbv (TDMA)
 - A) 定義されたスケジュールに従って Ethernet フレームの送信タイミングを制御します。この機能により、指定された時間枠内でのスケジュール型トラフィック送信が可能になります。
- IEEE802.1Qbu (Frame Preemption)
 - A) 高優先度フレームが低優先度フレームの送信を割り込みできるようにします。この機能により、時間依存性の高いトラフィックの遅延を低減できます。
- IEEE802.1Qci (PSFP)
 - A) ストリーム単位でトラフィックのフィルタリングを行います。この機能により、異常または予期しないトラフィックを制御し、TSN 通信の信頼性を高めます。
- IEEE802.1Qch (CQF)
 - A) 周期的なキュー動作に基づいてフレームを転送します。この機能により、確定的なネットワーク伝送を実現します。

また、本アプリケーションには以下の機能も包含しています。

- IEEE802.1AB(LLDP)
 - A) 接続デバイスやネットワーク構成に関する情報の通知および取得を行います。

2.1 IEEE802.1AS (gPTP)

gPTP は時刻同期を行うプロトコルである IEEE1588(PTP)におけるプロファイルの一つです。

gPTP による時刻同期は以下のように実現されます。

- gPTP 開始時、各デバイスは時刻情報の送信元である "timeTransmitter" (GrandMaster デバイス)と、それを受け取る "timeReceiver" (GrandMaster 以外のデバイス)に分けられます。
- timeTransmitter となったデバイスが時刻情報の発信源となります。
- timeReceiver が timeTransmitter から受け取った時刻情報をもとに自身の時刻を調整することで、時刻を同期させます。

gPTP には以下の特徴があります：

- 高精度な時刻同期
 - A) マイクロ秒以下の高精度時刻同期が可能です。
- 伝搬遅延測定
 - A) 時刻情報の送受信にかかる伝送時間も計算するため、通信経路による情報伝達速度の異なる状況下でも正確な時刻同期が可能です。
- 自動 timeTransmitter 選択機能
 - A) ネットワーク内の gPTP 対応デバイスの中から、最も高精度のクロックを持つデバイスを自動で timeTransmitter に設定することが可能です。

本サンプルアプリケーションでは、IEEE802.1AS-2020 に準拠した Renesas 製 gPTP ライブラリである IEEE 802.1AS Software Stack を使用して実装しています。

2.2 IEEE802.1Qbv (TDMA)

TDMA(時分割多元接続)方式は、通信トラフィックの送信タイミングを時間ごとに割り当てる方式です。

TDMA の動作は以下のように実現されます。

- 特定の周期を持つ TDMA 用サイクルを任意の時間ごとで分割します。
- 分割した時間ごとに各種類のトラフィックを割り当て、その期間内だけ送信されるようにします。

これにより各トラフィックは決められた間隔で送信されるようになります。

TDMA の実現によって、複数のトラフィックの最大遅延時間を保証しつつ多角的に運用することができるようになります。

これにより、TSN の実現を可能とします。

【注】 TDMA 動作の実現には、gPTP などによって通信デバイス間の時刻同期が保証されている必要があります。

2.3 IEEE802.1Qbu (Frame Preemption)

Frame Preemption は高優先度のフレームを送信するときに、低優先度のフレームが送信中でもそれを打ち切って高優先度のフレームを送信する機能です。

2.4 IEEE802.1Qci (PSFP)

PSFP は受信したトラフィックを一定の基準に従って破棄する機能です。timeTransmitter で有効にすることで、timeReceiver から送信されたトラフィックにフィルタをかけることができます。

【注】 RZ/T2M-RSK ではサポートされていません。

2.5 IEEE802.1Qch (CQF)

CQF はタイムセンシティブなトラフィックをネットワーク上のホップ数にのみ依存する確定的なレイテンシで伝送できるようにすることを目的としています。ホップ間のレイテンシとスイッチング遅延を取り除くためには、各中間デバイスが、前の期間内に受信したフレームを送信しながら、現在の期間内に受信したフレームをバッファに入れる必要があります。これにより、ネットワークとスイッチの遅延でフレームを期間内にすべて送受信できる場合には、ホップ間のレイテンシが 1 サイクルと等しくなります。

2.6 IEEE802.1AB (LLDP)

LLDP はネットワーク上で自機の情報を通知する機能です。

3. 開発環境

3.1 開発時の環境

本サンプルアプリケーションにおける開発環境は以下の通りです。

表 3-1 開発環境

Item	Description		
Board	RZ/T2M	Renesas Started Kit+ for RZ/T2M (*1) (R9A07G075M24GBG)	
	RZ/N2L	Renesas Started Kit+ for RZ/N2L (*3) (R9A07G084M04GBG)	
	RZ/T2N	RZ/T2N Evaluation Board (*2) (R9A07G076M48GBG)	
IDE	IAR Systems Embedded Workbench for ARM (*4) (Version: 9.60.3)		
	Renesas Electronics e² studio (Version: 2026-07)	ToolChain	GNU ARM Embedded (Version: 13.3.1.arm-13-24)
FSP	RZ Flexible Software Package (FSP) (v4.2.0)		
	FSP Smart Configurator (Version: 2026-07)		

【注】 *1: 以降 RZ/T2M-RSK と記載

【注】 *2: 以降 RZ/N2L-RSK と記載

【注】 *3: 以降 RZ/T2N-Evb と記載

【注】 *4: 以降 EWARM と記載

3.2 ハードウェア要件

サンプルアプリケーションの実行には以下の物が必要です。

表 3-2 必要物の一覧

Item	Description	Note
下記のいずれか		
● RZ/T2M-RSK	評価ボード	(*1)
● RZ/N2L-RSK		
● RZ/T2N-Evb		
I-jet	デバッグ	(*1)(*2)
USB ケーブル (USB A to Micro-B)	デバッグ接続用	(*1)
USB ケーブル (USB A to Mini-B)	ログメッセージ取得用 + 機能選択のための通信用	(*1)
Ethernet ケーブル	デバイス同士の接続用	
(オプション) 4 チャンネル以上のオシロスコープ	信号波形確認用	

【注】 *1: timeTransmitter 用と timeReceiver 用で同じデバイスを計 2 つ用意してください。(*3)

【注】 *2: EWARM 使用時のみ。

【注】 *3: "4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"のような 3 台構成で稼働させる場合、3 つ必要となります。

3.2.1 ボードのハードウェア設定

以下に各ボードのハードウェア設定事項を示します。

表 3-3 RZ/T2M-RSK ハードウェア設定

Reference	Setting	Description
CN8	Shorted Pin 2-3	QSPI Serial Flash(IC21)を使用
CN17	Shorted Pin 2-3	VCC1833_2 を 1.8V 電源ラインに接続 (Ethernet ポート 2 使用時)
J9	(Open/Short)	外部デバッグ使用の有無 (3.2.1.2 参照。)
SW4-1	(ON/OFF)	ブートモードの選択 (3.2.1.1 参照。)
SW4-2		
SW4-3		
SW4-4	ON	ハッシュによる JTAG 認証無効
SW4-5	OFF	ATCM 1 wait
SW5-3	ON	SCI_RTS 信号が有効
SW5-4	OFF	SCI_RXD 信号が有効
SW5-5	ON	
SW5-6	OFF	
SW5-7	OFF	SCK3 信号が有効
SW5-8	OFF	
SW5-9	ON	
SW5-10	OFF	外部バス以外の信号が有効(CAN、エミュレータ、I2C など)
SW6-1	OFF	
SW6-3	ON	
SW6-4	OFF	
SW6-5	OFF	
SW6-6	ON	
SW6-7	OFF	
SW6-8	ON	
SW6-9	OFF	
SW6-10	ON	

表 3-4 RZ/N2L-RSK ハードウェア設定

Reference	Setting	Description
CN8	Shorted Pin 2-3	QSPI Serial Flash(IC21)を使用
CN17	Shorted Pin 2-3	VCC1833_2 を 1.8V 電源ラインに接続 (Ethernet ポート 2 使用時)
J9	(Open/Short)	外部デバッガ使用の有無 (3.2.1.2 参照。)
SW4-1	(ON/OFF)	ブートモードの選択 (3.2.1.1 参照。)
SW4-2		
SW4-3		
SW4-4	ON	ハッシュによる JTAG 認証無効
SW4-6	OFF	トレース信号以外の信号が有効 (Motor, RS485, etc.)
SW4-7	ON	外部バス以外の信号が有効 (CAN, エミュレータ, I2C など)
SW4-8	OFF	SW3 が有効
SW8-1	OFF	LED_GREEN 信号が有効
SW8-2	ON	
SW8-3	OFF	
SW8-4	ON	LED5 が有効
SW8-5	OFF	
SW11-1	ON	LED_RED2 信号が有効
SW11-2	OFF	
SW11-3	OFF	

表 3-5 RZ/T2N-Evb ハードウェア設定

Referenc e	Setting	Description
SW8-1	ON	ブートモード設定 : "xSPI0 boot mode (x1 boot serial flash)"固定。
SW8-2	ON	(同上)
SW8-3	ON	(同上)
SW8-4	OFF	(未使用、出荷時 = OFF)
SW10-1	OFF	(未使用、出荷時 = OFF)
SW10-2	ON	VCC1833_1 = 1.8V (J15 : 1-2 short)
SW10-3	ON	VCC1833_2 = 1.8V (J18 : 1-2 short)
SW10-4	OFF	(未使用、出荷時 = OFF)
SW10-5	OFF	(未使用、出荷時 = OFF)
SW10-6	ON	VCC1833_5 = 1.8V (J39 : 1-2 short)
SW10-7	OFF	Cortex-R52 CPU0 ATCM wait cycle = 1 wait
SW10-8	OFF	Cortex-R52 CPU1 ATCM wait cycle = 1 wait
SW10-9	ON	JTAG mode = Normal mode
SW10-10	ON	2nd Boot CPU = Cortex-R52 CPU0
SW7-1	ON	P03_6,7、P04_0~3、P15_0 を microSD カード制御信号として使用
SW7-2	ON	P23_1,2,4~7、P24_0~4、P25_0 を Ethernet Port2 制御信号として使用
SW7-3	ON	P00_0~7、P01_0~5、P16_1、P16_7 を Ethernet Port3 制御信号として使用
SW7-4	ON	P01_6,7、P02_0~7、P03_0,1,4,5、P15_1 を Ethernet Port4 制御信号として使用
SW7-5	ON	P15_7、P16_2~5 を QSPI Flash および拡張 SPI 制御信号として使用
SW7-6	ON	P17_5、P18_3~6 を OctaFlash、HyperRAM 制御信号として使用
SW7-7	OFF	P04_4~7、P05_0~6 を HSPI、インバータボード INV3、PMOD2 制御信号として使用
SW7-8	OFF	(同上)
J29	(Open/Short)	外部デバッガ使用の有無 (3.2.1.2 参照。)

【注】 未記載の項目については出荷時のデフォルト設定としてください。
 詳細は"RZ/T2N Evaluation Board ユーザーズマニュアル"を参照してください。

実行環境によっては、一部のハードウェア設定を変更する必要があります。

以下に変更する必要があるハードウェア設定を示します。

- ブートモードの設定
- 外部エミュレータの有効/無効化

3.2.1.1 動作モードを設定する

本サンプルアプリケーションの実行方法は、RAM 実行と xSPI ブート(フラッシュ書き込み実行)から選択可能です。

【注】：RZ/T2N-Evb の場合、ハードウェア変更は「xSPI0 ブートモード(x1 ブートシリアルフラッシュ)」固定で変更不要です。

ただし、「動作モード切り替え時におけるフラッシュメモリの削除」は必須となります。

詳細は「3.2.1.1.1 フラッシュメモリの削除」を参照してください。

- RAM 実行の場合

- 本サンプルアプリケーションをデバッガから RAM に展開して実行する場合は、動作モードを「16 ビットバスブートモード(NOR フラッシュ)」または「xSPI0 ブートモード(x1 ブートシリアルフラッシュ)」に設定してください。

【注】 xSPI ブートから RAM 実行へ切り替える場合、事前に一度フラッシュメモ리를削除しておく必要があります。

詳細は「3.2.1.1.1 フラッシュメモリの削除」を参照してください。

- xSPI ブートの場合

- 本サンプルアプリケーションをフラッシュに書き込んで実行する場合は、動作モードを「xSPI0 ブートモード(x1 ブートシリアルフラッシュ)」に設定してください。

以下にそれぞれの動作モードで起動するために必要な設定を示します：

表 3-6 ハードウェア設定(動作モードの変更)

Reference	16ビットバスブートモード (NORフラッシュ)	xSPI0ブートモード (x1ブートシリアルフラッシュ)
SW4-1	ON	ON
SW4-2	OFF	ON
SW4-3	ON	ON

【注】 RZ/T2M-RSK, RZ/N2L-RSK 共通。

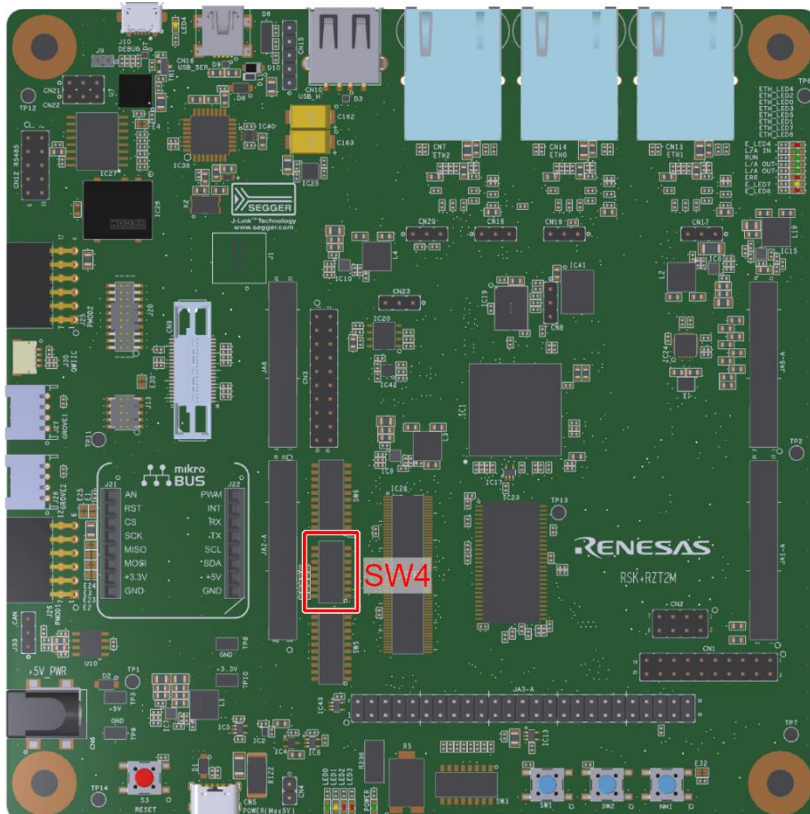


図 3-1 RZ/T2M-RSK ボード図(SW4)

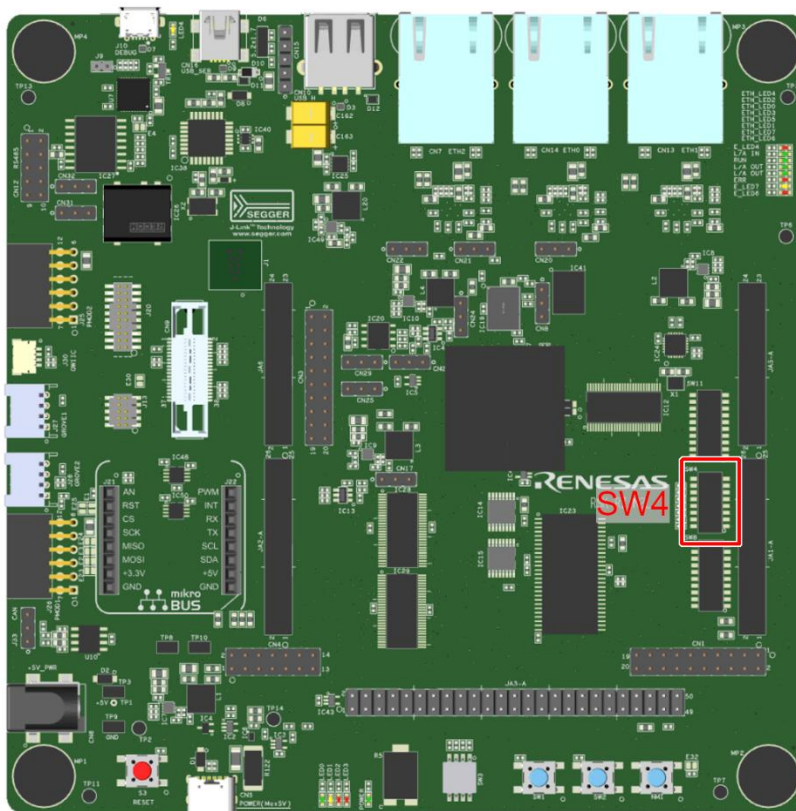


図 3-2 RZ/N2L-RSK ボード図(SW4)

3.2.1.1.1 フラッシュメモリの削除

【注】 この操作は xSPI ブートから RAM 実行に切り替える際に一度だけ実行しておく必要があります。
(切り替え後は繰り返し実行する必要はありません。)

参照 : "RZ/T2, RZ/N2 Getting Started with Flexible Software Package" : "9.1 How to Erase Flash Memory"

3.2.1.1.1.1 EWARM 使用時

【注】 "非対称型マルチコア"が有効となっているとメモリ削除を行うことができません。
以下の手順に従い無効化してください :

1. ツールバーの"プロジェクト"より、"オプション"を選択してください。
2. "デバッガ"の"マルチコア"タブより、"非対称型マルチコア"を"無効"に設定してください。

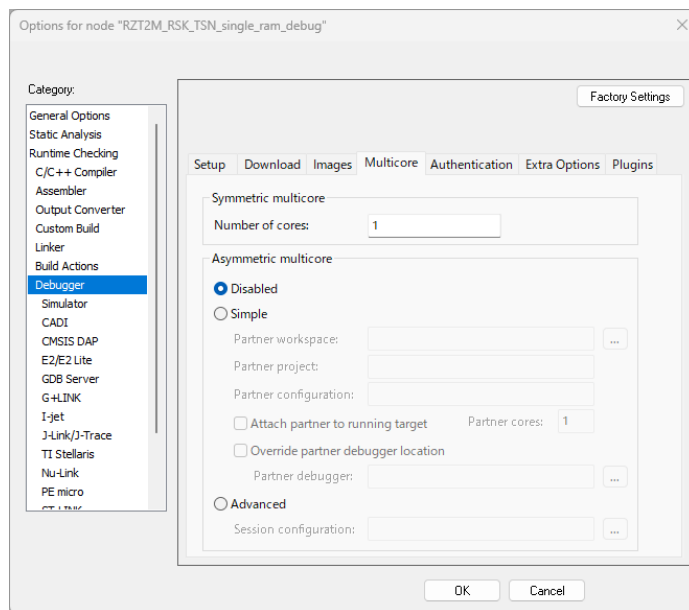


図 3-3 EWARM - 非対称型マルチコアの無効化

以下の手順でフラッシュメモリの削除を行います：

1. EWARM にてプロジェクトのワークスペースを開いてください。
(参照：4.1.1.1)
2. ツールバーの"プロジェクト"より、"ダウンロード" -> "メモリ削除"を選択してください。

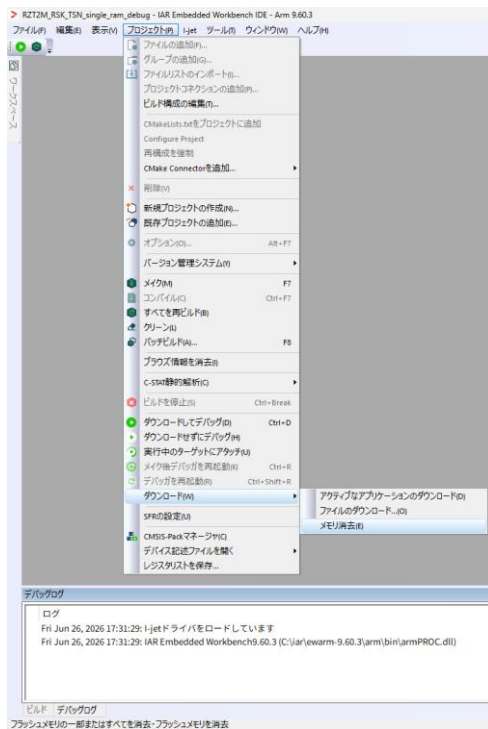


図 3-4 EWARM - メモリ削除の起動

3. 現在の動作モード(参照：3.2.1.1)に応じたメモリ領域を指定し、削除を実行してください。

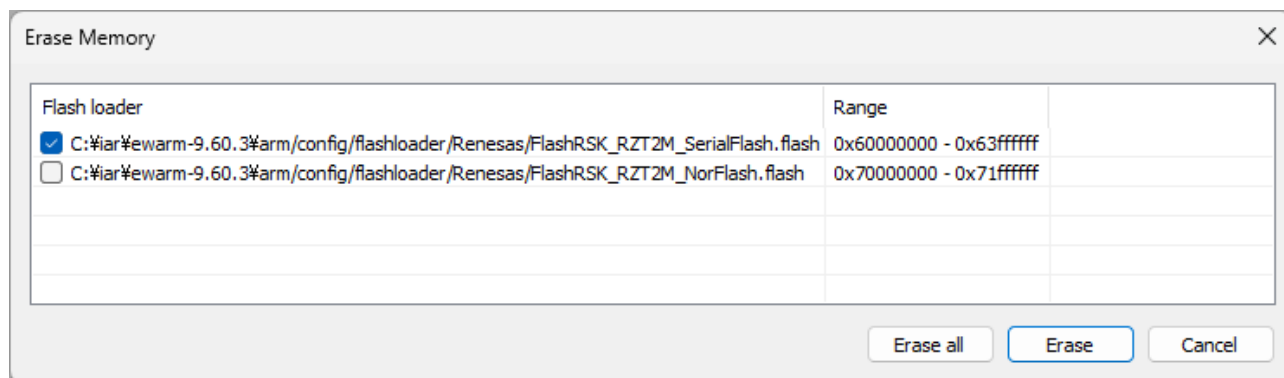


図 3-5 EWARM メモリ削除

- 動作モードとメモリ領域の対応は以下の通りです：

表 3-7 削除対象のメモリ領域 (RZ/T2M, RZ/N2L 使用時)

動作モード	削除対象のメモリ領域		
	アドレス空間	開始アドレス	終了アドレス
xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
16-bit bus	CS0	0x70000000	0x71FFFFFF

【注】 削除対象のメモリアドレスは RZ/T2M-RSK, RZ/N2L-RSK で同一です。

表 3-8 削除対象のメモリ領域 (RZ/T2N 使用時)

動作モード	削除対象のメモリ領域		
	アドレス空間	開始アドレス	終了アドレス
xSPI0 x1	xSPI0 CS0	0x40000000	0x43FFFFFF

4. 以下のダイアログが表示された後、エラーが発生しなければメモリ削除が完了します。

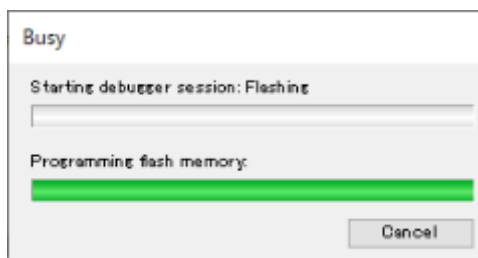


図 3-6 削除中の画面

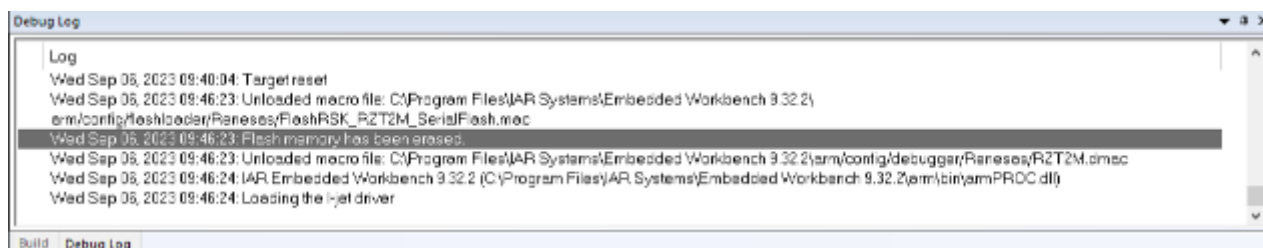


図 3-7 メモリ削除完了メッセージの例

3.2.1.1.1.2 e² studio 使用時

以下の手順でフラッシュメモリの削除を行います：

1. ボードと PC を接続した状態で、J-Link Commander を起動してください。
(参照：3.2.2)

