R-IN32M3 Module (RY9012A0)
ユーザ実装ガイド (uGOAL 版)

R30AN0402JJ0103
Rev.1.03
2023.12.15

要旨

本書では、"R-IN32M3 Module (RY9012A0) サンプルパッケージ" に同梱される R-IN32M3 Module を制御するホストマイコン用ソフトウェアを各ユーザの開発する産業イーサネット対応製品に適用するための実装方法について説明します。ユーザアプリケーションの実装方法、ハードウェア周辺の移植方法、各産業イーサネットプロトコルのプロファイル設定やデバイス定義ファイルについて、記述例を示しながら説明します。

動作確認デバイス

R-IN32M3 Module (RY9012A0)

目次

用語解説

本書で使用する用語は、以下に示すように定義して使用します。

用語	説明
本サンプル	R-IN32M3 Module 向け産業ネットワークサンプルプログラムのうち、R-IN32M3 Module を制御するホストマイコン用サンプルプログラム
API	Application Programming Interface
GOAL / uGOAL	Generic Open Abstraction Layer 詳細は、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照
FSP	Flexible Software Package Renesas RA ファミリを用いた組み込みシステムを開発するためのソフトウェアパッケージです。詳細は、ルネサス Web FSP サイトを参照してください、

関連文書

資料名	資料番号
R-IN32M3 Module (RY9012A0) データシート	R19DS0109JJ****
R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ハードウェア編	R19UH0122JJ****
R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編	R17US0002JJ****
R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド	R30AN0390JJ****
RA サンプルアプリケーション (uGOAL版)	R30AN0398JJ ****

1. はじめに

本書では、R-IN32M3 Module を制御するホスト CPU 用のサンプルソフトウェアに対する実装ガイドについて説明します。

2 章では、サンプルソフトウェアの全体概要、並びに、ユーザアプリケーションの実装ガイドについて説明します。

3 章では、R-IN32M3 Module を使用するためのハードウェア設計ガイドを含む、ハードウェア周辺のドライバおよびソフトウェアの実装ガイドについて説明します。

4 章から 6 章では、どのプロトコルでも共通して設定するベンダ ID やデバイス ID(以降、Device Identity)、及び、プロトコルに応じて取り扱うデータやサイズ(以降、Data Model)について、プロトコルの単位に分けて説明します。その他、Master 機器と通信する際に用いられる下記デバイス定義ファイルの記述例についても説明します。

対応プロトコルとデバイス定義ファイル

- PROFINET: GSDML ファイル
- EtherNet/IP: EDS ファイル
- EtherCAT: ESI ファイル

以降の説明ではプログラムファイルやデバイス定義ファイルの例を記しています。それらには、R-IN32M3 Module を制御するホストマイコンの 1 つである RA のサンプルプログラムを用いて説明します。

2. サンプルソフトウェア概要

2.1 全体構成

ホスト CPU 用のサンプルソフトでは、uGOAL ミドルウェアを備えており、その設計思想に基づいた構成となっています。uGOAL システムの構成図を Figure 2-1 に示します。

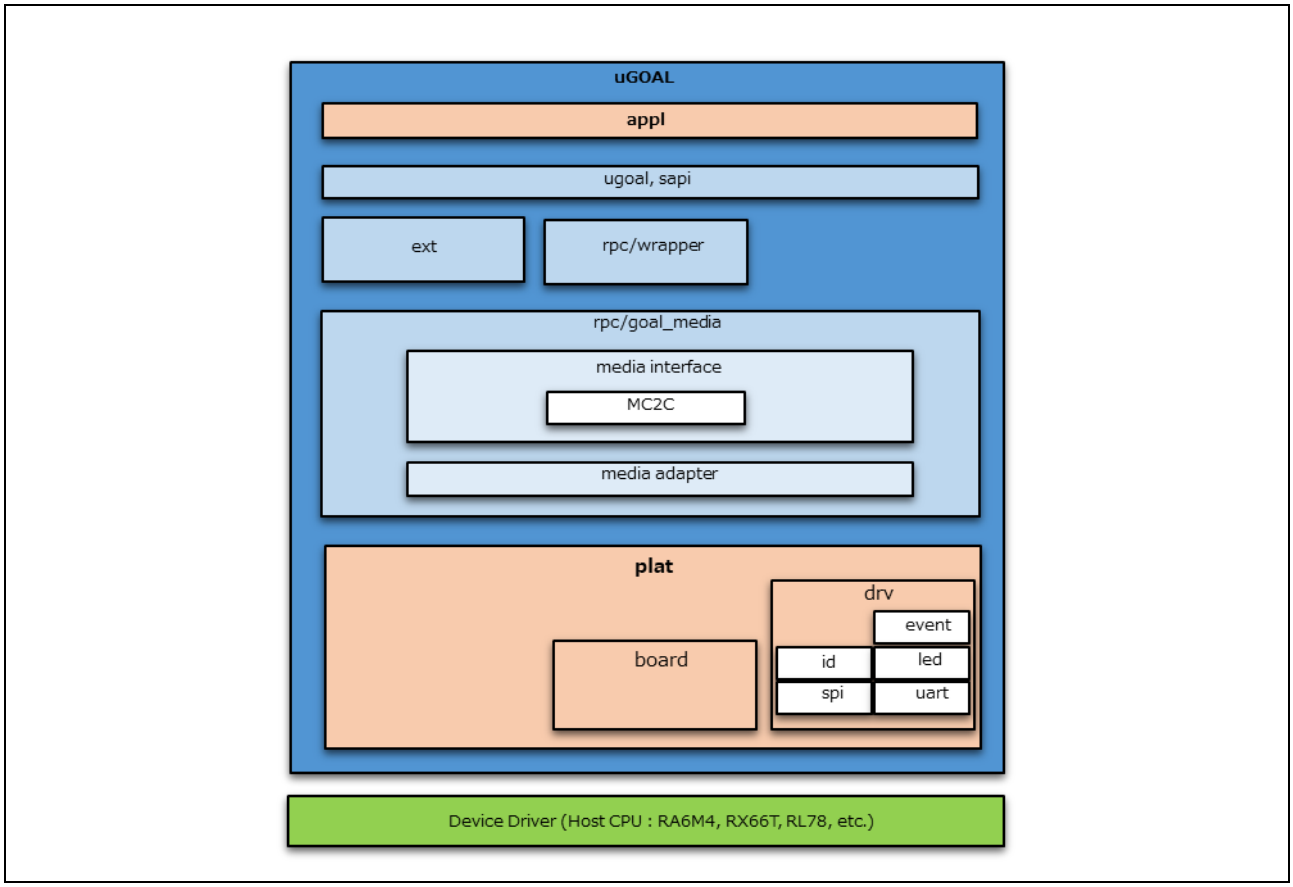

Figure 2-1 SW Configuration of uGOAL

uGOAL の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

2.1.1 フォルダ構成

サンプルソフトウェアのフォルダ構成を以下に示します。
構成は変更となる可能性があるため、各ホストマイコンに対応した最新のサンプルアプリケーションノートを参照ください。

RA6_uCCM_V***	
├──appl	ユーザアプリケーション
│