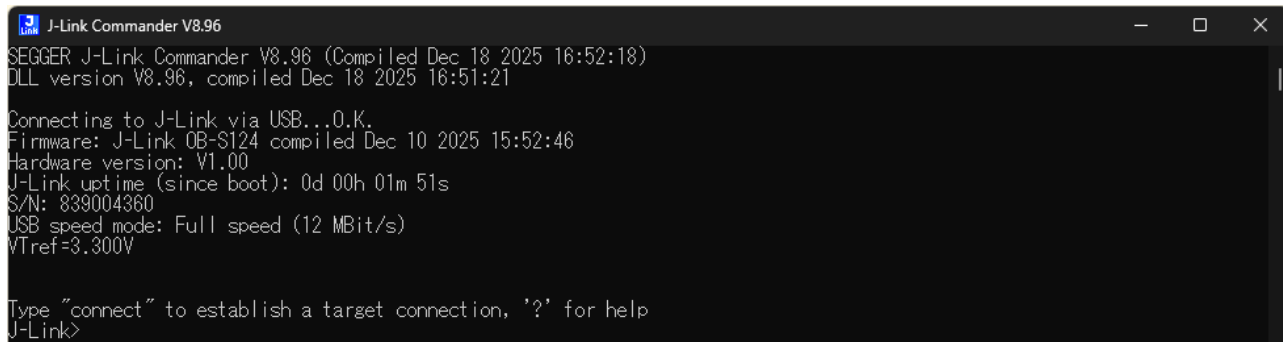


図 3-8 J-Link Commander - 起動時

2. J-Link Commander にて、`connect` と入力してください。
3. `Device>` には、以下に従いデバイスタイプを入力してください：

表 3-9 デバイスタイプ一覧

Board	Device type name
RZ/T2M-RSK	R9A07G075M24_CPU0
RZ/N2L-RSK	R9A07G084M04
RZ/T2N-Evb	R9A07G076M48_R52_0

4. `TIF>` には、`S` を入力してください。
5. `Speed>` には何も入力せずエンターキーを押してください。
(デフォルト設定が使用されます。)

6. その後、"Cortex-R52 identified"のメッセージが表示されることを確認してください。

```

J-Link>connect
Please specify device / core. <Default>: R9A07G084M04
Type '?' for selection dialog
Device>R9A07G075M24_CPU0
Please specify target interface:
J) JTAG (Default)
S) SWD
T) cJTAG
TIF>S
Specify target interface speed [kHz]. <Default>: 4000 kHz
Speed>
Device "R9A07G075M24_CPU0" selected.

Connecting to target via SWD
ConfigTargetSettings() start
ConfigTargetSettings() end - Took 178us
InitTarget() start
Authenticated device detected. Skipping authentication process.
  OCDREG STATUS: 0x00000001
InitTarget() end - Took 205ms
Found SW-DP with ID 0x6BA02477
DPIDR: 0x6BA02477
CoreSight SoC-400 or earlier
AP map detection skipped. Manually configured AP map found.
AP[0]: APB-AP (IDR: Not set, ADDR: 0x00000000)
AP[1]: APB-AP (IDR: Not set, ADDR: 0x00000000)
AP[2]: AXI-AP (IDR: Not set, ADDR: 0x00000000)
Using preconfigured AP[1] as APB-AP
AP[1]: APB-AP found
DebugRegs + CTI manually specified. ROM table scan skipped.
Cortex-R52 @ 0x80410000 (configured)
CoreCTI @ 0x80420000 (configured)
Debug architecture: ARMv8.0
8 code breakpoints, 8 data breakpoints
Processor features:
  EL0 support: AArch32
  EL1 support: AArch32
  EL2 support: AArch32
  EL3 support: N/A
  FPU support: Single + Double + Conversion
Add. info (CPU temp. halted)
Current exception level: EL1
Exception level AArch usage:
  EL0: AArch32
  EL1: AArch32
  EL2: AArch32
  EL3: AArch32
Non-secure status: Non-secure
Cache info:
  Inner cache boundary: none
  LoU Uniprocessor: 1
  LoC: 1
  LoU Inner Shareable: 1
I-Cache L1: 16 KB, 64 Sets, 64 Bytes/Line, 4-Way
D-Cache L1: 16 KB, 64 Sets, 64 Bytes/Line, 4-Way
Memory zones:
  Zone: "Default" Description: Default access mode
  Zone: "AP0" Description: MEM-AP (APB-AP)
  Zone: "AP1" Description: MEM-AP (APB-AP)
  Zone: "AP2" Description: MEM-AP (AXI-AP)
Cortex-R52 identified.
J-Link>

```

図 3-9 J-Link Commander - 接続成功

7. `exec EnableEraseAllFlashBanks`と入力してください。
8. 現在の動作モード(参照 : 3.2.1.1)に応じたメモリ領域を指定し、`erase`コマンドを実行してください。
 - 実行例: `erase 0x60000000,0x63FFFFFF`
 - 動作モードとメモリ領域の対応は以下の通りです :

表 3-7 削除対象のメモリ領域 (RZ/T2M, RZ/N2L 使用時) (再掲)

動作モード	削除対象のメモリ領域		
	アドレス空間	開始アドレス	終了アドレス
xSPI0 x1	xSPI0 CS0	0x60000000	0x63FFFFFF
16-bit bus	CS0	0x70000000	0x71FFFFFF

【注】 削除対象のメモリアドレスは RZ/T2M-RSK, RZ/N2L-RSK で同一です。

表 3-8 削除対象のメモリ領域 (RZ/T2N 使用時) (再掲)

動作モード	削除対象のメモリ領域		
	アドレス空間	開始アドレス	終了アドレス
xSPI0 x1	xSPI0 CS0	0x40000000	0x43FFFFFF

9. その後、"Erasing done."メッセージが表示されることを確認してください。

```

Cortex-R52 identified.
J-Link>exec EnableEraseAllFlashBanks
J-Link>erase 0x60000000 0x63FFFFFF
erase : Performing implicit reset & halt of MCU.
ResetTarget() start
Authenticated device detected. Skipping authentication process.
OCDREG_STATUS: 0x00000001
Disabled core power domain detected.
Enabling debug mode...
ResetTarget() end - Took 537ms
Device specific reset executed.
Erasing selected range...
J-Link: Flash download: Total time needed: 238.444s (Prepare: 1.718s, Compare: 0.000s, Erase: 236.464s, Program: 0.000s, Veri
fy: 0.000s, Restore: 0.262s)
J-Link: Flash download:
Flash sectors within Range [0x60000000 - 0x63FFFFFF] deleted.
Erasing done.
J-Link>_

```

図 3-10 J-Link Commander - erase 成功

10. `q`と入力し、J-Link Commander を終了してください。

3.2.1.2 外部エミュレータの有効/無効

使用する統合開発環境によって、外部エミュレータの有効/無効を切り替える必要があります。

- EWARM を使用する場合
 - I-jet デバッガ(外部エミュレータ)を使用します。
外部エミュレータを**有効**にしてください。
- e² studio を使用する場合
 - J-LinkOB(オンボードデバッガ)を使用します。
外部エミュレータを**無効**にしてください。

外部エミュレータの有効/無効化には、以下の操作が必要です。

表 3-10 ハードウェア設定(外部エミュレータの有効/無効化)

使用ボード	EWARM 使用時 (外部エミュレータ : 有効)	e ² studio 使用時 (外部エミュレータ : 無効)
RZ/T2M	J9:Short	J9:Open
RZ/N2L		
RZ/T2N	J29:Short	J29:Open

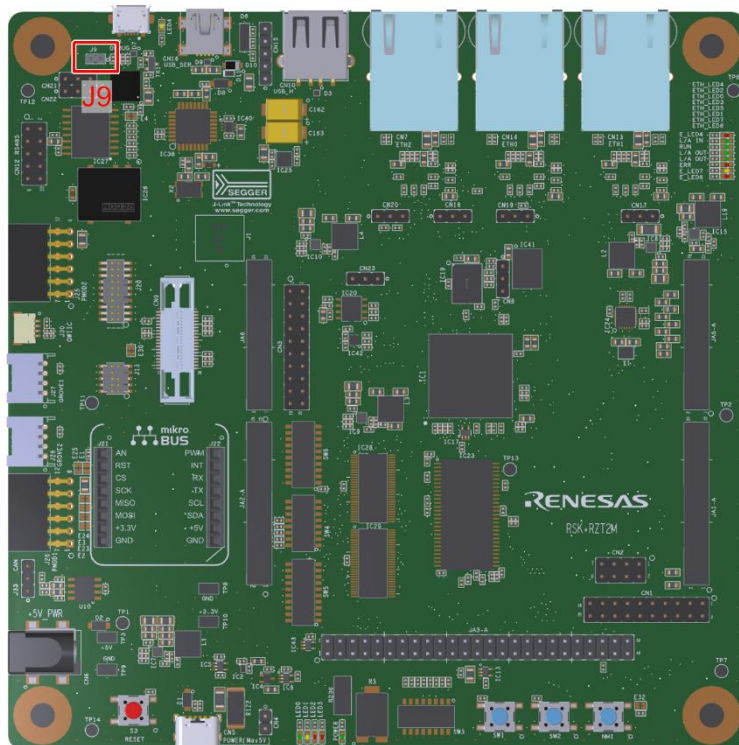


図 3-11 RZ/T2M-RSK ボード図(J9)

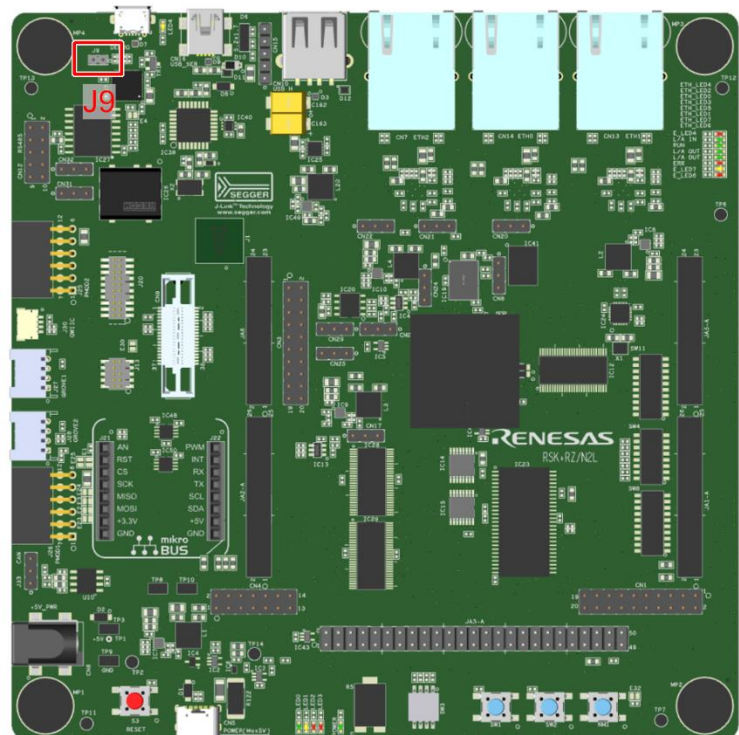


図 3-12 RZ/N2L-RSK ボード図(J9)

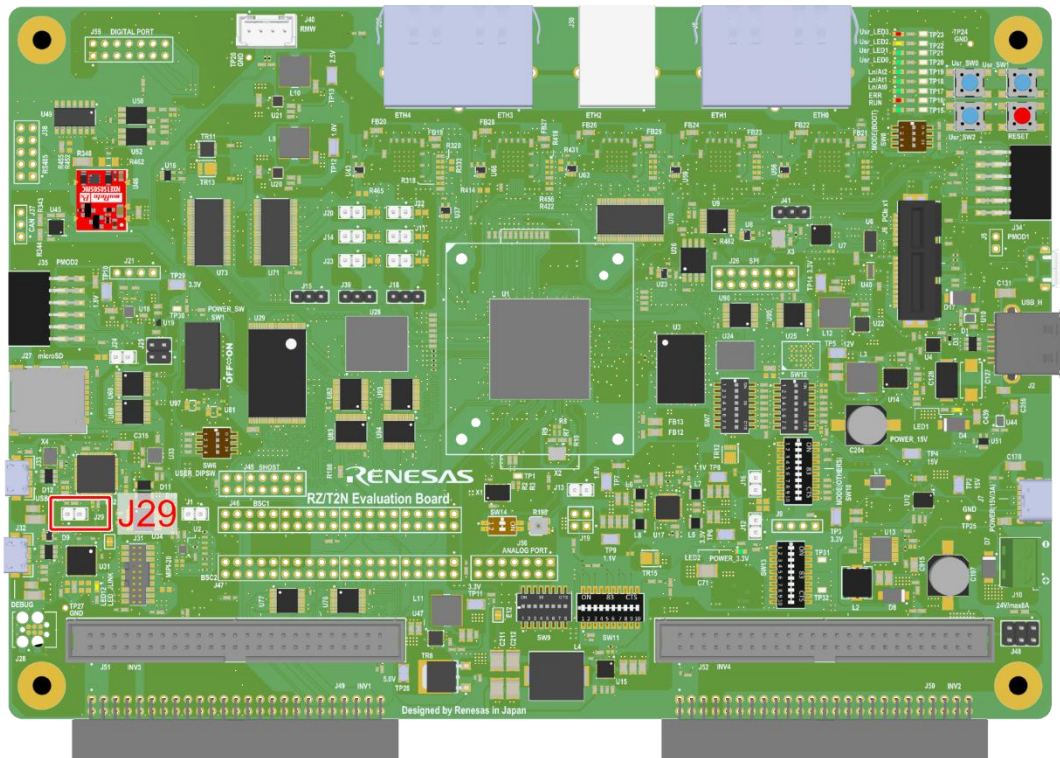


図 3-13 RZ/T2N-Evb ボード図 (J29)

3.2.2 ボードと接続する

3.2.2.1 EWARМ 使用時

ボードとホストとの接続には、I-jet デバッガを使用します。

まず I-jet とホスト PC を接続し、次に I-jet とボードを接続してください。

接続する端子の割り当ておよびボード上の配置を以下に示します。

表 3-11 I-jet 用端子割り当て

ボード	割り当て
RZ/T2M-RSK	J20
RZ/N2L-RSK	
RZ/T2N-Evb	J31

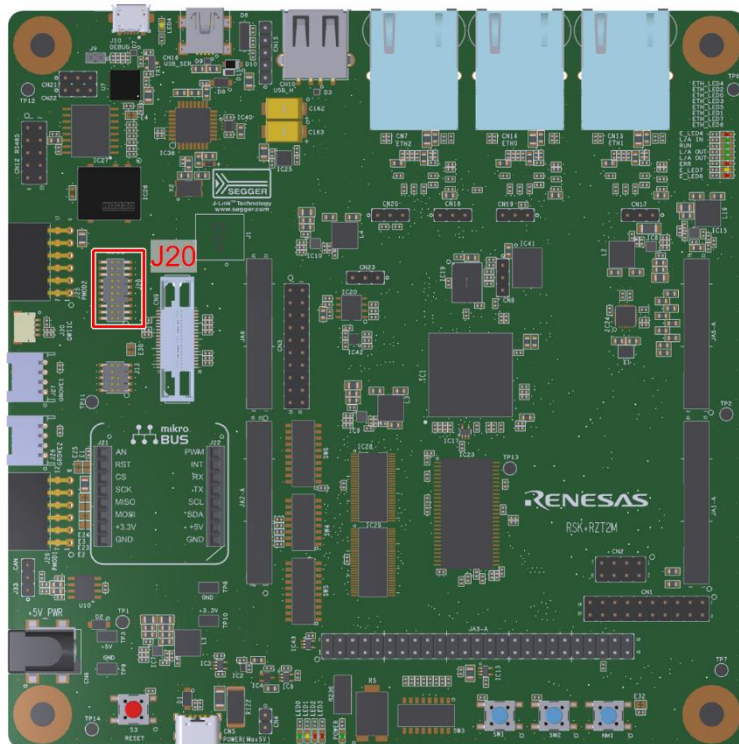


図 3-14 RZ/T2M-RSK ボード図(J20)

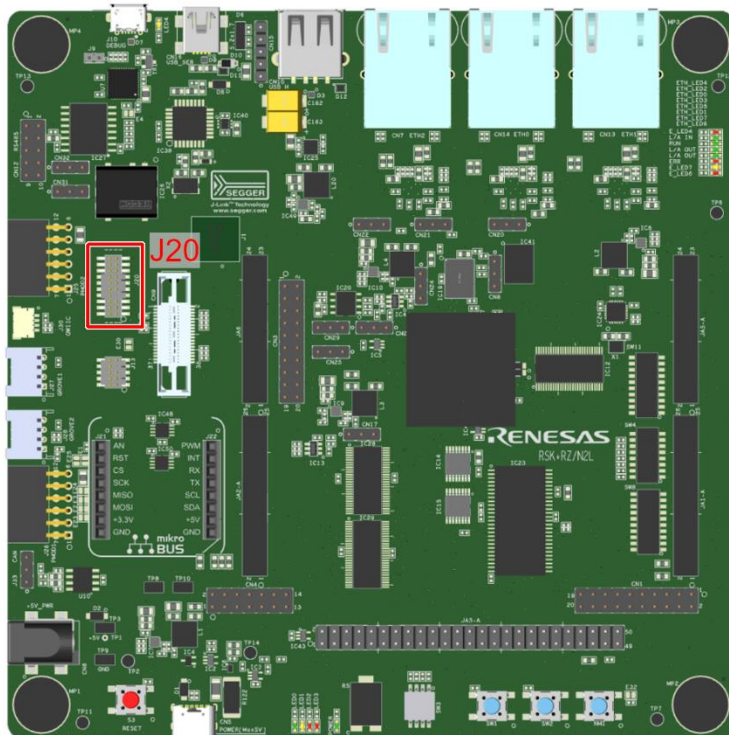


図 3-15 RZ/N2L-RSK ボード図(J20)

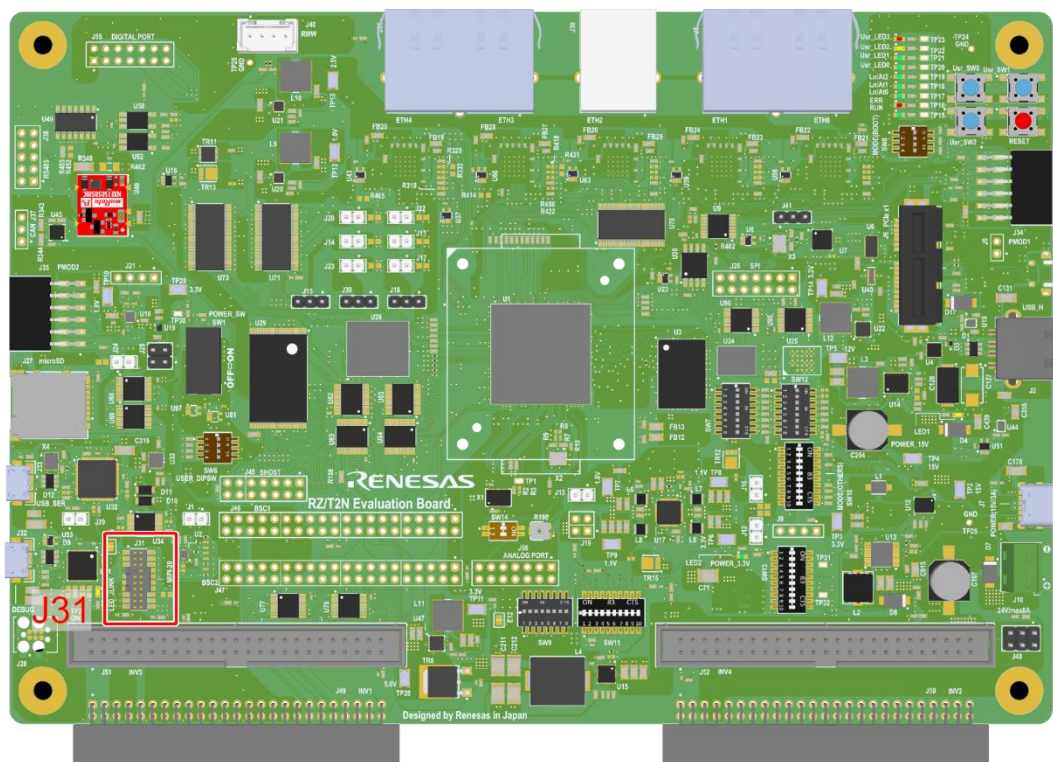


図 3-16 RZ/T2N-Evb ボード図 (J31)

正常に接続されると、I-jet の"USB"と"DBG"のランプが点灯します。

【注】 "TPWR"のランプは消灯している必要があります。

3.2.2.2 e² studio 使用時

ボードとホストとの接続には、オンボードデバugg(J-LinkOB)を使用します。

ボードの"J-Link OB USB Connector"とホスト PC を接続してください。

接続する端子の割り当ておよびボード上の配置を以下に示します。

表 3-12 I-jet 用端子割り当て

ボード	割り当て
RZ/T2M-RSK	J10
RZ/N2L-RSK	
RZ/T2N-Evb	J32

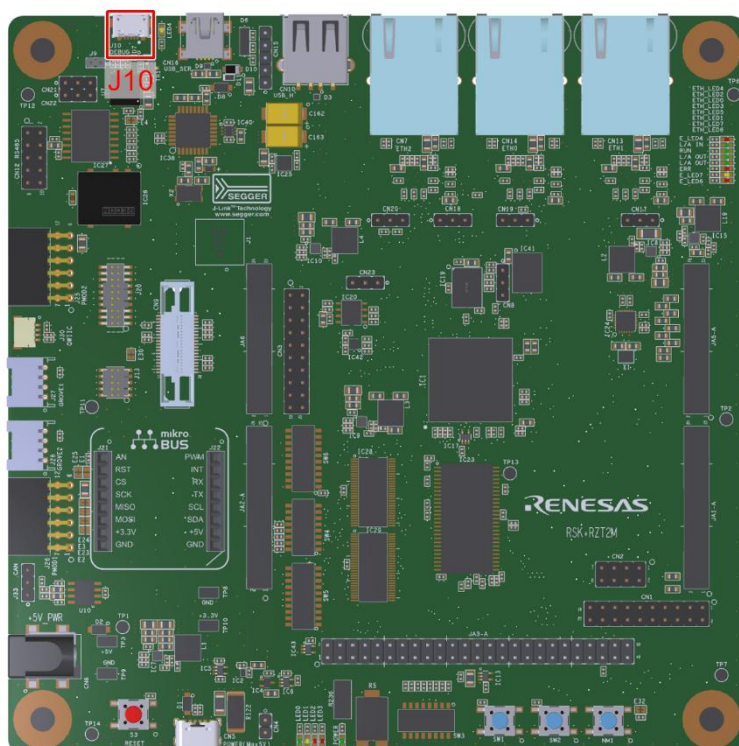


図 3-17 RZ/T2M-RSK ボード図(J10)

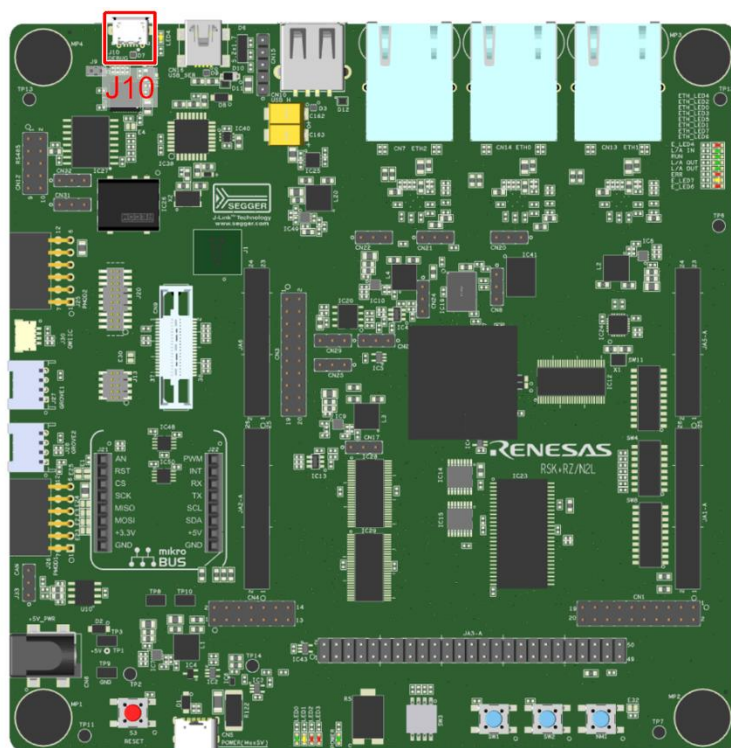


図 3-18 RZ/N2L-RSK ボード図(J10)

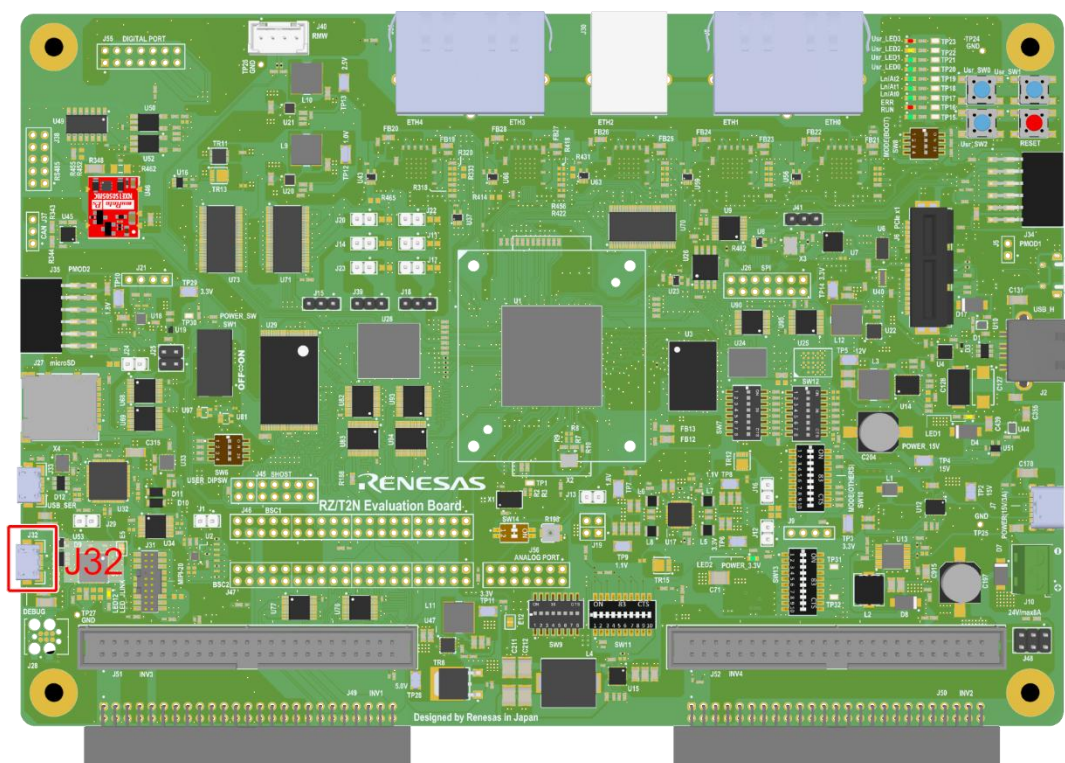


図 3-19 RZ/T2N-Evb ボード図(J32)

【注】 正しく接続できているかは、以下の手順で確認することができます。

1. e² studio 上のツールバーより"Launch Configuration"横の歯車アイコンをクリック
2. 現れたダイアログより"Debugger"タブを選択
3. 現れた画面より"Connection Setting"タブを選択

4. "J-Link" -> "J-Link Serial" の項目にて、接続したボードの J-Link シリアル番号が選択できれば接続は成功です。

3.3 ソフトウェア要件

サンプルアプリケーションの実行に必要なソフトウェアを示します。

3.3.1 サンプルアプリケーションのダウンロードに必要なソフトウェア

サンプルアプリケーションのビルド及び各ボードへのダウンロードには以下のソフトウェアが必要です。

- 統合開発環境

EWARM と **e² studio** から選択できます。

A) **EWARM** (*1)(*2)

- 【注】 *1: 本ソフトウェアの使用にはライセンス認証が必要です。

➤ 【注】 *2: 各ボードを使用するために、下記のソフトウェアの導入が必要です。

RZ FSP SC(スマートコンフィグレータ) and RZ FSP インストーラ(EWARM 向け)

"setup_rzfsp_v4_2_0_rzsc_v2026-07.exe"

【注】 これらのソフトは上記の統合開発環境 から起動します。

起動方法については” 3.3.1.1 FSP Smart Configurator を 統合開発環境 から起動する方法” を参照してください。

B) **e² studio**

e² studio and RZ FSP インストーラ

"setup_rzfsp_v4_2_0_e2s_v2026-07.exe"

【注】 "e² studio and RZ FSP インストーラ"および"RZ FSP SC(スマートコンフィグレータ) and FSP インストーラ(EWARM 向け)"は、以下のリンクより入手可能です：

<https://github.com/renesas/rz-fsp/releases>

【注】 各ソフトウェアバージョン等の詳細情報については、"3.1 開発時の環境"を参照してください。

3.3.1.1 統合開発環境に関するセットアップ

3.3.1.1.1 FSP Smart Configurator を 統合開発環境 から起動する方法

3.3.1.1.1.1 EWARМ の場合

1. ツールバーより"ツール" -> "ツールの設定"を選択して下さい。

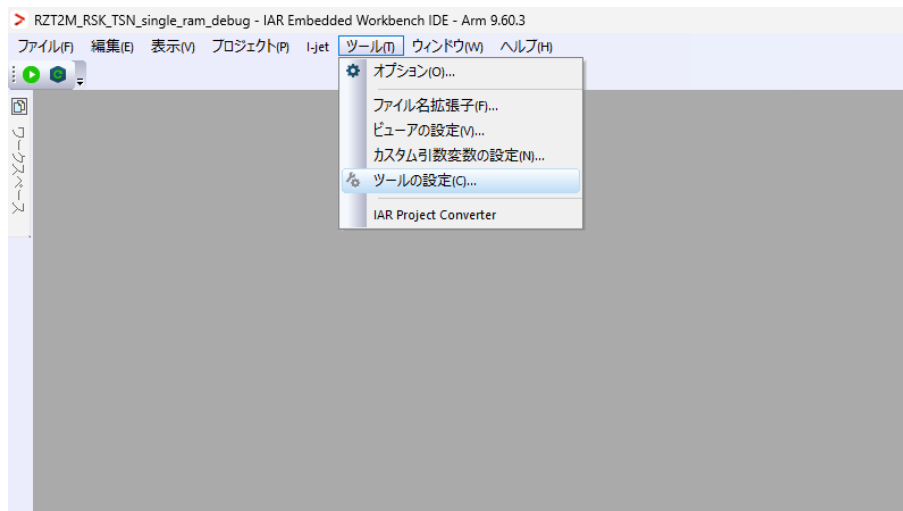


図 3-20 EWARМ - ツールバー(設定前)

2. "ツールの設定"より"新規作成"を選択して下さい。
3. 以下の内容を設定して下さい。

表 3-13 ツールの設定 における指定内容

項目	内容
メニューテキスト	FSP Smart Configurator
コマンド	<FSP Smart Configurator(rasc.exe)のパス>
引数	--compiler IAR configuration.xml
初期ディレクトリ	\$PROJ_DIR\$

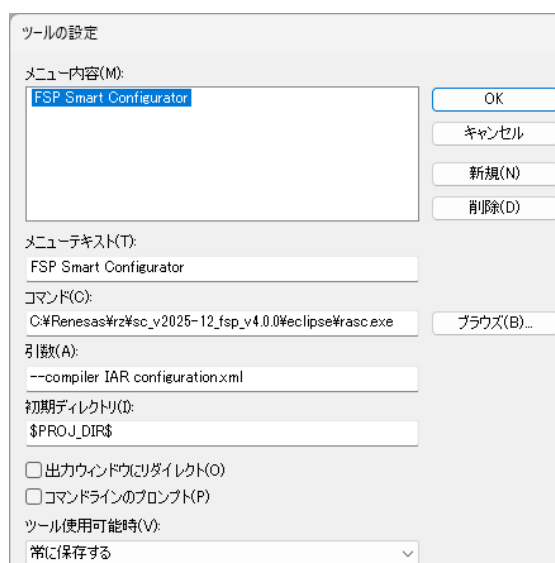


図 3-21 EWARМ - "ツールの設定"

4. 設定が完了したら、ツールバーより"ツール" -> "FSP Smart Configurator"を選択してください。

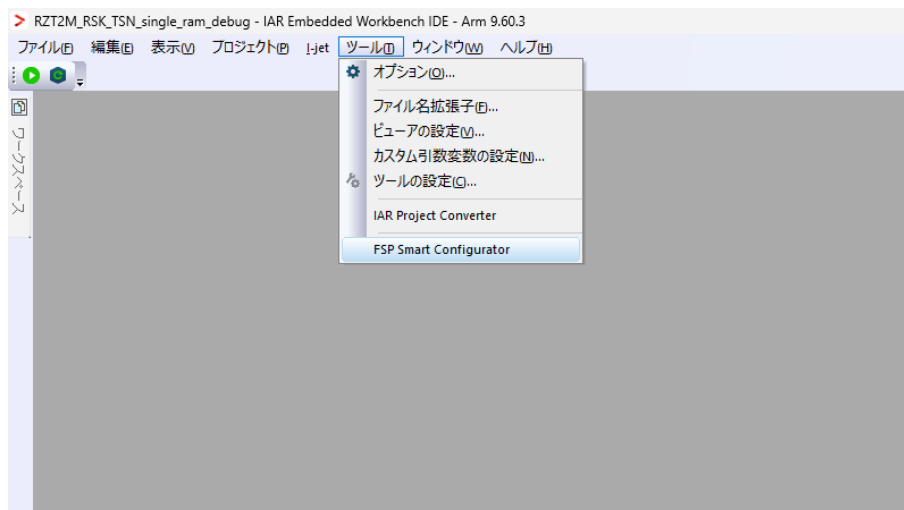


図 3-22 EWARM - ツールバー(設定後)

FSP Smart Configurator が起動します。

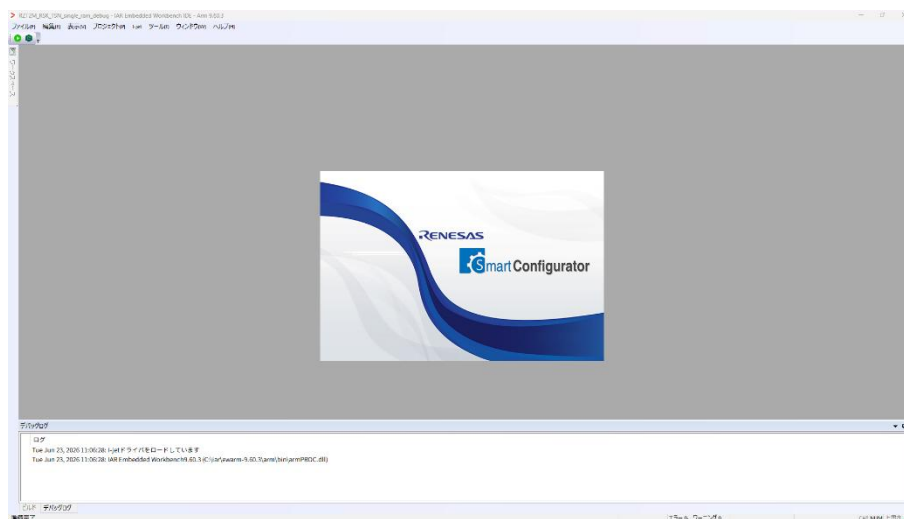


図 3-23 EWARM - FSP Smart Configurator 起動時の様子

3.3.1.1.1.2 e² studio の場合

1. e² studio 上のファイルエクスプローラより、"configuration.xml"を開いてください。
FSP Smart Configurator が起動します。

3.3.1.1.2 複数のデバイス接続時にダウンロード先を選択できるようにする

3.3.1.1.2.1 EWARМ の場合

1. ツールバーより、"プロジェクト" -> "オプション"を選択してください。
【注】または IDE のファイルエクスプローラよりプロジェクトを右クリックし、"オプション"を選択してください。
2. "カテゴリ:"ボックス内から"I-jet"を選択してください。
3. "設定"タブ内より、エミュレータの項にある"プローブ選択プロンプトを常に表示"のチェックボックスにチェックを付けてください。

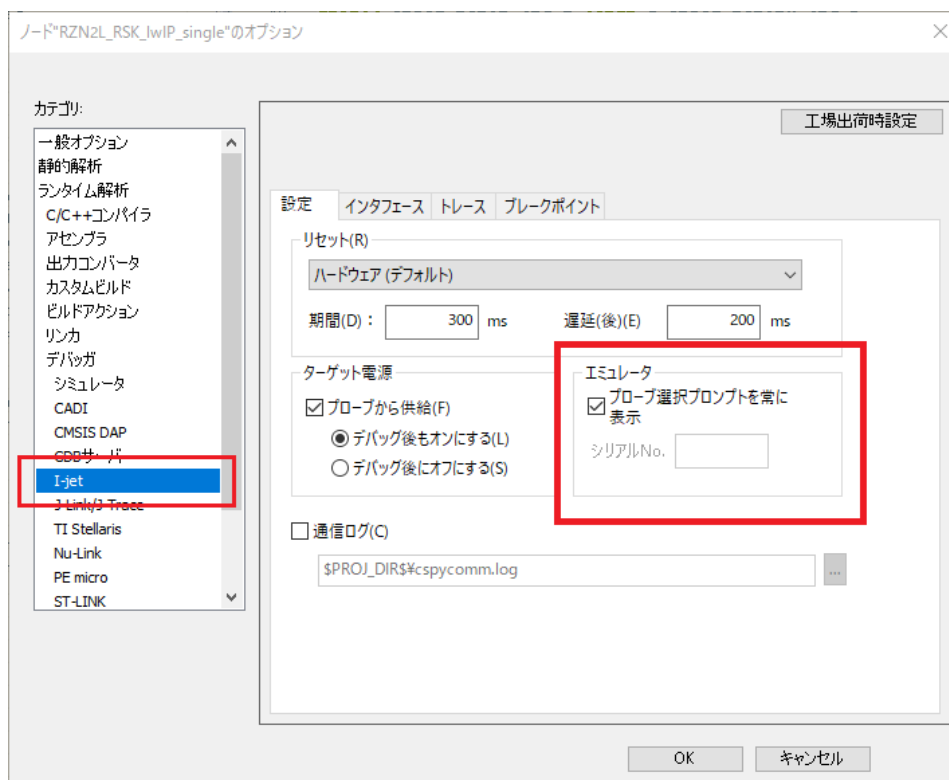


図 3-24 設定画面

以降、アプリケーションダウンロード時に使用する I-jet のシリアルを選択するダイアログが表示されるようになります。

ダイアログ内にて、アプリケーションをダウンロードさせたいボードが接続されている I-jet を指定してください。

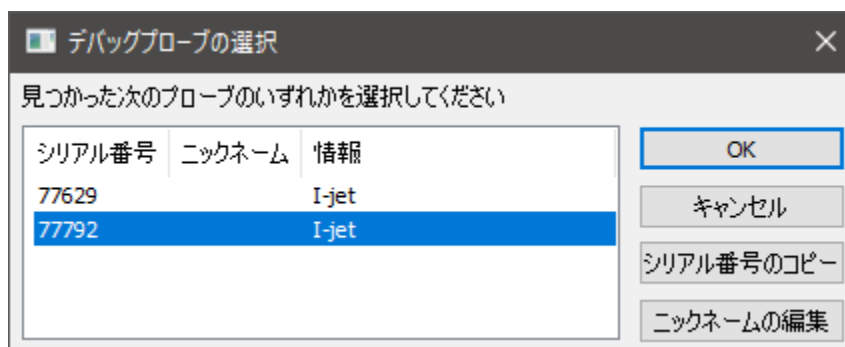


図 3-25 I-jet 選択画面

3.3.1.1.2.2 e² studio の場合

1. e² studio 上のツールバーより"Launch Configuration"横の歯車アイコンをクリック
2. 現れたダイアログより"Debugger"タブを選択
3. 現れた画面より"Connection Setting"タブを選択
4. "J-Link" -> "J-Link Serial" の項目にて、接続したいボードの J-Link シリアル番号を選択してください

3.3.2 サードパーティソフトウェア

3.3.2.1 シリアルターミナルツール

サンプルアプリケーションは UART 経由でログメッセージを出力します。

ログメッセージの確認には、シリアルターミナルツールを使用してください。

2. TeraTerm

<https://teratermproject.github.io/index-en.html>

TeraTerm におけるシリアルポート設定は以下の通りです。

表 3-14 シリアルポート設定

項目	設定内容	
ボーレート	115200	
データ	8bit	(初期設定)
パリティ	none	(初期設定)
ストップビット	1bit	(初期設定)
フロー制御	none	(初期設定)

4. サンプルアプリケーションの実行

本章は、サンプルアプリケーションのビルドから実行までの一連の手順を示します。

4.1 ビルドとダウンロード

4.1.1 プロジェクトを開く

本パッケージ内には、以下の 16 プロジェクトが内包されています。

表 4-1 プロジェクト一覧

#	対象ボード	ビルド環境	動作モード	プロジェクトディレクトリ	備考
1	RZ/T2M-RSK	EWARM	RAM 実行	/project/rzt2m_rsk/tsn_single/ewarm_ram_debug/	
2			xSPI ブート	/project/rzt2m_rsk/tsn_single/ewarm_xspi_boot/	
3		e ² studio	RAM 実行	/project/rzt2m_rsk/tsn_single/e2studio_ram_debug/	
4			xSPI ブート	/project/rzt2m_rsk/tsn_single/e2studio_xspi_boot/	
5	RZ/N2L-RSK	EWARM	RAM 実行	/project/rzn2l_rsk/tsn_single/ewarm_ram_debug/	
6			xSPI ブート	/project/rzn2l_rsk/tsn_single/ewarm_xspi_boot/	
7		e ² studio	RAM 実行	/project/rzn2l_rsk/tsn_single/e2studio_ram_debug/	
8			xSPI ブート	/project/rzn2l_rsk/tsn_single/e2studio_xspi_boot/	
9	RZ/T2N-Evb	EWARM	RAM 実行	/project/rzt2n_rsk/tsn_multi_cr52_cr52/ewarm_ram_debug_cpu0/	CR52_0 用
10				/project/rzt2n_rsk/tsn_multi_cr52_cr52/ewarm_ram_debug_cpu1/	CR52_1 用
11			xSPI ブート	/project/rzt2n_rsk/tsn_multi_cr52_cr52/ewarm_xspi_boot_cpu0/	CR52_0 用
12				/project/rzt2n_rsk/tsn_multi_cr52_cr52/ewarm_xspi_boot_cpu1/	CR52_1 用
13		e ² studio	RAM 実行	/project/rzt2n_rsk/tsn_multi_cr52_cr52/e2studio_ram_debug_cpu0/	CR52_0 用
14				/project/rzt2n_rsk/tsn_multi_cr52_cr52/e2studio_ram_debug_cpu1/	CR52_1 用
15			xSPI ブート	/project/rzt2n_rsk/tsn_multi_cr52_cr52/e2studio_ram_xspi_boot_cpu0/	CR52_0 用
16				/project/rzt2n_rsk/tsn_multi_cr52_cr52/e2studio_ram_xspi_boot_cpu1/	CR52_1 用

以下の手順に従い、それぞれのビルド環境に適合したプロジェクトを開いてください。

4.1.1.1 EWARM を使用する場合

プロジェクトの含まれている zip ファイル("RZT2M_RZN2L_TSN_RSK.zip")を解凍してください。

解凍によって得られる下記のファイルを EWARM で開いてください。

1. RZ/T2M-RSK の場合

A) RAM 実行の場合

```
/project/rzt2m_rsk/tsn_single/ewarm_ram_debug/
RZT2M_RSK_TSN_single_ram_debug.eww
```

B) xSPI ブートの場合

```
/project/rzt2m_rsk/tsn_single/ewarm_xspi_boot/
RZT2M_RSK_TSN_single_xspi_boot.eww
```

2. RZ/N2L-RSK の場合

A) RAM 実行の場合

```
/project/rzn2l_rsk/tsn_single/ewarm_ram_debug/
RZN2L_RSK_TSN_single_ram_debug.eww
```

B) xSPI ブートの場合

```
/project/rzn2l_rsk/tsn_single/ewarm_xspi_boot/  
RZN2L_RSK_TSN_single_xspi_boot.eww
```

3. RZ/T2N-Evb の場合

【注】 ビルド・実行はCPU0 側のプロジェクト(プライマリプロジェクト)が先である必要があります。

詳細は"4.1.4 ビルドとボードへのダウンロード (マルチプロセス版)"を参照してください。

A) RAM 実行の場合

◇ CPU0 側 (プライマリプロジェクト)

```
/project/rzt2n_evb/tsn_multi_cr52_cr52/ewarm_ram_debug_cpu0/  
RZT2N_EVB_TSN_multi_ram_debug_cpu0.eww
```

◇ CPU1 側 (セカンダリプロジェクト)

```
/project/rzt2n_evb/tsn_multi_cr52_cr52/ewarm_ram_debug_cpu1/  
RZT2N_EVB_TSN_multi_ram_debug_cpu1.eww
```

B) xSPI ブートの場合

◇ CPU0 側 (プライマリプロジェクト)

```
/project/rzt2n_evb/tsn_multi_cr52_cr52/ewarm_xspi_boot_cpu0/  
RZT2N_EVB_TSN_multi_xspi_boot_cpu0.eww
```

◇ CPU1 側 (セカンダリプロジェクト)

```
/project/rzt2n_evb/tsn_multi_cr52_cr52/ewarm_xspi_boot_cpu1/  
RZT2N_EVB_TSN_multi_xspi_boot_cpu1.eww
```

4.1.1.2 e² studio を使用する場合

以下の手順にしたがってプロジェクトをインポートしてください。

1. プロジェクトの含まれている zip ファイル("RZT2M_RZN2L_TSN_RSK.zip")を解凍してください。
2. 解凍によって得られるディレクトリ("RZT2M_RZN2L_TSN_RSK")をプロジェクト生成先となるディレクトリに移動させてください。
3. e² studio を起動し、ツールバーの"File"より"Open Projects from fileSystem..."を選択してください。

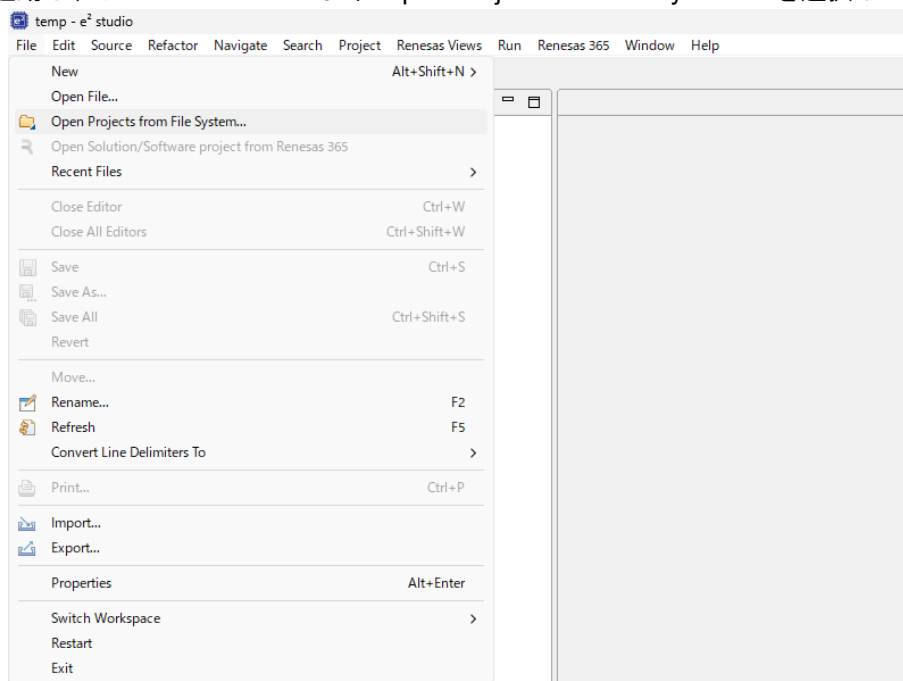


図 4-1 e² studio - ツールバー

4. 現れたウィンドウより"[Directory...]"をクリックし、先ほど移動させたディレクトリ("RZT2M_RZN2L_TSN_RSK")を選択してください。

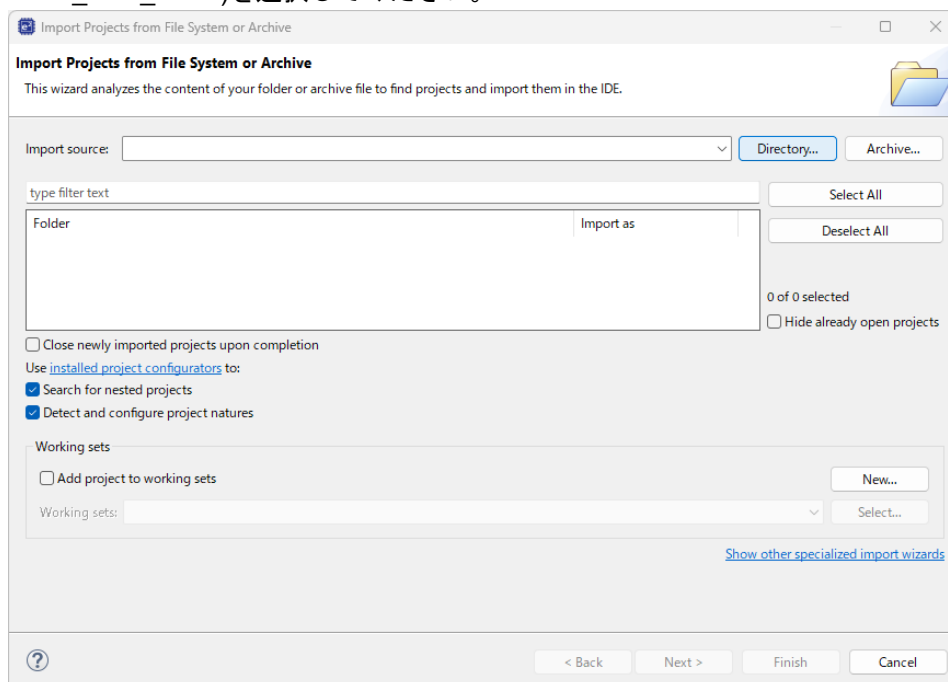


図 4-2 e² studio - インポート画面(ディレクトリ選択前)

5. インポート対象となる e² studio 用プロジェクトの一覧がウィンドウ内に表示されます。
この中からインポートしたいプロジェクトを選択してください(複数選択可)。

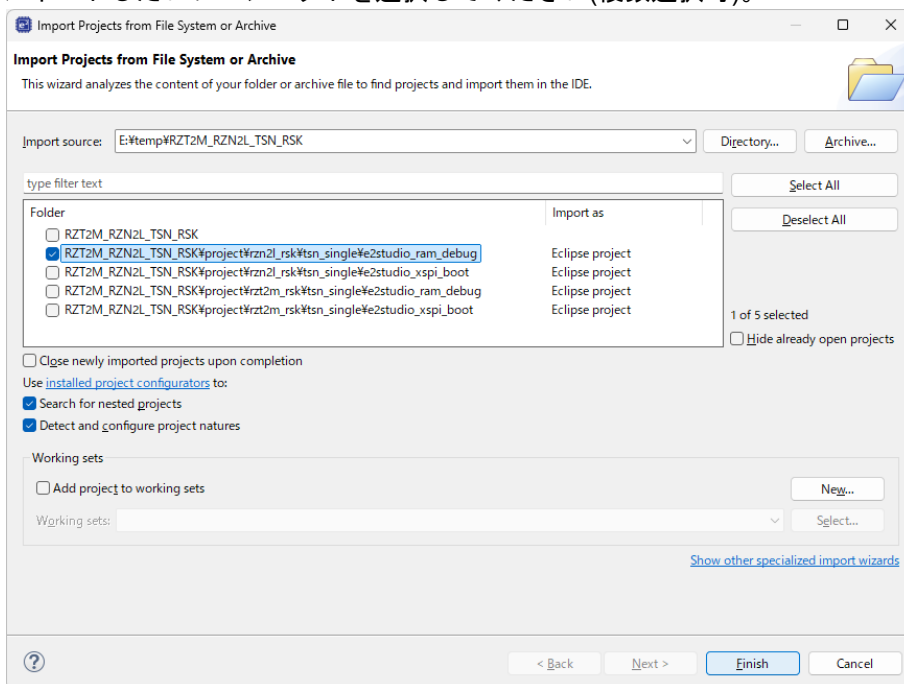


図 4-3 e² studio - インポート画面(プロジェクト選択後)

【注】"RZT2M_RZN2L_TSN_RSK"そのもの(リスト最上)は選択する必要はありません。

6. "[Finish]"をクリックすることでプロジェクトがインポートされます。

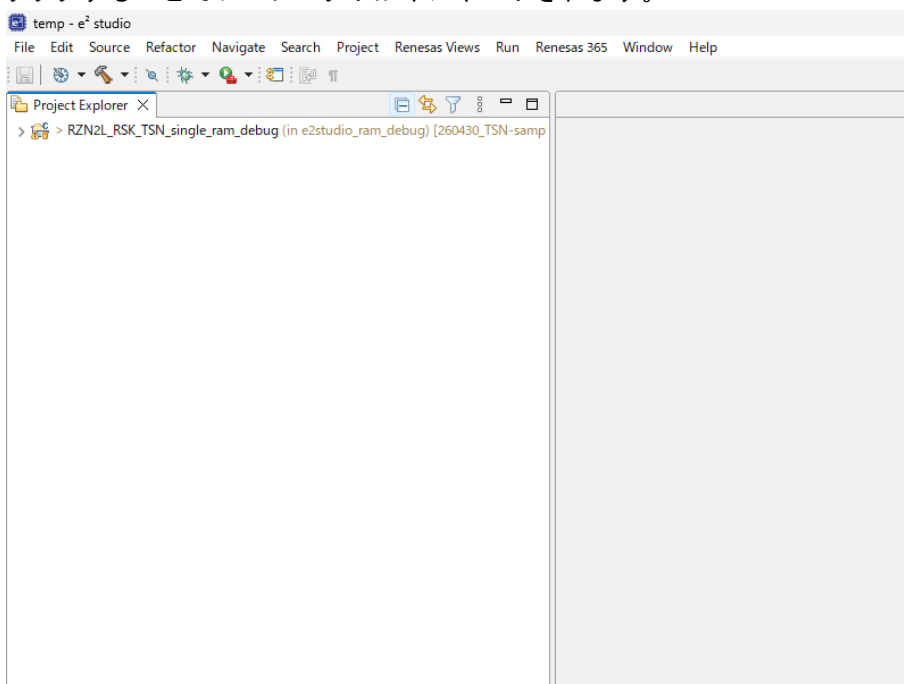


図 4-4 e² studio - プロジェクトインポート後の様子

4.1.2 デバイスの事前設定

ビルド前に、デバイスごとの設定を行う必要があります。

設定内容は以下の通りです。

- デバイスの MAC アドレス
- デバイスの IP アドレス(*1)
- timeTransmitter / timeReceiver 切り替え
- TDMA サイクルのテーブル(*2)
- PROFINET フレーム通信(RT 通信)設定(*2)
- UDP フレーム通信(NRT 通信)設定(*2)
- パルスジェネレータの周期(*2)
- Frame Preemption の設定
- UDP パケット送信元の MAC アドレス(*3)
- PSFP の設定
- CQF の設定(*4)

【注】 *1: "4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"のような 3 台構成を行うときのみこの設定変更が必要です。

【注】 *2: 全てのデバイスで設定を統一する必要があります。

【注】 *3: クローズイベントによるプリエンプトを使用することなく(、かつ TDMA サイクルテーブルを"表 5-7 TDMA スロット設定内容(Frame Preemption)"から変更しない状態で)プリエンプションを発生させる際に必要となる設定です。

【注】 *4: CQF の動作確認にて、"4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"のような 3 台構成を行うときのみ設定変更が必要です。

【注】 *5: マルチプロセス版プロジェクトを使用する場合、これら設定は CPU1 用プロジェクトの方で設定してください。

4.1.2.1 MAC アドレス変更方法

1. FSP Smart Configurator を起動して下さい。

【注】 起動方法については"3.3.1.1.1 FSP Smart Configurator を 統合開発環境 から起動する"を参照してください。

2. FSP Smart Configurator にて、FSP Configuration 関連ビュー下部にある"Stack"タブを開いてください。

3. ビュー左上の"Threads"のリストより、Main Thread 内にある g_ether0 Ethernet Driver on r_ether を選択して下さい。

4. Properties ビューの Setting より、Module g_ether0 Ethernet Driver on r_ether->General まで展開して下さい。

5. General 内にある"MAC address"がデバイスの MAC アドレスとなります。

6. 割り振りたいアドレスに応じて値を書き換えて下さい。

【注】 デバイスごとに異なる値を設定して下さい。

4.1.2.2 IP アドレス変更方法

【注】 3 台構成を行うときのみ設定変更が必要です。

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

app_config.h の マクロ TCPIP_IP_MASTER と TCPIP_IP_SLAVE によって、timeTransmitter と timeReceiver の IP アドレスを指定しています。

```
#define TCPIP_IP_MASTER LWIP_MAKEU32(192,168, 10,170)
#define TCPIP_IP_SLAVE LWIP_MAKEU32(192,168, 10,171)
```

【注】 デバイスが timeTransmitter の場合は TCPIP_IP_MASTER で指定されている IP アドレスが割り当てられます。

【注】 デバイスが timeReceiver の場合は TCPIP_IP_SLAVE で指定されている IP アドレスが割り当てられます。

割り振りたいアドレスに応じて値を書き換えて下さい。

【注】 デバイスごとに異なる値を設定してください。

4.1.2.3 timeTransmitter, timeReceiver 割り当て設定の変更方法

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

app_config.h の マクロ GPTP_MODE_MASTER の定義によって、timeTransmitter と timeReceiver を切り替えることが可能です。

```
#define GPTP_MODE_MASTER
```

【注】 timeReceiver 側のみこの記述をコメントアウトして下さい。

4.1.2.4 TDMA サイクルテーブルの変更方法

プロジェクトのディレクトリより、"gtp_sample_thread_entry.c"を開いてください。

```
/common/renesas/application/gtp_sample_thread_entry.c
```

gtp_sample_thread_entry.c の tdma_slot_config によって、TDMA サイクルテーブルを設定します。

```
static tdma_slot_config_t tdma_slot_config[] =
{
    /*    Period,Qgate,Pmask, GPIO,Callback,Preempt,In_ct,Out_ct */
    { 200 * 1000, 0xFC, 0x03, 0x03,  false,  false,false,false},
#ifdef GPTP_MODE_MASTER
    { 200 * 1000, 0x02, 0x03, 0x00,  false,  false,false,false},
#else
    { 200 * 1000, 0x02, 0x03, 0x00,  true,  false,false,false},
#endif /* GPTP_MODE_MASTER */
    { 400 * 1000, 0x01, 0x03, 0x02,  false,  false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x00,  false,  false,false,false}
};
```

実現したい TDMA テーブル構成に応じて内容を変更して下さい。

【注】 各設定内容に関しては"5.1.3 TDMA 動作仕様"を参照してください。

4.1.2.5 TDMA サイクルテーブルの変更方法(Frame Preemption)

プロジェクトのディレクトリより、"gtp_sample_thread_entry.c"を開いてください。

`/common/renesas/application/gtp_sample_thread_entry.c`

gtp_sample_thread_entry.c の tdma_slot_config_preemption によって、TDMA サイクルテーブルを設定します。

```
static tdma_slot_config_t tdma_slot_config_preemption[] =
{
#ifdef GTP_MODE_MASTER
    /* Period,Queue_Gate,Port_mask,GPIO,Callback,Preempt,In_ct,Out_ct */
    { 400000, 0x02/*(0000'0010)*/, 0x03, 0x02, false, false, false, false },
    { 1000, 0x01/*(0000'0001)*/, 0x03, 0x03, false, false, false, false },
    { 100000, 0xFC/*(1111'1100)*/, 0x03, 0x02, false, false, false, false },
    { 299000, 0x01/*(0000'0001)*/, 0x03, 0x03, false, false, false, false },
    { 200000, 0x00/*(0000'0000)*/, 0x03, 0x03, false, false, false, false }
#else /*GTP_MODE_MASTER*/
    /* Period,Queue_Gate,Port_mask,GPIO,Callback,Preempt,In_ct,Out_ct */
    { 400000, 0x02/*(0000'0010)*/, 0x03, 0x02, true, false, false, false },
    { 1000, 0x01/*(0000'0001)*/, 0x03, 0x03, false, false, false, false },
    { 100000, 0xFC/*(1111'1100)*/, 0x03, 0x02, false, false, false, false },
    { 299000, 0x01/*(0000'0001)*/, 0x03, 0x03, false, false, false, false },
    { 200000, 0x00/*(0000'0000)*/, 0x03, 0x03, false, false, false, false }
#endif /*GTP_MODE_MASTER*/
};
```

実現したい TDMA テーブル構成に応じて内容を変更して下さい。

【注】 各設定内容に関しては"5.1.5 Frame Preemption 動作仕様"を参照してください。

4.1.2.6 TDMA サイクルテーブルの変更方法(CQF)

プロジェクトのディレクトリより、"gptp_sample_thread_entry.c"を開いてください。

/common/renesas/application/gptp_sample_thread_entry.c

gptp_sample_thread_entry.c の tdma_slot_config_cqf によって、TDMA サイクルテーブルを設定します。

```
static tdma_slot_config_t tdma_slot_config_cqf[] =
{
#ifdef GPTP_MODE_MASTER || !defined(CQF_SEND_FRAME)
    /* Period,Qgate,Pmask, GPIO,Callback,Preempt,In_ct,Out_ct */
    { 800 * 1000, 0x0B, 0x03, 0x02, false, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x02, false, false,false,false},
    { 800 * 1000, 0x07, 0x03, 0x03, false, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x03, false, false,false,false},
    { 800 * 1000, 0x0B, 0x03, 0x00, false, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x00, false, false,false,false},
    { 800 * 1000, 0x07, 0x03, 0x01, false, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x01, false, false,false,false}
#else /* GPTP_MODE_MASTER */
    /* Period,Qgate,Pmask, GPIO,Callback,Preempt,In_ct,Out_ct */
    { 800 * 1000, 0x0B, 0x03, 0x02, true, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x02, false, false,false,false},
    { 800 * 1000, 0x07, 0x03, 0x03, true, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x03, false, false,false,false},
    { 800 * 1000, 0x0B, 0x03, 0x00, true, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x00, false, false,false,false},
    { 800 * 1000, 0x07, 0x03, 0x01, true, false,false,false},
    { 200 * 1000, 0x00, 0x00, 0x01, false, false,false,false}
#endif /* GPTP_MODE_MASTER */
};
```

実現したい TDMA テーブル構成に応じて内容を変更して下さい。

【注】 各設定内容に関しては"5.1.7 CQF 動作仕様"を参照してください。

4.1.2.7 PROFINET フレーム通信(RT 通信)設定の変更方法

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

app_config.h の マクロ RT_APP_COUNT、RT_APP_TASK_PRIORITY_T 及び、RT_APP_TASK_PRIORITY_RX によって、PROFINET フレーム通信の設定を行います。

```
/* profinet config */
#define RT_APP_COUNT          1
#define RT_APP_TASK_PRIORITY_TX 6      /* profinet send task */
#define RT_APP_TASK_PRIORITY_RX 7      /* profinet recv task */
#define RT_APP_ADDR_DEST      {0x00, 0x11, 0x22, 0x33, 0x44, 0xaa}
```

各項目で以下の設定が可能です：

1. `#define RT_APP_COUNT`

- TDMA サイクルに送信される PROFINET フレームの数です。

【注】上限は 4 フレームです。

2. `#define RT_APP_TASK_PRIORITY_TX`
`#define RT_APP_TASK_PRIORITY_RX`

- それぞれ PROFINET フレームの送信・受信を行うタスクの優先度です。

3. `#define RT_APP_ADDR_DEST`

- RT フレームの宛先 MAC アドレスを指定します。

4.1.2.8 UDP 通信(NRT 通信)設定の変更方法

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

app_config.h の マクロ NRT_APP_TASK_PRIORITY、NRT_APP_PORT、NRT_APP_INTERVAL、NRT_APP_TX_PAYLOAD_SIZE 及び、NRT_APP_TX_COUNT によって、UDP フレーム通信の設定をおこないます。

```
/* udp perf config */
#define NRT_APP_TASK_PRIORITY      2          /* udp perf task */
#define NRT_APP_PORT              5000U
#define NRT_APP_INTERVAL          (10 * 1000) /* 10sec */
#define NRT_APP_TX_PAYLOAD_SIZE   1472U
#define NRT_APP_TX_COUNT          (4U)
```

各項目で以下の設定が可能です：

1. `#define NRT_APP_TASK_PRIORITY`

➤ UDP 通信を行うタスクの優先度です。

2. `#define NRT_APP_PORT`

➤ UDP 通信で使用するポート番号です。

3. `#define NRT_APP_INTERVAL`

➤ UDP フレームのスループット測定間隔([ms])です。

4. `#define NRT_APP_TX_PAYLOAD_SIZE`

➤ 送信される UDP フレームにおけるペイロードのサイズです。

【注】実際に送信されるフレームサイズは、この値から Ethernet ヘッダと UDP ヘッダのサイズを加算したものとなります。

5. `#define NRT_APP_TX_COUNT`

➤ TDMA サイクルに送信される UDP フレームの数です。

【注】UDP フレームの数を多くしすぎると、指定したフレーム数の送信や受信ができない、ログメッセージの出力が止まるなどの影響が出ます。

4.1.2.9 パルスジェネレータ周期の切り替え方法

プロジェクトのディレクトリより、"device_properties.h"を開いてください。

```
/common/renesas/module/ptp/device_properties.h
```

device_properties.h の マクロ GPTP_OUTPUT_PULSE_PERIOD の定義によって、パルスジェネレータ周期を 1[s]か 1[ms]に切り替えることができます。

- パルスジェネレータ周期を 1[s]にする場合、
GPTP_PULSE_PERIOD_SEC を設定してください。

```
#define GPTP_OUTPUT_PULSE_PERIOD GPTP_PULSE_PERIOD_SEC
```

- パルスジェネレータ周期を 1[ms]にする場合、
GPTP_PULSE_PERIOD_MSEC を設定してください。

```
#define GPTP_OUTPUT_PULSE_PERIOD GPTP_PULSE_PERIOD_MSEC
```

4.1.2.10 Frame Preemption の設定

プロジェクトのディレクトリより、"gptp_sample_thread_entry.c"を開いてください。

```
/common/renesas/application/gptp_sample_thread_entry.c
```

gptp_sample_thread_entry.c の qbu_config に Frame Preemption の設定を記述しています。

```
qbu_config_t qbu_config =
{
    /* queue,  qclose */
    { {0x01}, {0x01} },
    {
        /*P1*/{ true, { false, false, 0, 100, true, false, false, false, false } },
        /*P2*/{ true, { false, false, 0, 100, true, false, false, false, false } },
        /*P3*/{ true, { false, false, 0, 100, true, false, false, false, false } },
    }
};
```

各設定事項は以下の通りです：

表 4-2 Frame Preemption の設定(1 of 2)

設定事項	フィールド名	説明	備考
全体設定	preempt_queues	preempt_queue.preempt_queues	プリエンプトキューの指定
	preempt_on_qclose	preempt_queue.preempt_on_qclose	キュークローズイベントによってプリエンプトを行うキューの指定
ポートごとの設定	Enable	preempt_port_ctrl[ポート番号].enable	プリエンプト有効化
	Verify dis-enable	preempt_port_ctrl[ポート番号].config.verify_dis	検証プロセス無効化
	Response dis-enable	preempt_port_ctrl[ポート番号].config.response_dis	検証フレームの応答無効化
	Frag size	preempt_port_ctrl[ポート番号].config.addfragsize	フレームプリエンプト時の最小フラグメントサイズ
	Verify time-out	preempt_port_ctrl[ポート番号].config.tx_verify_time	検証プロセスのタイムアウト時間
	Pre-amble check	preempt_port_ctrl[ポート番号].config.rx_strict_pre	プリアンプルのチェック方式

指定キューがプリエンプトキューの場合のみ有効。

64 バイト単位で調整されます。(64 ~ 256) 各値の割り当ては以下の通りです：

- 0: 64
- 1: 128
- 2: 192
- 3: 256

1 ~ 127 ms

【注】 802.3br で要求される 20%精度を満たすために 5 ms 以上の値が推奨されます。

- true: SFD 以外の全バイトが 0x55 となるようチェックを行う
 - false: 最後の 2 バイトのみチェックを行う
- 【注】 802.3br のトラフィックを使用する場合、true に設定することが推奨されます。

表 4-3 Frame Preemption の設定(2 of 2)

設定事項	フィールド名	説明	備考
ポートごとの設定	No de-code SMD	preempt_port_ctrl[ポート番号].config.rx_br_smd_dis	SMD 複合化の無効化
	Strict check	preempt_port_ctrl[ポート番号].config.rx_strict_br_ctl	検証プロセスにおける、各フレームに対するチェック厳格化の有効化
	TX CRC in-version	preempt_port_ctrl[ポート番号].config.tx_mrcrc_inv	送信フレームにおける、mCRC 計算用 32 ビット XOR マスクの反転有効化
	RX CRC in-version	preempt_port_ctrl[ポート番号].config.rx_mrcrc_inv	受信フレームにおける、mCRC 計算用 32 ビット XOR マスクの反転有効化

4.1.2.11 UDP パケット送信元の MAC アドレスの指定方法

【注】 この設定は、クローズイベントによるプリエンプトを使用することなく(、かつ TDMA サイクルテーブルを"表 5-7 TDMA スロット設定内容(Frame Preemption)"から変更しない状態で)プリエンプシヨンを発生させる際に必要となります。
詳細は"5.1.5.1 クローズイベントによるプリエンプトを用いないプリエンプトの確認方法"を参照してください。

プロジェクトのディレクトリより、"gptp_sample_thread_entry.c"を開いてください。

```
/common/renesas/application/gptp_sample_thread_entry.c
```

マクロ TESTPACKET_MACADDR によって、UDP パケット送信元の MAC アドレスを指定することができます。

```
#define TESTPACKET_MACADDR {0x00,0x00,0x5E,0x00,0x53,0x00}
```

4.1.2.12 PSFP の設定

4.1.2.12.1 フィルタ設定

プロジェクトのディレクトリより、"gptp_sample_thread_entry.c"を開いてください。

```
/common/renesas/application/gptp_sample_thread_entry.c
```

gptp_sample_thread_entry.cにある以下の変数にある項目を変更することで設定できます。

表 4-4 フィルタ設定

設定項目	変数名	項目	説明
ストリームフィルタ設定	g_efp_cfg_qci_stream_filter_entry	MAC_select	MAC アドレス選択
		MAC	MAC アドレス
		MAC_mask	MAC アドレスマスク
ストリームフィルタ有効設定	g_efp_cfg_qci_stream_filter_ena	g_efp_cfg_qci_stream_filter_ena	ストリームフィルタ有効
ストリームゲートフィルタ設定	g_efp_cfg_gating_check_entry	enable	ストリームゲートフィルタ有効
		gateID	ストリームゲートフィルタ ID
		gate mode	ストリームゲートモード
最大 SDU サイズフィルタ設定	g_efp_cfg_sdu_max_entry	enable	最大 SDU サイズフィルタ有効
		maxSDUsize	最大 SDU サイズ
		MSDU mode	最大 SDU サイズモード
フローメーター設定	g_efp_cfg_flow_meter_entry	id_enable	フローメーターID 有効
		id	フローメーターID
		coupling	カップリングフラグ
		mode	フローメーターモード
		red_drop	レッドフレームの破棄
		burst_size	コミットしたバーストサイズ
		info_rate	コミットした情報レート
		enable	フローメーター有効

4.1.2.12.2 PSFP ログ設定変更方法

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

各項目で以下の設定が可能です：

1. `#define PSFP_PORT_MASK`
 - PSFP ログを表示するポートを指定します。
 - フィルタ設定をしているポート番号(マスク)を指定してください。
2. `#define PSFP_STREAM_MASK`
 - PSFP ログを表示するストリーム ID を指定します。
 - フィルタ設定をしているストリーム ID(マスク)を指定してください。

4.1.2.13 CQF の設定

【注】 Qch の動作確認で 3 台構成を行うときのみ必要になります

プロジェクトのディレクトリより、"app_config.h"を開いてください。

```
/common/renesas/application/app_config.h
```

app_config.h の マクロ CQF_SEND_FRAME を定義することで CQF 有効時に timeReceiver から PROFINET フレームを送信します。

```
#define CQF_SEND_FRAME
```

【注】 "4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"に示すような 3 台構成で動作確認を行う場合、timeTransmitter (server)と直接接続している timeReceiver (client)のみこの記述をコメントアウトして下さい。

4.1.3 ビルドおよびボードへのダウンロード

【注】 マルチプロセス版のプロジェクトを使用する場合は、"4.1.4 ビルドとボードへのダウンロード (マルチプロセス版)"の方を参照してください。

4.1.3.1 プロジェクトソースの生成

ビルドを行う前に、FSP Smart Configurator を用いてコード生成を実行してください。

1. IDE から FSP Smart Configurator を起動してください。

【注】 起動方法については"3.3.1.1.1 FSP Smart Configurator を 統合開発環境 から起動する"を参照してください。

2. FSP Smart Configurator 内より、"Generate Project Content"を選択してください。
プロジェクトが生成されます。

4.1.3.2 ビルドとダウンロード

4.1.3.2.1 EWARM の場合

1. プロジェクト生成後、ツールバーより"プロジェクト" -> "メイク" を選択してください。
アプリケーションのビルドが実行されます。
2. ビルド完了後、"プロジェクト" -> "ダウンロードしてデバッグ"を選択してください。
アプリケーションのボードへのダウンロードが開始します。

4.1.3.2.2 e² studio 使用時

1. プロジェクト生成後、ツールバーより"Project" → "build" を選択してください。
アプリケーションのビルドが実行されます。
2. ビルド完了後、"Run" → "Debug"を選択してください。
アプリケーションのボードへのダウンロードが開始します。

4.1.4 ビルドとボードへのダウンロード (マルチプロセス版)

4.1.4.1 プロジェクトソースの生成とビルドの実行 (マルチプロセス版)

1. CPU0 側のプロジェクト(プライマリプロジェクト)にてコード生成を実行してください。

【注】 コード生成方法については"4.1.3.1 プロジェクトソースの生成"を参照してください。

2. CPU0 側のプロジェクトをビルドしてください。

【注】 ビルド実行方法については"4.1.3.2 ビルドとダウンロード"を参照してください。

【注】 EWARM にて xSPI 用プロジェクトをビルドする場合、"4.1.4.1.1 EWARM 使用時のビルドにおける注意事項"に記載された手順が追加が必要となります。

3. CPU1 側のプロジェクトにてコード生成を実行してください。

【注】 先に CPU0 側のプロジェクトでビルドを実行しておかないと、CPU1 側でコード生成を行うことはできません。

4. CPU1 側のプロジェクトをビルドしてください。
5. CPU0 側のプロジェクトをクリーンし、再度ビルドし直してください。

クリーン手順は以下の通りです：

- a. EWARM の場合

- i. "プロジェクト(P)"タブより、"クリーン(L)"を選択してください。
- b. e²studio の場合
 - i. プロジェクトエクスプローラより、クリーンを実行したいプロジェクトを右クリックしてください。
 - ii. コンテキストメニューより、"Clean Project"を選択してください。

4.1.4.1.1 EWARM 使用時のビルドにおける注意事項

EWARM でマルチプロセス版の xSPI 用プロジェクトをビルドする場合、CPU1 側の Debug フォルダ内 bin ファイルが配置されている必要があります。

【注】 配置されていない場合、CPU0 側のプロジェクトにてビルドエラーが発生します。

初回ビルド時は bin ファイルが存在しないため、以下の手順でダミーの bin ファイルを配置してください：

1. ...

4.1.4.2 ダウンロード (マルチプロセス版)

4.1.4.2.1 EWARM の場合

1. デバッガの機能「非対称型マルチコア」を無効化する。

【注】 この操作を行わないと、以降の一部の手順を行うことができません。

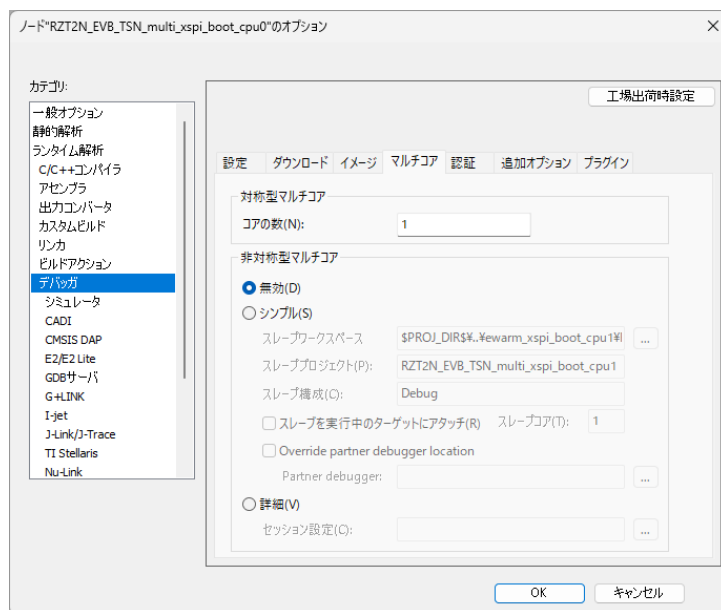


図 4-5 EWARM 「非対称型マルチコア」の無効化

2. (必要な場合)フラッシュメモリの削除

詳細は"3.2.1.1.1 フラッシュメモリの削除"を参照してください。

【注】 この操作を行う場合、あらかじめデバッガの機能「非対称側マルチコア」を無効化しておく必要があります。

3. (xSPI 実行時のみ)

CPU0 側のプロジェクトにて、"プロジェクト(P)"タブ → "ダウンロード(W)" → "ファイルのダウン

ロード...(O)"を選択し、Debug/Exe/ ディレクトリ内に生成されている CPU0 用プログラムの*.out ファイルを選択する。

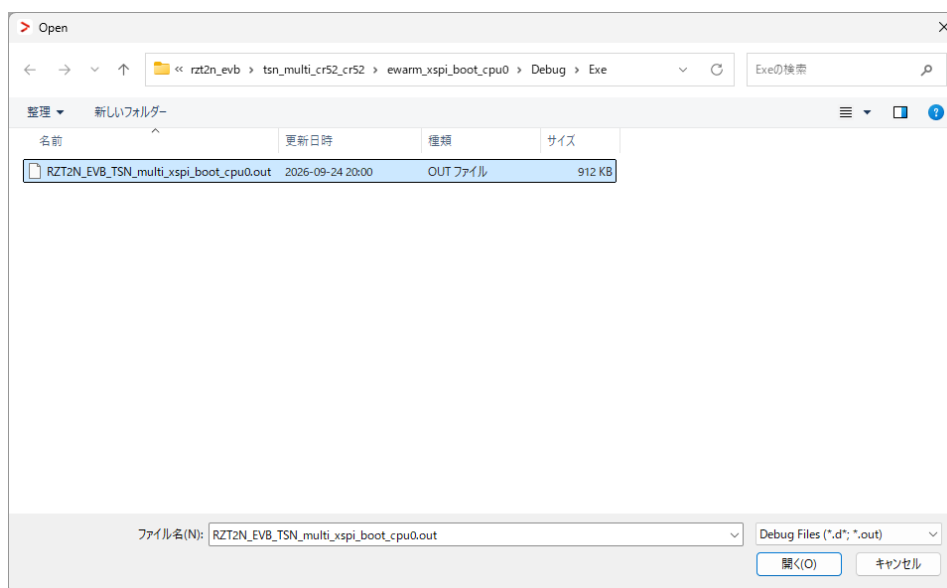


図 4-6 ファイルのダウンロード

【注】 この操作を行う場合、あらかじめデバッガの機能「非対称側マルチコア」を無効化しておく必要があります。

これによりボードの CPU0 側に、CPU0 用のプログラムがダウンロードされます。

4. デバッガの機能「非対称側マルチコア」を有効化する(元の設定に戻す)。

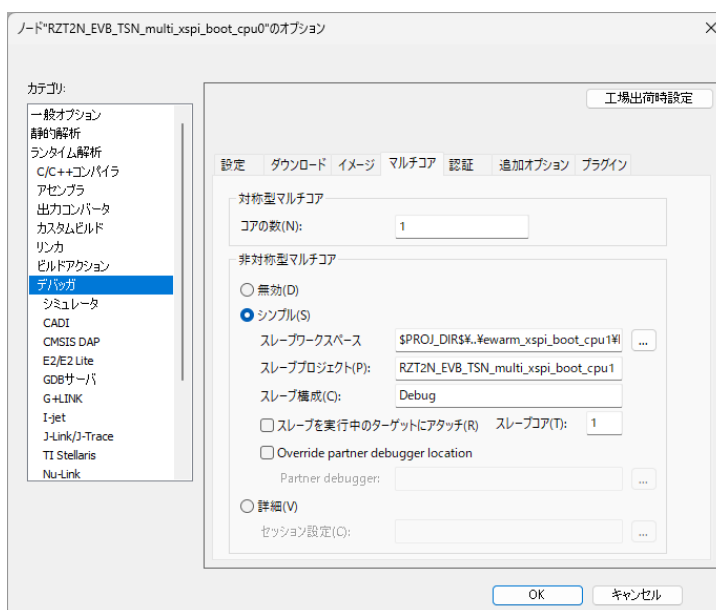


図 4-7 EWARM「非対称型マルチコア」の再有効化

5. CPU0 側でデバッガを起動する。

【注】 RAM 実行時、xSPI 実行時で操作内容が異なります。

- (RAM 実行時)
CPU0 側のプロジェクトにて、「プロジェクト(P)」 → 「ダウンロードしてデバッグ(D)」を選択。
- (xSPI 実行時)
CPU0 側のプロジェクトにて、「プロジェクト(P)」 → 「ダウンロードせずにデバッグ(H)」を選択。

【注】 CPU0 側のデバッガを起動すると、自動で CPU1 側のデバッガも起動します。
(別ウィンドウが立ち上がります。)

6. (xSPI 実行時)

CPU0 側のデバッガを開始させる。

xSPI 実行の場合、このタイミングでボードの CPU1 側に CPU1 用のプログラムがダウンロードされます。

4.1.4.2.2 e² studio の場合

1. (必要な場合)フラッシュメモリの削除

詳細は"3.2.1.1.1 フラッシュメモリの削除"を参照してください。

2. CPU0 側で"Run"タブ → "Debug"を選択し、デバッガを起動してください。

ボードの CPU0 側に、CPU0 用のプログラムがダウンロードされます。

【注】 この時点ではまだデバッガを開始させません。

CPU0 側のデバッガは起動待機状態のまま、次の手順に進んでください。

3. CPU1 側で"Run"タブ → "Debug"を選択し、デバッガを起動してください。

- a. 「起動中のデバッガを停止させますか?」と記載されたダイアログには、「No」を選択してください。

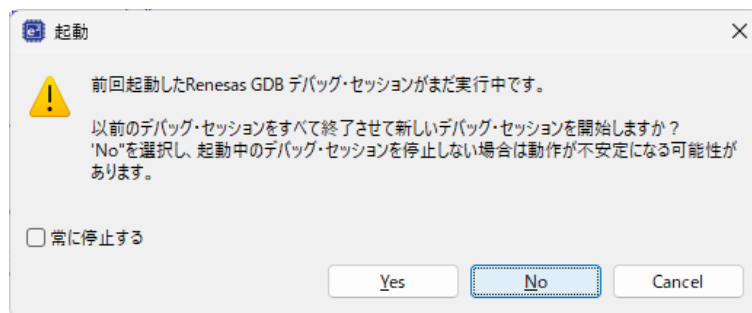


図 4-8 e² studio デバッガ起動前(その 1)

- b. 「Proceed with launch?」と記載されたダイアログが表示された場合は、「Yes」を選択してください。

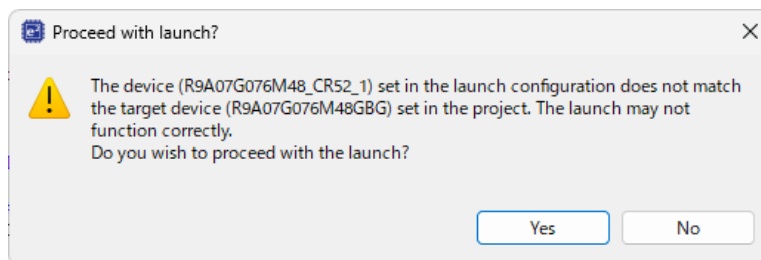


図 4-9 e² studio デバッガ起動前(その 2)

これでボードの CPU1 側に CPU1 用のプログラムがダウンロードされます。

4.2 サンプルアプリケーションの起動

4.2.1 RAM 実行の場合

RAM 実行の場合、ボードへのプログラムのダウンロードが完了するとアプリケーション実行待機状態となります。

以下の手順でアプリケーションを起動して下さい。

【注】 動作確認内容については"5.2 アプリケーション動作確認内容"を参照してください。

4.2.1.1 シングルプロセスの場合

4.2.1.1.1 EWARM 使用時

ツールバーより"デバッグ" → "実行"を選択してください。
アプリケーションが起動します。

4.2.1.1.2 e² studio 使用時

ツールバーより"Run" → "Resume"を選択してください。
アプリケーションが起動します。

4.2.1.2 マルチプロセスの場合

【注】 マルチプロセスプロジェクトの場合、本アプリケーションは CPU1 側で動作します。
CPU0 側のプログラムは何も実行しないようになっています。

4.2.1.2.1 EWARM 使用時

1. CPU0 側のプロジェクトにて、ツールバーより"デバッグ" → "実行"を選択してください。
 2. CPU1 側のプロジェクトにて、ツールバーより"デバッグ" → "実行"を選択してください。
- これにより、アプリケーションが起動します。

【注】 アプリケーションを停止させたい場合、CPU1 側で停止を実行してください。
なお CPU1 側を停止させると、自動で CPU0 側も停止します。

【注】 CPU0 側で停止させると、次のデバッグ時に正しくプログラムを起動できなくなります。
この場合は、CPU1 側から停止を実行してください。
(これによって次回以降は正常に起動できるようになります。)

4.2.1.2.2 e² studio 使用時

1. Debug ウィンドウより、CPU0 側のデバッガを右クリック、"Resume"を実行してください。
【注】 ブレークした場合は再度"Resume"を実行してください。
 2. Debug ウィンドウより、CPU1 側のデバッガを右クリック、"Resume"を実行してください。
【注】 ブレークした場合は再度"Resume"を実行してください。
- これにより、アプリケーションが起動します。

4.2.2 xSPI 実行の場合

xSPI 実行の場合、別途操作は不要です。
プログラムをダウンロードした後は、ボードの電源を再投入することでプログラムが起動します。

【注】 動作確認内容については"5.2 アプリケーション動作確認内容"を参照してください。

4.3 サンプルアプリケーションの動作確認の例

以下に「RSK ボード 2 台を使用し、1 台目を timeTransmitter、2 台目を timeReceiver として動作させる」構成での動作確認手順を示します。

1. デバイスのマスタ/スレーブ設定を行う。
 - A) 1 台目を timeTransmitter に設定する :
 - ① デバイスを *timeTransmitter* に設定する。(参照 : 4.1.2.3)
 - ② MAC アドレスを `11:22:33:44:55:aa` に設定。(参照 : 4.1.2.1)
 - B) 2 台目を timeReceiver に設定する :
 - ① デバイスを *timeReceiver* に設定する。(参照 : 4.1.2.3)
 - ② MAC アドレスを `11:22:33:44:55:bb` に設定。(参照 : 4.1.2.1)

※ 設定後は、それぞれのデバイスでプログラムをビルド・書込みを行います。

2. 各デバイスの ETH0 同士を Ethernet ケーブルで接続する。

構成は以下の通りとなります :

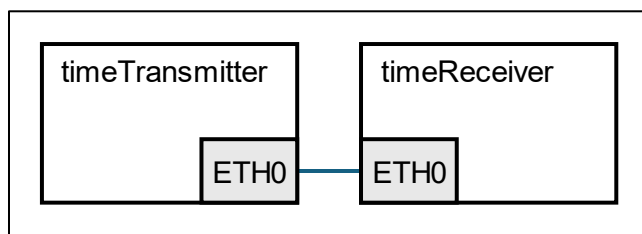


図 4-10 接続構成例

3. 各デバイスを起動する。
4. (以降、5.2 の方法で gPTP + TDMA の動作確認が可能です。)

4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)

以下に「RSK ボード 3 台を使用し、1 台目を timeTransmitter、2 台目と 3 台目を timeReceiver として動作させる」構成での動作確認手順を示します。

1. デバイスのマスタ/スレーブ設定を行う。

A) 1 台目を timeTransmitter に設定する：

- ① デバイスを *timeTransmitter* に設定する。(参照：4.1.2.3)
- ② MAC アドレスを `11:22:33:44:55:aa` に設定。(参照：4.1.2.1)
- ③ IP アドレスを `192.168.10.170` に設定。(参照：4.1.2.2)

B) 2 台目を timeReceiver1 に設定する：

- ① デバイスを *timeReceiver* に設定する。(参照：4.1.2.3)
- ② MAC アドレスを `11:22:33:44:55:bb` に設定。(参照：4.1.2.1)
- ③ IP アドレスを `192.168.10.171` に設定。(参照：4.1.2.2)
- ④ CQF 設定 ``#define CQF\_SEND\_FRAME`` をコメントアウト(参照：4.1.2.13)

C) 3 台目を timeReceiver2 に設定する：

- ① デバイスを *timeReceiver* に設定する。(参照：4.1.2.3)
- ② MAC アドレスを `11:22:33:44:55:cc` に設定。(参照：4.1.2.1)
- ③ IP アドレスを `192.168.10.172` に設定。(参照：4.1.2.2)

※ 設定後は、それぞれのデバイスでプログラムをビルド・書込みを行います。

2. 各デバイスの ETH 同士を Ethernet ケーブルで接続する。

構成は以下の通りとなります：

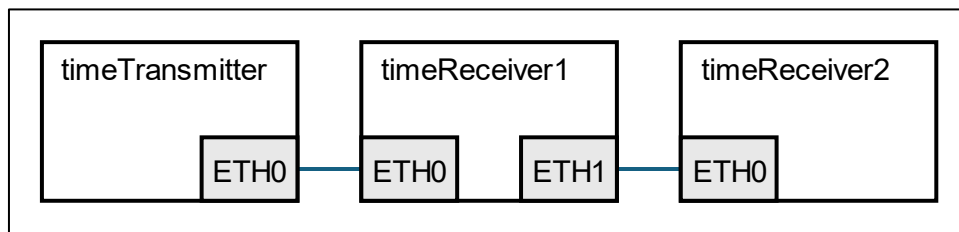


図 4-11 接続構成例(Qch)

3. 各デバイスを起動する。

5. TSN サンプルアプリケーション

本サンプルアプリケーションはボード同士の対向環境で実現される、TSN アプリケーションとなります。

5.1 アプリケーション動作概要

ボードの ETH0 を通じて各デバイス同士を接続(*2)(*3)することによって、デバイス間の時刻同期が開始されます。

時刻同期に成功すると TDMA が起動し、アプリケーション動作を開始します。

gPTP 開始時、各デバイスは timeTransmitter または timeReceiver に割り当てられます。(*1)

【注】*1: 本アプリケーションではパケット受信側(server)デバイスが timeTransmitter として、パケット送信側(client)デバイスが timeReceiver として動作する仕様となっています。[TSN-000-0073]

そのため、事前にデバイスを timeTransmitter か timeReceiver のどちらに割り当てるかを設定しておく必要があります。

timeTransmitter, timeReceiver 割り当ての設定方法に関しては"4.1.2.3 timeTransmitter, timeReceiver 割り当て設定の変更方法"を参照してください。

【注】*2: ETH1 でも接続することは可能ですが、PROFINET フレームは ETH0 からしか送信されません。したがって timeTransmitter 側のデバイスと接続するポートには ETH0 を使用する必要があります。

【注】*3: ETH2 は使用不可。

【注】*4: マルチプロセスプロジェクトの場合、本アプリケーションは CPU1 側で動作します。CPU0 側のプログラムは何も実行しないようになっています。

本アプリケーションは TDMA 動作が有効で、かつ gPTP の時刻同期の確立後、RT(Real Time)通信として PROFINET フレーム、NRT(Non-Real Time)通信として UDP フレームの送受信を行います(*1)(*2)。

本アプリケーションにおける TDMA では、PROFINET フレームは専用の TDMA スロットで転送され(*3)、UDP フレームはデフォルトフレーム用 TDMA スロットで転送されるように設定されています。(*2)

Frame Preemption 有効時は、PROFINET フレーム(RT)はエクスプレスキューで送信され、UDP/デフォルトフレーム(NRT)はプリエンプトキューで送信されるように設定されています。

したがって、PROFINET フレームによって UDP フレームがプリエンプトされるように動作します。(*4)

Qci 有効時は、PROFINET フレームおよび UDP フレームをフィルタ対象として timeTransmitter が受信した際にフィルタを行います。

Qch 有効時は、PROFINET フレームの送受信を周期的に行います。
詳しくは"2.5 IEEE802.1Qch (CQF)"を参照してください。

また、この時 PROFINET フレームは上限の 4 フレームが送受信されます。LLDP 有効時は、LLDP フレームの送信も行います。

【注】*1: timeReceiver がフレームを送信(client として動作)し、timeTransmitter が受信(server として動作)します。

【注】*2: TDMA サイクルのテーブルは編集可能です。
編集方法は"4.1.2.4 TDMA サイクルテーブルの変更方法"を参照して下さい。

【注】*3: PROFINET フレームは RT フレームのテスト用として使用しているものであり、本アプリケーションにおいて PROFINET プロトコルの動作は行っていません。

【注】*4: ただし hold_req 有効時の間はプリエンプトキューがブロックされ続ける(UDP フレームが送信されなくなる)ため、プリエンプションは発生しなくなります。

各フレーム送受信は以下のように行われます:

- PROFINET フレームは TDMA サイクルごとに 1 回ずつ送信されます。
 - 一度に送信する PROFINET フレーム数は変更することが可能です(*)。
変更方法は"4.1.2.7 PROFINET フレーム通信(RT 通信)設定の変更方法"を参照してください。
 - 【注】上限は 4 フレームです。
 - 複数送信する場合、送信されるフレームごとに 0x101 から連番でフレーム ID が割り振られます。
 - ✧ フレームが送られる Queue は、そのフレームのフレーム ID によって決定されます。
(下表参照)
 - ✧ timeTransmitter が PROFINET フレームを受信すると、そのフレーム ID に対応したボード上の LED をトグル動作させます。
 - ✧ 各フレーム ID における Queue と LED の対応は以下の通りとなります：

表 5-1 各フレーム ID における Queue と LED の対応

フレーム ID	Queue	LED
0x101	2	LED0
0x102	3	LED1
0x103	4	LED2
0x104	5	LED3

- UDP フレームも TDMA サイクルごとに 1 回ずつ送信されます。
 - 一度に送信する UDP フレーム数や、フレームのサイズは変更することが可能です。
変更方法は"4.1.2.8 UDP 通信(NRT 通信)設定の変更方法"を参照してください。

本サンプルアプリケーションでは動作確認用の機能として以下の動作が実装されています。

- UART によるログ出力によって動作状態を表示します。
- GPIO ピンを用いて gPTP 動作確認用クロック波形の出力と、TSN 動作状態確認用の出力を行います。

5.1.1 メニュー

デバイス起動時はメニューが起動し、機能の選択を行うことでアプリケーションが起動します。

本アプリケーションは、各ボードの"USB to Serial Port"から UART 経由でメニューの出力および入力を行います。

各ボードの"USB to Serial Port"の割り当ておよびボード上の配置を以下に示します。

表 5-2 "USB to Serial Port"の割り当て

ボード	割り当て
RZ/T2M-RSK	CN16
RZ/N2L-RSK	
RZ/T2N-Evb	J33

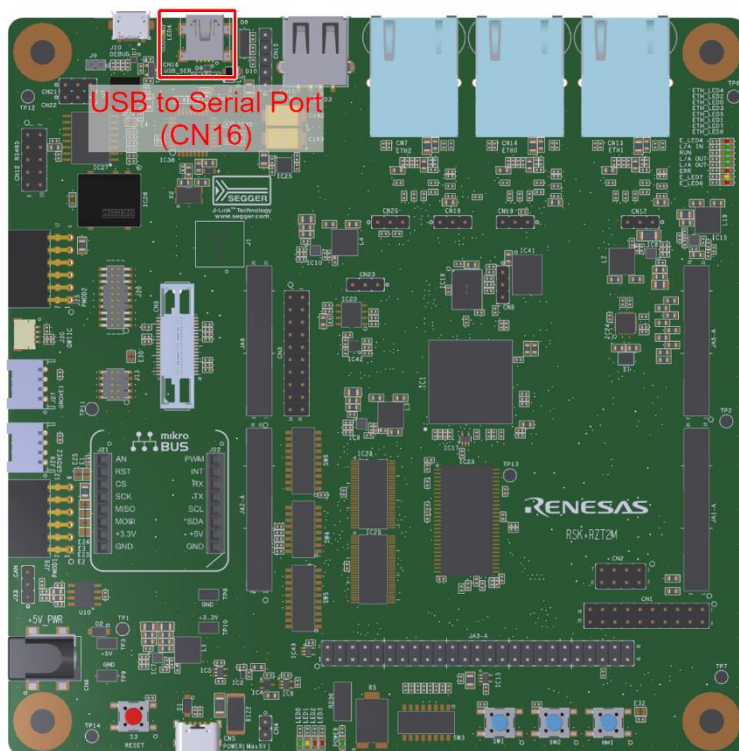


図 5-1 RZ/T2M-RSK ボード図 (USB to Serial Port)

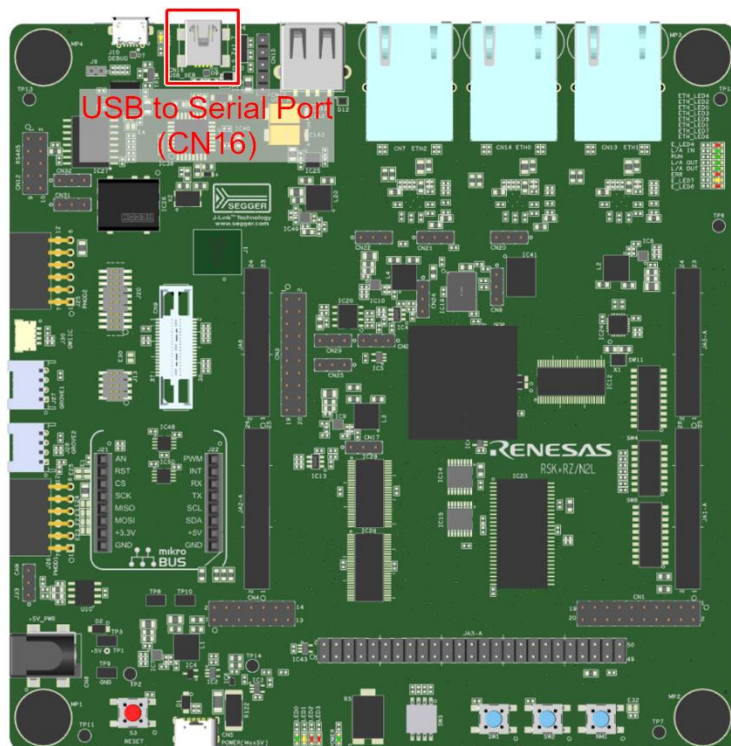


図 5-2 RZ/N2L-RSK ボード図 (USB to Serial Port)

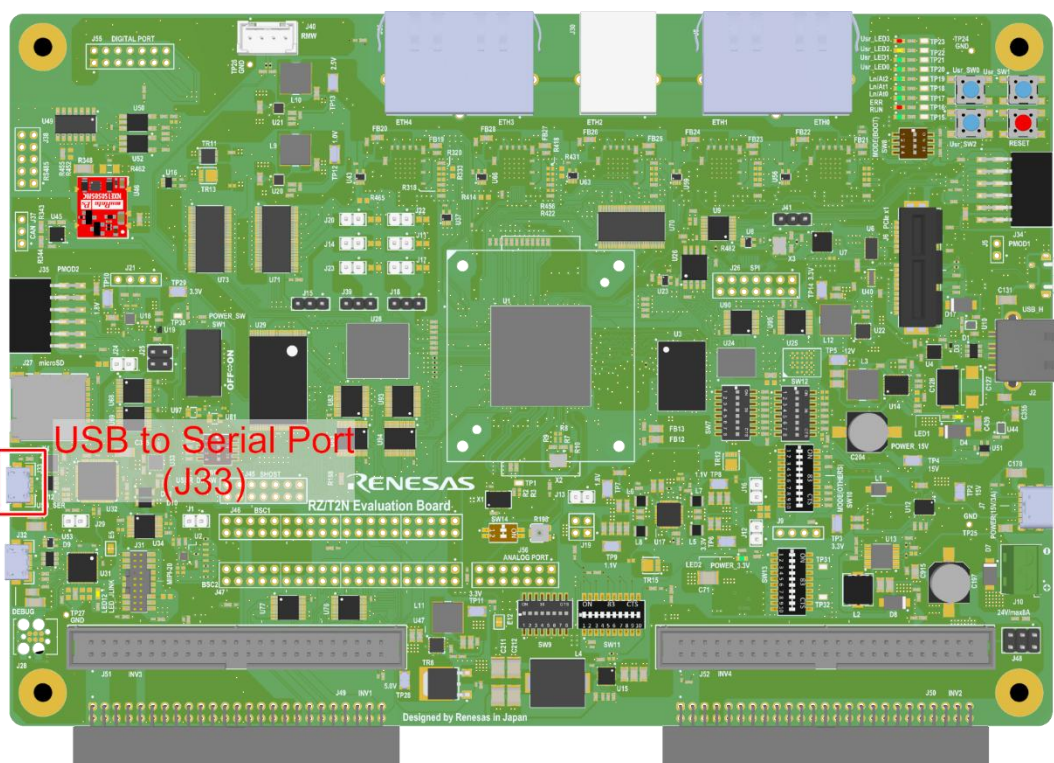


図 5-3 RZ/T2N-Evb ボード図 (USB to Serial Port)

メニューは以下のように表示されます。

```
[Menu]
[0]: gPTP
[1]: Preemption
[2]: Preemption(Hold Request)
[3]: Qch
[4]: Qci
[5]: LLDP
>>
```

それぞれの機能は以下の通りです。

表 5-3 メニューで選択できる機能と内容

番号	項目名	内容
0	gPTP	gPTP
1	Preemption	gPTP+Frame Preemption
2	Preemption(Hold Request)	gPTP+Frame Preemption(Hold Request 有効)
3	Qch	gPTP+CQF
4	Qci	gPTP+PSFP
5	LLDP	gPTP+LLDP

起動したい機能に合わせて数字を入力してください。

5.1.2 gPTP 動作仕様

gPTP の動作仕様は以下の通りです。

表 5-4 gPTP 仕様

SYNC 送信間隔	PDELAY 送信間隔
125 msec	1000 msec

5.1.3 TDMA 動作仕様

TDMA の動作仕様は以下の通りです。

- 開始/停止 タイミング

表 5-5 TDMA 動作タイミング

デバイス	TDMA 開始タイミング	TDMA 停止タイミング
timeTransmitter	"NETDEV_UP" 通知時	"NETDEV_DOWN" 通知時
timeReceiver	"CLOCK_SYNCED" 通知時	"CLOCK_NOTSYNC" 通知時

- スロットごとの設定(TDMA サイクルテーブル)

- 【注】 TDMA サイクルのテーブルは編集可能です。
編集方法は"4.1.2.4 TDMA サイクルテーブルの変更方法"を参照して下さい。
- 【注】 Frame Preemption 有効時の TDMA サイクルテーブルは以下のものとは異なります。
詳しくは"5.1.5 Frame Preemption 動作仕様"を参照して下さい。
- 【注】 CQF 有効時の TDMA サイクルテーブルは以下のものとは異なります。
詳しくは"5.1.7 CQF 動作仕様"を参照して下さい。

表 5-6 TDMA スロット設定内容

Slot	Period [us]	Queue-Gate (*2)(*3)								Port Mask (*2)				gpio pattern (TDMAOUTn)	Trigger (activate callback)	Preemption hold_req	In cut- through	Out cut- through	Description (*3)
		7	6	5	4	3	2	1	0	2	1	0	1	0					
0	200	0	0	0	0	0	0			0	0	H	H	false	false	false	false	false	RT traffic slot
1	200							0		0	0	L	L	true(*1)	false	false	false	false	PTP traffic slot
2	400								0	0	0	H	L	false	false	false	false	false	NRT traffic slot
3	200											L	L	false	false	false	false	false	Guard band

- 【注】 *1: timeTransmitter (server) では false、timeReceiver (client) では true に設定してください。
- 【注】 *2: 「O」がオープン、空白がクローズされることを示しています。
- 【注】 *3: 本アプリケーションでは、キュー0がNRTフレーム、キュー1がPTPフレーム、キュー2~7がRTフレームにそれぞれ割り当てられています。

- TDMA サイクルの周期(スロットごとの Period 時間の合計)

: **1000[us]**

5.1.4 RT フレーム・NRT フレーム動作仕様

RT フレーム、NRT フレーム有効時、各デバイスはそれぞれ以下の動作を行います。

- timeReceiver 側
 - TSN が実行されている間、Slot1 開始タイミングをトリガーとして PROFINET テストフレームと UDP フレームをそれぞれ GMAC へ送信します。
 【注】 ETHSW で各フレームが実際に送信されるのは以下のタイミングとなります：
 - PROFINET Slot0
 - UDP Slot2
- timeTransmitter 側
 - PROFINET テストフレームを受信したタイミングで LED をトグル操作して点滅させます。
 ➤ どの LED をトグル操作するかはフレーム ID によって決定されます。
 【注】 対応表については"5.1 アプリケーション動作概要"を参照してください。

5.1.5 Frame Preemption 動作仕様

本サンプルアプリケーションでは TDMA を用いることでフレーム送信タイミングを制御し、異なるキューのフレーム送信タイミングを意図的に衝突させることによってプリエンプットの発生する状況を疑似的に作り出しています。

これにより、デバイスの持つ Frame Preempt 機能の確認が可能です。そのため、Frame Preemption 有効時は通常とは異なる専用の TDMA サイクルテーブルが適用されます。
 Frame Preemption 有効時の TDMA サイクルテーブルは以下の通りになります。

【注】 TDMA サイクルのテーブルは編集可能です。
 編集方法は"4.1.2.5 TDMA サイクルテーブルの変更方法(Frame Preemption)"を参照して下さい。

表 5-7 TDMA スロット設定内容(Frame Preemption)

Description (*3)		In cut- through	Out cut- through	Preemption hold_req	Trigger (activate callback)	gpio pattern (TDMaOutn)		Port Mask (*2)		Queue-Gate (*2)(*3)							Period [us]	Slot				
						0	1	0	1	0	1	2	0	1	2	3	4	5	6	7		
PTP traffic slot		false	false	false	true(*1)	L	L	0	0	0	0		0								400	0
NRT traffic slot		false	false	false	false	L	H	0	0	0	0	0									1	1
RT traffic slot		false	false	false	false	H	H	0	0	0	0					0	0	0	0	0	100	2
NRT traffic slot		false	false	false	false	L	H	0	0	0	0		0								299	3
Guard band		false	false	false	false	L	L														200	4

- 【注】 *1: timeTransmitter (server) では false、timeReceiver (client) では true に設定してください。
- 【注】 *2: 「O」がオープン、空白がクローズされることを示しています。
- 【注】 *3: 本アプリケーションでは、キュー0がNRTフレーム、キュー1がPTPフレーム、キュー2~7がRTフレームにそれぞれ割り当てられています。

Frame Preemption の設定は以下の通りとなっています。

- プリエンプトキュー : Queue0
- キュークローズによるプリエンプトが有効 : Queue0

表 5-8 Frame Preemption 設定内容

Port	Enable	Verify dis-enable	Response dis-enable	Frag size	Verify time-out [ms]	Pre-amble check	No de-code SMD	Strict check	TX CRC in-version	RX CRC in-version
1	True	False	False	0	100	True	False	False	False	False
2	True	False	False	0	100	True	False	False	False	False
3	True	False	False	0	100	True	False	False	False	False

5.1.5.1 クローズイベントによるプリエンプトを用いないプリエンプトの確認方法

クローズイベントによるプリエンプトを使用しないプリエンプシヨン発生は、以下の方法で確認が可能です。

1. qbu_config の qclose を 0x00 に設定してください。
(参照 : "4.1.2.10 Frame Preemption の設定")
2. 2 台構成の上、timeReceiver (client) に PC 等の外部デバイス(UDP パケット送信用)を接続してください。(図 5-4 参照。)
3. 外部デバイスから、以下の設定の UDP パケットを timeReceiver(client)に向けて継続的に送信してください。

表 5-9 外部デバイスの UDP パケット設定

項目	設定内容	備考
サイズ	1512 Byte	
送信間隔	8000 Packet/s	
VLAN タグ	なし(Untagged)	
宛て先 MAC アドレス	11:22:33:44:55:BB	timeReceiver の MAC アドレス
送信元 MAC アドレス	マクロ"TESTPACKET_MACADDR"で指定されているアドレス	参考: 4.1.2.11
宛て先 IP アドレス	192.168.10.170	timeTransmitter の IP アドレス
送信元 IP アドレス	(任意のアドレス)	各デバイスと重複しないもの

これにより、timeReceiver (client) は上記 UDP パケットを timeTransmitter (server) へ転送し続けている状態にします。

4. 上記の状態ではボード同士の時刻同期を開始させることで、プリエンプトを発生させることができます。
 - 時刻同期の開始については、timeReceiver (client) と timeTransmitter (server) をつなぐ LAN ケーブルを一度切断し、再接続することで行います。
 - プリエンプトが発生しなかった場合、再度上記の再接続を行ってください。

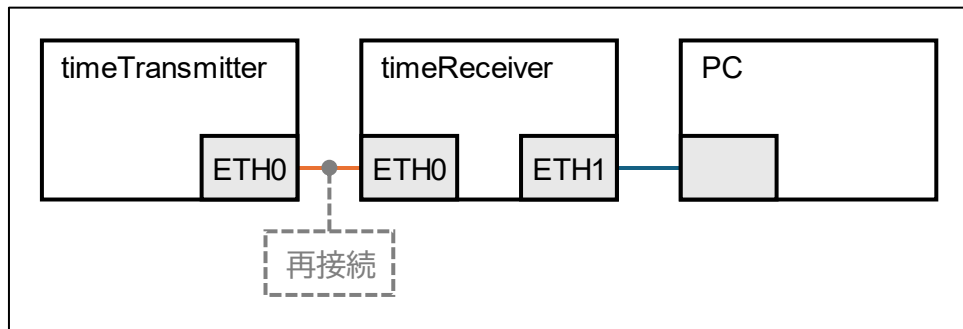


図 5-4 接続構成例(クローズイベントによるプリエンプトを用いないプリエンプトの確認)

5.1.6 PSFP 動作仕様

PSFP 有効時にフレームを受信する側だけにフィルタ設定が行われます。

デフォルトの動作では、ポート 0 の Stream ID(SID) 0 が使用されます。

ポート 0, SID0 の各 PSFP 設定内容は以下の通りです。

各設定項目については"4.1.2.12.1 フィルタ設定"を参照してください。

表 5-10 PSFP デフォルト設定(ポート 0, SID0)

機能	設定項目	設定内容	詳細説明
ストリームフィルタ設定	MAC_select	1	送信元 MAC アドレスが {00:11:22:33:44:BB} のフレーム(※)をフィルタリング対象とする ※: timeReceiver から送信されるフレームが該当。
	MAC	{00:11:22:33:44:BB}	
	MAC_mask	{00:FF:FF:FF:FF:FF}	
ストリームフィルタ有効設定	([0])	1	SID:0 を有効化 SID:0 でのストリームゲートフィルタを有効化
	enable	true	
	gateID	0	
ストリームゲートフィルタ設定	gate mode	false	TDMAOUT0 を使用するフレーム破棄状態のクリアは不要 SID:0 での最大 SDU サイズフィルタを有効化 サイズ(FCS 含む)が 100Byte 以上のデータ(※)を破棄する。 ※: NRT フレーム(UDP フレーム)が該当。 フレーム破棄状態のクリアは不要
	enable	true	
	maxSDUsize	100	
フローメーター設定	MSDU mode	false	SID:0 でのフローメーターを有効化 SID:0 ではメーター0 を使用する メーター0 のカップリングを有効化 フレーム破棄状態のクリアは不要 赤色フレームを破棄する CBS(トークン最大量) CIR(トークン追加量) メーター0 を有効化
	id_enable	true	
	id	0	
	coupling	true	
	mode	false	
	red_drop	true	
	burst_size	0xFFFF	
	info_rate	0xA000	
	enable	true	

上記設定により、timeReceiver から送信された RT, NRT フレームが SID0 に分類されフィルタリング対象となります。

【注】 PTP フレームは ETHSW により管理フレームとして認識されるため、フィルタリングの対象外となります。
詳細は各ボードのハードウェアマニュアルより EFP(Extended Frame Parser)の章を参照してください。

表 5-11 ハードウェアマニュアルにおける EFP 関連の章

ハードウェアマニュアル	章
RZ/N2L Group	28.3.8 Extended Frame Parser Registers
User's Manual: Hardware	28.4.19 Extended Frame Parser (EFP)
RZ/T2N Group	29.3.8 Extended Frame Parser Registers
User's Manual: Hardware	29.4.19 Extended Frame Parser (EFP)

デフォルト設定では、フレーム受信時スロットの TDMAOUT0 の値に基づき、NRT フレームのみストリームゲートフィルタによって破棄されます。

(NRT フレーム受信時の TDMAOUT0 は L に設定されています。詳細は"5.1.3 TDMA 動作仕様"を参照してください。)

RT フレームはストリームゲートフィルタ、最大 SDU サイズフィルタ、フローメーターの順にそれぞれ破棄されることなく通過します。

その他の動作は gPTP 動作と同等となります。
詳細は"5.1.2 gPTP 動作仕様"を参照してください。

5.1.7 CQF 動作仕様

本サンプルアプリケーションでは TDMA を用いることでフレーム送信タイミングを制御しており、CQF 有効化時は通常とは異なる専用の TDMA サイクルテーブルが適用されます。
CQF 有効時の TDMA サイクルテーブルは以下の通りになります。

【注】 TDMA サイクルのテーブルは編集可能です。

編集方法は"4.1.2.6 TDMA サイクルテーブルの変更方法(CQF)"を参照して下さい。

表 5-12 TDMA スロット設定内容(CQF)

Slot	Period [us]	Queue-Gate (*2)(*3)								Port Mask (*2)				gpio pattern (TDMAOUTn) (*4)		Trigger (activate callback)	Preemption hold_req	In cut- through	Out cut- through	Description (*3)
		7	6	5	4	3	2	1	0	2	1	0	1	0						
0	800					0	0	0	0	0	0	H	L	true(*1)	false	false	false	EVEN(ID=0x101)		
1	200												H	L	false	false	false	Guard band		
2	800						0	0	0	0	0	H	H	true(*1)	false	false	false	ODD(ID=0x102)		
3	200												H	H	false	false	false	Guard band		
4	800					0		0	0	0	0	L	L	true(*1)	false	false	false	EVEN(ID=0x103)		
5	200												L	L	false	false	false	Guard band		
6	800						0	0	0	0	0	L	H	true(*1)	false	false	false	ODD(ID=0x104)		
7	200												L	H	false	false	false	Guard band		

【注】*1: timeTransmitter (server) では false、timeReceiver (client) では true に設定してください。

【注】*2: 「0」がオープン、空白がクローズされることを示しています。

【注】*3: 本アプリケーションでは、キュー0がNRTフレーム、キュー1がPTPフレーム、キュー2~3がRTフレームにそれぞれ割り当てられています。
また、キュー2~3がCQFキューに設定されています。

【注】*4: 本アプリケーションでは、CQF キューの EVEN/ODD の切り替えは TDMAOUT0 に従うよう設定されています。
また上記 TDMA テーブルのキューゲート設定により、片方がキューイングしている間にもう片方のキューがオープンしフレームが送信されるようになっています。

表 5-13 CQF キューの動作

TDMAOUT0	CQF キュー	キュー イング 先	キュー オープン (フレーム 送信)
Low	EVEN	Queue2	Queue3
High	ODD	Queue3	Queue2

各スロットでの TDMAOUT および送信される profinet フレームのフレーム ID の対応は以下の通りです。

表 5-14 TDMA スロットとフレーム ID 対応表

Slot	TDMAOUT1	TDMAOUT0	フレーム ID 送信される
0	H	L	0x101
2	H	H	0x102
4	L	L	0x103
6	L	H	0x104

各 CQF サイクルは、TDMA サイクルにおける Slot 0,2,4,6 のタイミングで切り替わります。

よって、TDMA サイクルの 1 周期は CQF サイクル 4 周期分に相当します。

CQF 有効時、profinet フレームは以下のような流れでキューイング/デキューイング(送信)されます：

		Slot0		Slot2	
timeReceiver	Application	0x101			
			↓Send		
	ETHSW	Queue3	Queue2 0x101	Queue3	Queue2 0x101
	Queue Gate		Close	Close	↓Send
timeTransmitter	ETHSW	Queue2	Queue3	Queue2	Queue3 0x101
					↓Send
	Application			0x101	

図 5-5 CQF によるキュー動作 (2 台構成時)

また、"4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"に示すような 3 台構成の場合、キュー動作は以下の通りとなります：

		Slot0		Slot2		Slot4	
timeReceiver2	Application	0x101					
			↓Send				
	ETHSW	Queue3	Queue2 0x101	Queue3	Queue2 0x101	Queue3	Queue2
	Queue Gate		Close	Close	↓Send		Close
timeReceiver1	ETHSW	Queue2	Queue3	Queue2	Queue3 0x101	Queue2	Queue3 0x101
	Queue Gate	Close			Close	Close	↓Send
timeTransmitter	ETHSW	Queue3	Queue2	Queue3	Queue2	Queue3	Queue2 0x101
							↓Send
	Application					0x101	

図 5-6 CQF によるキュー動作 (3 台構成時)

このようにフレームの受信と送信が交互に切り替わるため、ホップ数によって受信までの時間が 1CQF サイクルずつ増加する(ずれていく)ようになります。

実際の挙動については"5.2.2.3 CQF 動作確認"を参照してください。

5.1.8 LLDP 動作仕様

LLDP 有効時、定期的に LLDP フレームを送信します。

また、MIB 設定(SNMP)にも対応しており、コンソールを使って確認することができます。

その他の動作は gPTP 動作と同じです。詳しくは”5.1.2 gPTP 動作仕様”を参照してください。

5.2 アプリケーション動作確認内容

各種出力によって確認できる内容を以下に示します。

5.2.1 ログ出力内容

本アプリケーションは、各ボードの USB to Serial Port から UART 経由でログメッセージを出力します。ボード上でのコネクタ配置は"5.1.1 メニュー"を参照してください。

ログメッセージによって以下の内容が確認できます。

- gPTP 動作状況
- PROFINET フレーム送信状況(ロストフレーム発生時のみ)
- Frame Preemption 動作状況
- PSFP 動作状況
- MIB 情報(LLDP)

【注】 CQF 機能にはログで表示する情報がありません。

5.2.1.1 gPTP ログメッセージ

gPTP 関連のログメッセージは以下の通りです。

5.2.1.1.1 接続時のログ出力

デバイス接続時、以下のログが出力されます。

出力内容はデバイスの割り当て(timeTransmitter/timeReceiver)によって変化します。

- timeTransmitter 側

NETDEV_UP. port=[0]

- timeReceiver 側

NETDEV_UP. port=[0]
CLOCK_SYNCED

各ログメッセージは以下の状態を示しています。

- " NETDEV_UP. portl=[0]"
 - ポート 0 がリンクアップした
- "CLOCK_SYNCED"
 - timeTransmitter clock との時刻補正が安定した

5.2.1.1.2 定期的なログ出力

5[s]周期で定期的に gPTP の各情報がログに出力されます。
 これによって、デバイスごとの gPTP 稼働状況を確認することが可能です。

出力内容はデバイスの状態(未接続時/timeTransmitter 動作時/timeReceiver 動作時)によって変化します。

● 未接続時

```
-----NO SYNC ROLE-----
instance number 0 with domain number 0x0
CLOCK_IDENTITY 0x 00 11 22 33 44 aa 00 00
AS_CAPABLE P1 0 | P2 0 | P3 0 |
PORT_STATE P1 3 | P2 3 | P3 3 |

-----

MEAN_LINK_DELAY P1 0 | P2 0 | P3 0 |
time 4 999881808
=====
```

● timeTransmitter 動作時

```
-----MASTER-----
instance number 0 with domain number 0x0
CLOCK_IDENTITY 0x 00 11 22 33 44 aa 00 00
AS_CAPABLE P1 1 | P2 0 | P3 0 |
PORT_STATE P1 6 | P2 3 | P3 3 |

-----

MEAN_LINK_DELAY P1 13 | P2 0 | P3 0 |
time 14 999880608
=====
```

● timeReceiver 動作時

```
-----SLAVE-----
GM_CLOCK_IDENTITY 0x 00 11 22 33 44 aa 00 00
STEPS_REMOVED 1
OFFSET_FROM_MASTER -53
instance number 0 with domain number 0x0
CLOCK_IDENTITY 0x 00 11 22 33 44 bb 00 00
AS_CAPABLE P1 1 | P2 0 | P3 0 |
PORT_STATE P1 9 | P2 3 | P3 3 |

-----

MEAN_LINK_DELAY P1 15 | P2 0 | P3 0 |
time 14 996934159
=====
```

各項目の示す内容は以下の通りです。

表 5-15 定期的な PTP ログ出力の表示内容

項目	説明	補足
(先頭行)	デバイスの現在の割り当て	NO_SYNC_ROLE : 未接続 MASTER : Master(timeTransmitter)として動作中 SLAVE : Slave(timeReceiver)として動作中
GM_CLOCK_IDENTITY (*1)	GrandMaster の CLOCK_IDENTITY	
STEP_REMOVED (*1)	GrandMaster からのステップ数	GrandMaster からの中継デバイス数に応じて増加。 (直接接続された場合 1 となる。)
OFFSET_FROM_MASTER (*1)	GrandMaster の時刻に対するオフセット値	
instance number...	稼働しているインスタンスの番号と、 それが属するドメインの番号	
CLOCK_IDENTITY	デバイスの CLOCK_IDENTITY	本アプリケーションでは、 CLOCK_IDENTITY は MAC アドレスの値を基に自動生成される。
AS_CAPABLE	ポートの対向デバイスが AS_CAPABLE かどうか	1: 有効 0: 無効
PORT_STATE	ポートの状態	3: PTP は動作していない 6: 時刻情報を送信 9: 時刻情報を受信
MEAN_LINK_DELAY	ポートごとに算出された伝搬遅延値	
time	PTP タイマの値	時刻同期によって調整されるタイマ

【注】*1: timeReceiver でのみ表示される項目。

また、これに加えて有効にした機能に応じてログ出力内容が追加されます。詳細はそれぞれの章を参照してください。

5.2.1.2 PROFINET 送受信ログメッセージ

PROFINET フレームに関するログメッセージは以下の通りです。

【注】このメッセージはロストフレーム検出時(*1)にのみ出力されます。

```
profinet(0101):lost frame. [0x41e0,0xef80]
```

【注】*1: ロストフレーム検出は PROFINET フレーム受信時のタイミングで行われます。

新たな PROFINET フレームを受信した際、前回受信した PROFINET フレームとの CycleCounter 値(フレーム送信ごとに一定値インクリメントされる)による整合性チェックが行われ、その結果をもとにロストフレームの発生を検出します。

したがってロストフレームが実際に検出されるのは、ロストフレームが発生した直後の PROFINET フレーム受信のタイミングとなります。

5.2.1.3 Frame Preemption ログメッセージ

Frame Preemption に関するログメッセージは以下の通りです。

port	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
<0>	0x3,	0,	0	0x00,	0x04	0x00000000,	0x00000012,	0x0000,
[8]	[9]	[A]	[B]	[C]	[D]	[E]	[F]	[G]
0x0000,	0x00000012,	0x00000000	0x01,	0x00,	0x01,	0x00,	0x01,	0x03

各項目は以下の情報を示しています。

表 5-16 Frame Preemption に関するログメッセージの説明

index	Name	Description
port	port	ポート番号
0	verify_state	検証プロセスの結果。 値は以下の通り。 0x1:プリエンプシオン無効 0x3:成功 0x4:失敗 0x5:検証プロセス無効
1	preempt_state	そのポートがプリエンプトされているかを示す
2	hold_req_state	hold_req が有効かどうかを示す
3	min_pfrm_adj	最小 mPacket 長
4	last_pfrm_adj	mPacket 長の調整値
5	TxFragCount	送信された mPacket の数
6	RxFragCount	受信された mPacket の数
7	RxSmdErrCount	SMD-Cx 付きのフレームを受信したが、 組み立てが進行中ではない数
8	RxAssyErrCount	プリエンプシオンフレームの組み立てに失敗した数
9	RxAssyOkCount	プリエンプシオンフレームの組み立てに成功した数
A	TxHoldReqCount	hold_req が FALSE から TRUE になった数
B	RxResponseOkCount	応答フレームを受信した数
C	RxResponseBadCount	エラー応答フレームを受信した数
D	RxVerifyOkCount	検証フレームを受信した数
E	RxVerifyBadCount	エラー検証フレームを受信した数
F	TxResponseOkCount	応答フレームを送信した数
G	TxVerifyOkCount	検証フレームを送信した数

Preemption が正常に動作している場合、各デバイスのログにて以下の動作を確認することができます：

表 5-17 Preemption 動作確認内容

関連動作	device	ログ内容	詳細
検証プロセスの成功	timeReceiver (client)	[G] がインクリメント	検証フレームを送信している
		[B] がインクリメント	自身が送信した検証フレームに対する timeTransmitter (client)からの応答フレームを受信している
	timeTransmitter (server)	[0] が 0x3	検証プロセスステータスが"成功"を示している
		[D] がインクリメント	timeReceiver (client)から送信された検証フレームを受信している
		[F] がインクリメント	timeReceiver (client)から送信された検証フレームに対する応答フレームを送信している
Preemption の発生	timeReceiver (client)	[5] がインクリメント	Preemption の発生によってパケットが分割されている
	timeTransmitter (server)	[6] がインクリメント	Preemption によって分割されたパケットが受信されている
		[9] がインクリメント	受信した分割パケットの再構成に成功している

5.2.1.4 PSFP ログメッセージ

PSFP に関するログメッセージは以下の通りです。

Qci Stream Match Packet Count	[port:0 > sid:0] : 25000
Qci MSDU Passed Packet Count	[port:0 > sid:0] : 5000
Qci MSDU Reject Packet Count	[port:0 > sid:0] : 0
Qci Gate g Passed Packet Count	[port:0 > gaid:0] : 5000
Qci Gate g Dropped Packet Count	[port:0 > gaid:0] : 20000
Qci Meter Green Packet Count	[port:0 > meid:0] : 5000
Qci Meter Red Packet Count	[port:0 > meid:0] : 0
VLAN Ingress Check Drop Frame Counter	[port:0] : 0
DST Address Lookup Hit Counter	[port:0] : 0
Error Interrupt - StatusQci Stream Filter	[port:0] : 0x00
Error Interrupt - StatusQci Gate	[port:0] : 0x00
Error Interrupt - StatusQci Meter	[port:0] : 0x00
Frame Parser Runtime Error Status	[port:0] : 0x00

各項目は以下の情報を示しています。

表 5-18 PSFP に関するログメッセージの説明

項目	説明
Qci Stream Match Packet Count	設定されたストリームフィルタに一致するフレームのカウン ト
Qci MSDU Passed Packet Count	最大 SDU サイズフィルタを通過したフレームのカウン ト
Qci MSDU Reject Packet Count	最大 SDU サイズフィルタで破棄されたフレームのカウン ト
Qci Gate g Passed Packet Count	ストリームゲートフィルタを通過したフレームのカウン ト
Qci Gate g Dropped Packet Count	ストリームゲートフィルタで破棄されたフレームのカウン ト
Qci Meter Green Packet Count	フローメーターを通過したフレームのカウン ト
Qci Meter Red Packet Count	フローメーターで破棄されたフレームのカウン ト
VLAN Ingress Check Drop Frame Counter	VLAN Ingress 検証で破棄されたフレームのカウン ト
DST Address Lookup Hit Counter	アクティブストリーム識別でヒットしたフレームのカウン ト
Error Interrupt - StatusQci Stream Filter	ストリームフィルタのエラー割り込みのステータス
Error Interrupt - StatusQci Gate	ストリームゲートフィルタのエラー割り込みのステータス
Error Interrupt - StatusQci Meter	フローメーターのエラー割り込みのステータス
Frame Parser Runtime Error Status	フレームパーサーエラー割り込みのステータス

5.2.1.5 MIB 情報(LLDP)

LLDP 起動時、以下のメニューが表示されます。

```
[main menu]
[0]: Info gPTP
[1]: MIB console
>>
```

このメニューで 0 を入力すると以下の表示が出力され、その後 gPTP 動作状況に関するログが表示されます。

ログ出力内容については"5.2.1.1 gPTP ログメッセージ"を参照してください。

```
[main menu]
[q]: quit
>>
```

この状態で q を入力すると LLDP 起動時のメニューに戻ります。

LLDP 起動時のメニューで 1 を入力すると以下のように表示され、MIB 情報を閲覧するためのコンソールが起動します。

【注】 コンソールでは、各コマンドの入力完了時に"Ctrl + Enter" ("CRLF")を入力する必要があります。

```
Started MIB Console.
>>
```

この状態で"snmp getnext 1"のコマンドを入力すると、ターゲットデバイスのすべての MIB 情報が取得できます。

この状態で"q"のコマンドを入力すると、LLDP 起動時のメニューに戻ります。

5.2.2 ピン出力内容

本アプリケーションは、GPIO ピンを用いて gPTP 動作確認用クロック波形の出力と、TSN 動作状態確認用の出力を行います。

5.2.2.1 gPTP 動作確認

本アプリケーションは、gPTP による時刻同期動作確認用に PTPOUT ピンへクロック波形を出力しています。

この信号はデバイスのもつ内部タイマによって生成されるクロックパルスです。

gPTP によってこの内部タイマが同期することで、デバイスごとの信号の位相が一致するよう調整されます。

これにより、これら信号の遅延をもとに時刻同期の精度を測定することが可能です。

【注】 クロック波形の周期は、"1[s]"(初期設定)と"1[ms]"から選択することが可能です。
変更する場合は"4.1.2.9 パルスジェネレータ周期の切り替え方法"を参照してください。

各ボードにおける各種ピンの割り当ておよびボード上の配置を以下に示します。

表 5-19 PTPOUT の割り当て

ボード	割り当て
RZ/T2M-RSK	CN2-8
RZ/N2L-RSK	CN1-11
RZ/T2N-Evb	J55-8





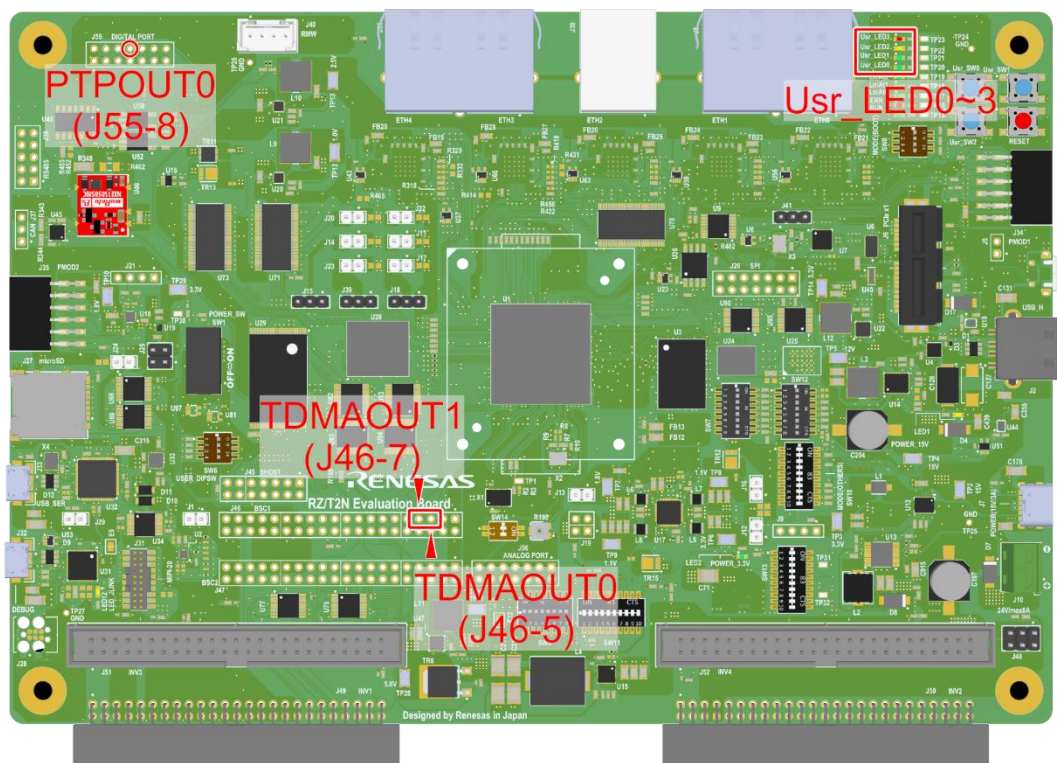


図 5-9 RZ/T2N-Evb ボード図(TDMA)

gPTP 動作時におけるピンの動作例を以下に示します。

EDUX1052A, CN63430212: Tue Jun 30 17:36:00 2026

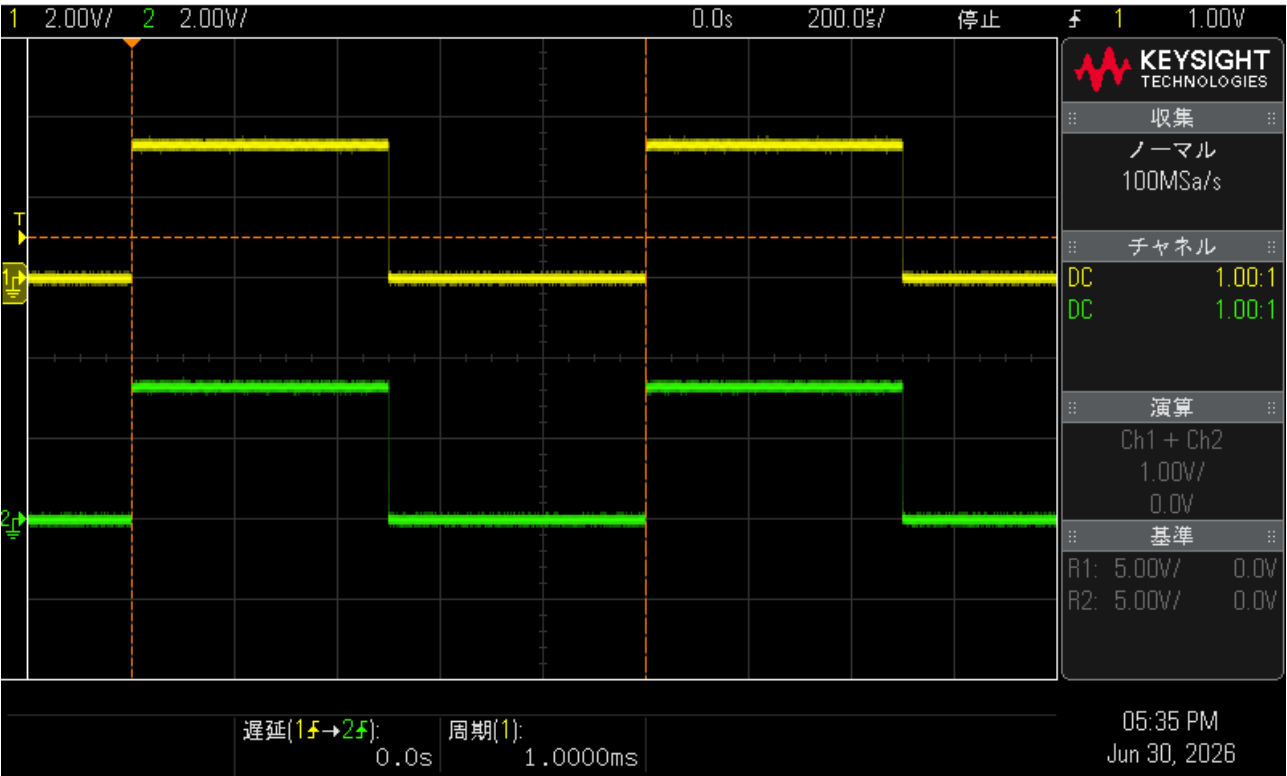


図 5-10 gPTP 動作時の PTPOUT

【注】 画像は RZ/T2M 使用時のものです。

【注】 画像は 1[ms]動作時のものです。

上図における各ピンの内容は以下の通りです。

表 5-20 各チャンネルの内容

チャンネル	色	出力ピン	説明
CH1	黄	PTPOUT0 (timeTransmitter)	現在のクロックが波形で出力される。
CH2	緑	PTPOUT0 (timeReceiver)	同上

5.2.2.2 TDMA 動作確認

本アプリケーションは、TSN における各動作に応じて TDMAOUT ピンと LED ピンの出力を変化させます。

よって、オシロスコープを用いて TSN 動作の様子を確認することができます。

出力内容は以下の通りです。

表 5-21 TDMA 出力ピン

ピン	説明	詳細
TDMAOUT0	現在の TDMA スロットの状態に応じて切り替わる	Slot0 の時のみ "1"
TDMAOUT1	同上	Slot0, Slot2 の 時のみ "1"
LED0,1,2,3 (*1)	PROFINET フレームを受信するたびにトグル動作	フレーム ID に対応した LED が点滅

【注】 *1: timeTransmitter のみ

各ボードにおける各種ピンの割り当ておよびボード上の配置を以下に示します。

表 5-22 TDMA 動作確認用ピンの割り当て

ボ ー ド	TDMAOUT0	TDMAOUT1	LED0	LED1	LED2	LED3
RZ/T2M-RSK	CN1-4	ETH_LED0	LED0	LED1	LED2	LED3
RZ/N2L-RSK						
RZ/T2N-EVb	J46-5	J46-7	Usr_LED0 (LED8)	Usr_LED1 (LED9)	Usr_LED2 (LED10)	Usr_LED3 (LED11)



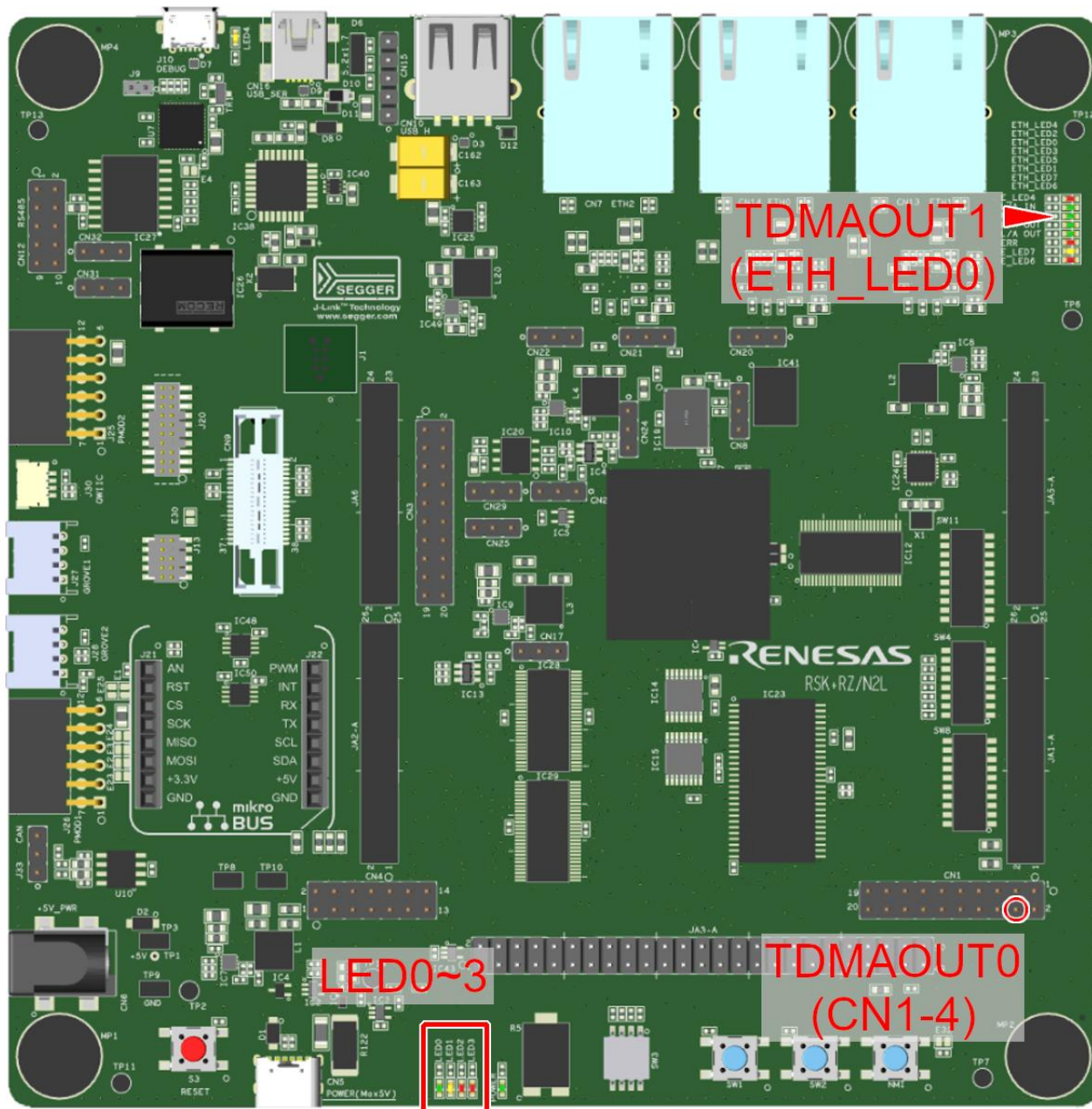


図 5-12 RZ/N2L-RSK ボード図(TDMA)



TDMA 動作時における各ピンの動作例を以下に示します。

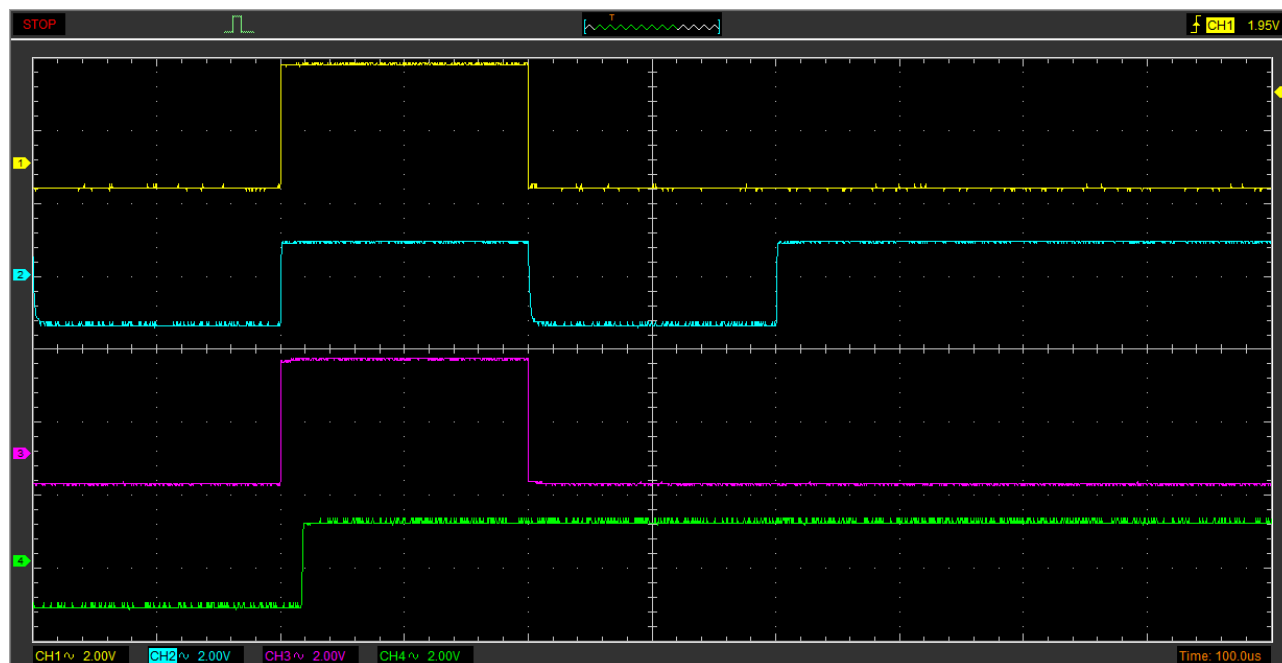


図 5-14 TDMA 動作時のピン動作

上図における各ピンの内容は以下の通りです。

表 5-23 各チャンネルの内容

チャンネル	色	出力ピン	説明
CH1	黄	TDMAOUT0 (timeTransmitter)	現在の TDMA スロットの状態に応じて切り替わる
CH2	青	TDMAOUT1 (timeTransmitter)	同上
CH3	赤	TDMAOUT0(timeReceiver)	同上
CH4	緑	LED0(timeTransmitter)	テストフレーム受信タイミングでトグル動作

上図に示したような各ピンの波形により、以下のことが確認できます。

- CH1 (timeTransmitter の TDMAOUT0)と CH2 (timeTransmitter の TDMAOUT1)により、TDMA スロットが指定したタイミングで切り替わっていることが確認できます。
- CH1 (timeTransmitter の TDMAOUT0)と CH3 (timeReceiver の TDMAOUT0)により、timeTransmitter と timeReceiver が正しく時刻同期していることが確認できます。
- CH1 (timeTransmitter の TDMAOUT0)と CH4 (timeTransmitter の LED0)により、適切な TDMA スロットのタイミングでテストフレームが受信されていることが確認できます。

送受信されるパケットの種類および送信タイミングに関しては、"5.1 アプリケーション動作概要"を参照して下さい。

5.2.2.3 CQF 動作確認

本アプリケーションは、CQF における各動作に応じて TDMAOUT ピンと LED ピンの出力を変化させます。

よって、オシロスコープを用いて CQF 動作の様子を確認することができます。

出力内容は以下の通りです。

表 5-24 CQF 動作時の TDMA 出力ピン

ピン	説明	詳細
TDMAOUT0	現在の TDMA スロットの状態に応じて切り替わる	CQF 周期が ODD のとき "1" (Slot2, 3, 6, 7 の時のみ "1")
TDMAOUT1	同上	Slot0, 1, 2, 3 の時のみ "1"
LED0,1,2,3 (*1)	PROFINET フレームを受信するたびにトグル動作	フレーム ID に対応した LED が点滅

【注】 *1: timeTransmitter のみ

- 各ボードにおける各種ピンの割り当てについては、"表 5-22 TDMA 動作確認用ピンの割り当て"を参照してください。

2 台構成での CQF 動作時における各ピンの動作例を以下に示します。

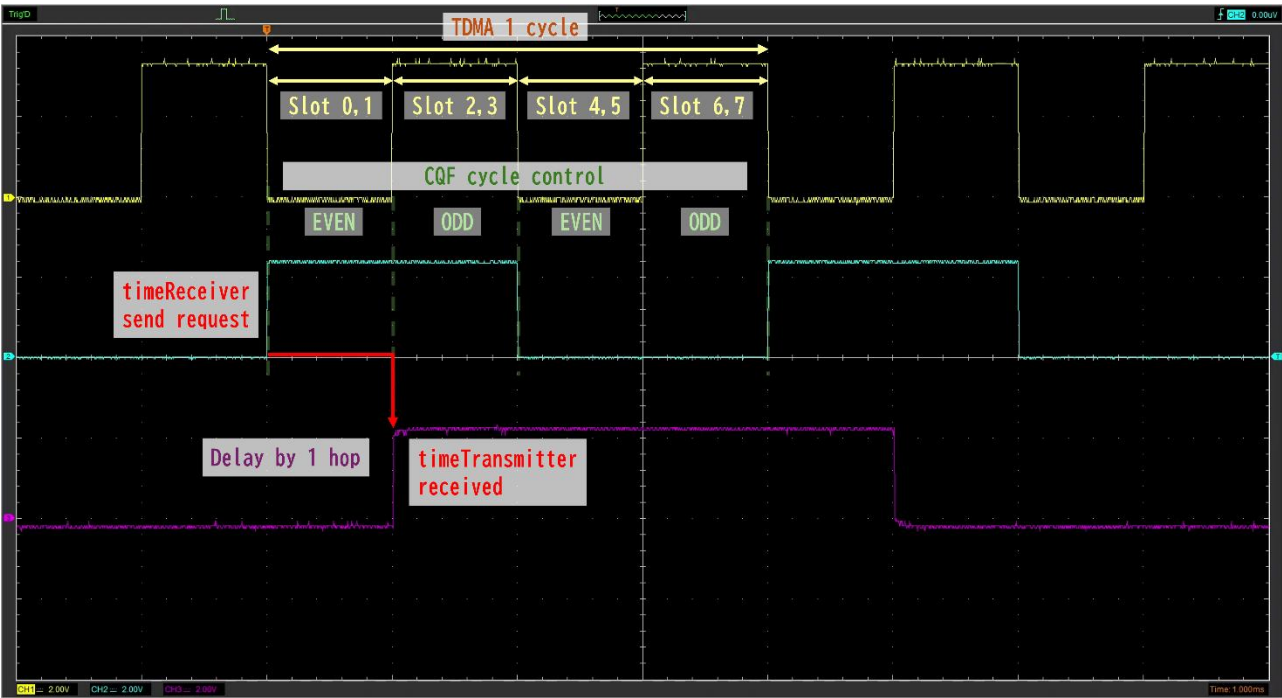


図 5-15 CQF 動作時のピン動作(2 台構成)

上図における各ピンの内容は以下の通りです。

表 5-25 各チャンネルの内容

チャンネル	色	出力ピン	説明
CH1	黄	TDMAOUT0 (timeReceiver)	現在の TDMA スロットの状態に応じて切り替わる
CH2	青	TDMAOUT1 (timeReceiver)	同上
CH3	赤	LED0 (timeTransmitter)	profinet フレーム(フレーム ID: 0x101)受信タイミングでトグル動作

上図に示したような各ピンの波形により、以下のことが確認できます。

- CH1(timeReceiver の TDMAOUT0)と CH2(timeReceiver の TDMAOUT1)、CH3(timeTransmitter の LED0)により、フレーム ID が 0x102 の profinet フレームが送信されるタイミングでフレーム ID が 0x101 の profinet フレームを受信していることが確認できます。
(CQF により、受信タイミングがホップ数 1 の分 1 周期遅延している。)

"4.4 サンプルアプリケーションの動作確認の例(Qch 3 台構成)"に示すような 3 台構成での CQF 動作時における各ピンの動作例を以下に示します。

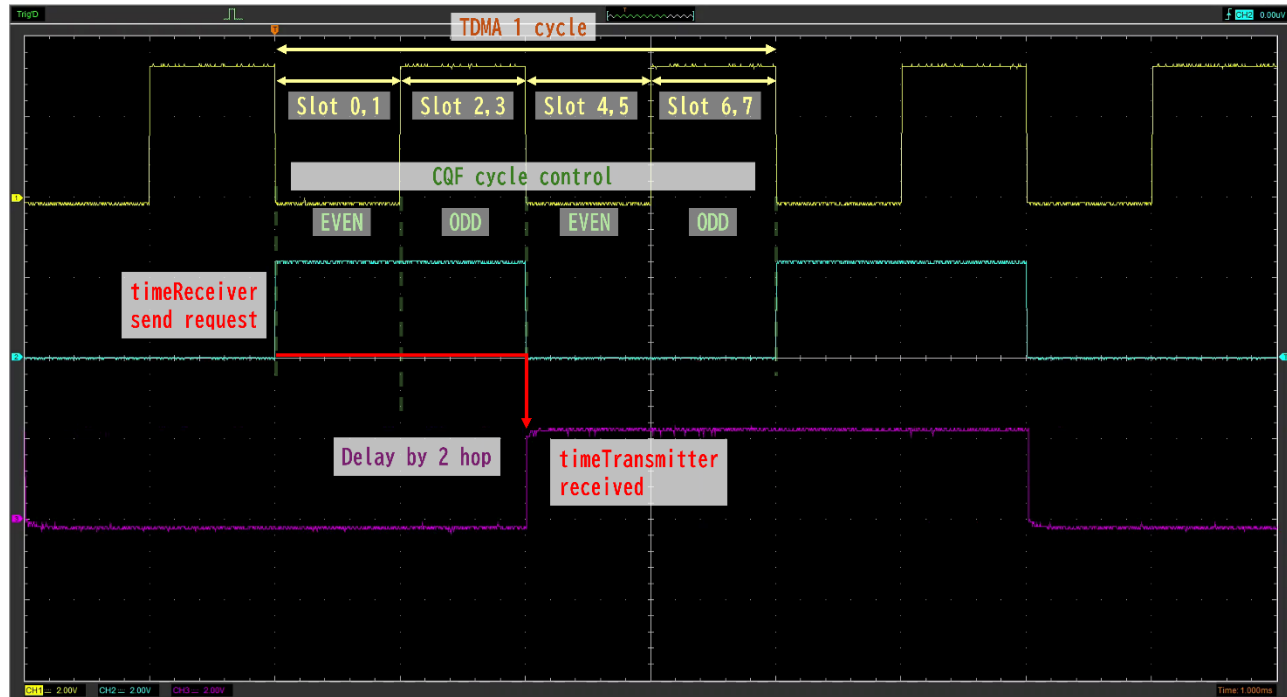


図 5-16 CQF 動作時のピン動作(3 台構成)

上図における各ピンの内容は以下の通りです。

表 5-26 各チャンネルの内容

チャンネル	色	出力ピン	説明
CH1	黄	TDMAOUT0 (timeReceiver2)	現在の TDMA スロットの状態に応じて切り替わる
CH2	青	TDMAOUT1 (timeReceiver2)	同上
CH3	赤	LED0 (timeTransmitter)	profinet フレーム(フレーム ID: 0x101)受信タイミングでトグル動作

上図に示したような各ピンの波形により、以下のことが確認できます。

- CH1(timeReceiver2 の TDMAOUT0)と CH2(timeReceiver2 の TDMAOUT1)、CH3(timeTransmitter の LED0)により、フレーム ID が 0x103 の profinet フレームが送信されるタイミングでフレーム ID が 0x101 の profinet フレームを受信していることが確認できます。
(CQF により、受信タイミングがホップ数 2 の分 2 周期遅延している。)

5.3 アプリケーション構成

5.3.1 モジュール構成

5.3.1.1 全体説明

サンプルアプリケーションの環境では、「TCP/IP」スタックと「GMAC」ドライバの間に「NETIF」モジュールが存在しています。

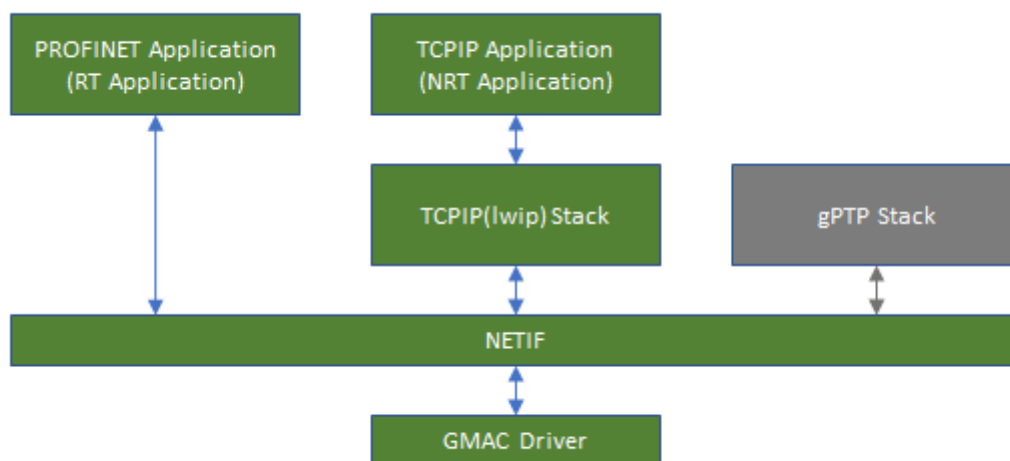


図 5-17 ソフトウェアモジュール構成

5.3.1.1.1 注意事項 – 開発環境の違いによる差異について

上記図の「NETIF」モジュールは、e2studio の場合「NETWORK」モジュールとなります。

「NETIF」モジュールと「NETWORK」モジュールとでは構成が異なります。

5.3.1.2 詳細説明

5.3.1.2.1 フレーム受信時

「NETIF」モジュールの「Ethernet Reader Task」がフレームの受信を行います。

受信後、「Ethernet Reader Task」が各 RxQueue への振り分けを行います。

各 RxQueue 内のフレームを、Queue に対応したアプリケーション(プロトコル)が取得します。

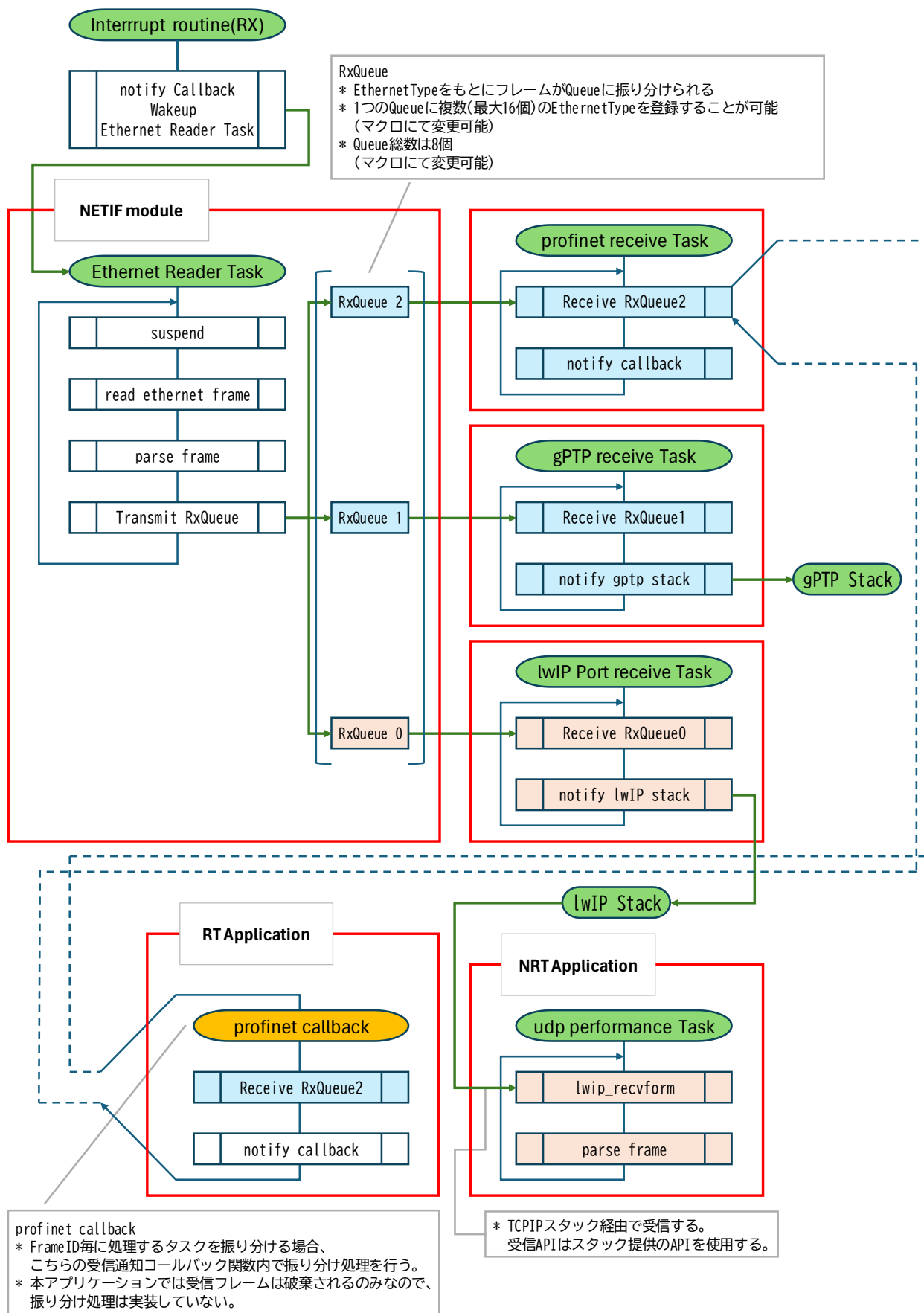


図 5-18 ソフトウェアモジュール動作フロー(フレーム受信関連)

5.3.1.2.2 フレーム送信時

「NETIF」モジュールの「Ethernet Writer Task」がフレームの送信を行います。

「Ethernet Writer Task」は優先度の高い TxQueue のフレームを優先して送信します。

各 Queue への振り分けには「QueueID」が使用されます。

「QueueID」は送信フレームのもつ管理タグ内で設定されています。

各アプリケーション(プロトコル)は、フレーム送信時に TxQueue を意識する必要はありません。

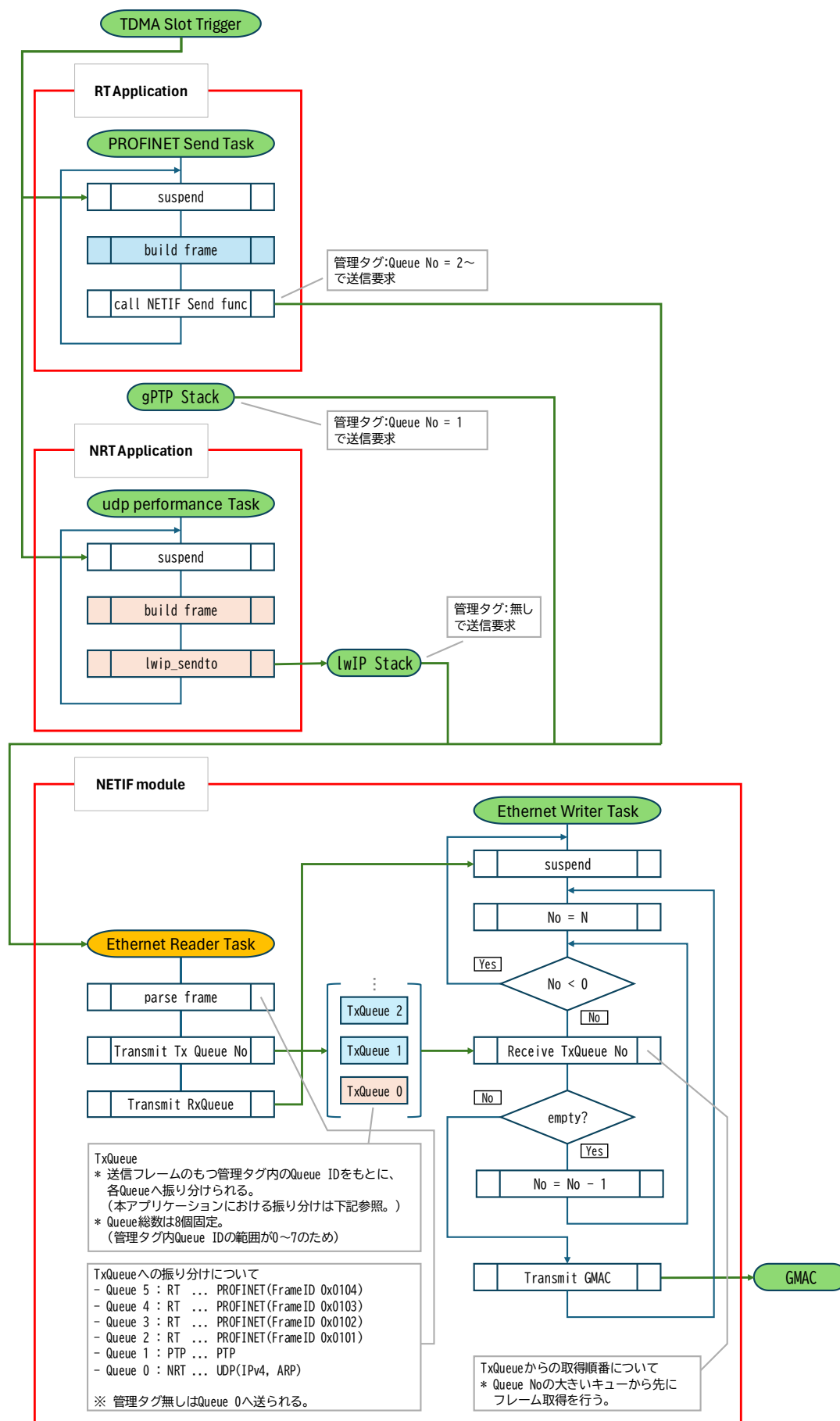


図 5-19 ソフトウェアモジュール動作フロー(フレーム送信関連)

5.3.2 注意事項 – サンプルアプリケーションのタスク優先度

タスクの優先度は以下の通りとしてください。

表 5-27 タスク優先度

優先度	タスク	注釈
上位	● LLDP タスク	受信タスク・LINKED イベント通知タスクは、これらより低い優先度となります(下記参照)。 gPTP タスクの処理時間を確保する必要があります。 SYNCED イベント通知タスク "EventHandler" のみ、これらより低い優先度となります(下記参照)。
	● シリアル入力タスク (メニュー操作用)	
	● NETIF 送信タスク	
	● RT 通信タスク全般	
	● gPTP タスク全般	
下位	● NETIF 受信タスク	
	● LINKED イベント通知タスク (NETIF イベント通知タスク) "Ethernet Monitor Task"	
	● SYNCED イベント通知タスク (gPTP イベント通知タスク) "EventHandler"	
	● NRT 通信タスク	
	● シリアル出力タスク (ログ出力用)	

【注】 タスク優先度を変更する場合、上記順位付けに準拠するように設定してください。
上記を満たさない場合の動作は保証されません。

5.3.2.1 備考 – タスク優先度の変更による configuration.xml の優先度設定

RTOS のタイマタスクやメインタスクなどの基本的なタスクは、configuration.xml にて優先度をはじめとした各種設定がなされています。

上記でタスク優先度を変更する場合は、これらタスクの優先度も考慮する必要があります。

configuration.xml で設定するタスクの優先度設定は以下の通りとしてください。

表 5-28 configuration.xml の優先度設定

タスク	優先度	Default
Main Thread Priority ("main_thread")	最上位	28
Timer Task Priority ("Tmr Svc")	Main Thread Priority より下、 ただし他タスクより上	27

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2025/12/19	-	新規作成
1.01	2026/03/17		IEEE802.1Qbu(Frame Preemption)関連の記載を追加
1.02	2026/05/29		IEEE802.1Qci IEEE802.1Qch IEEE802.1AB 関連の記載を追加
1.03	2026/09/24		RZ/T2N-Evb 追加 上記に伴うマルチプロセス関連の追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違くと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
 2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
 3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
 4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
 5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
 6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
 標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
 高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
 当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
 7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因したまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
 8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
 9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
 10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
 11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
 12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものとしたします。
 13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
 14. 本資料に記載されている内容または当社製品についてご不明点がございましたら、当社の営業担当者までお問合せください。
- 注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。
- 注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。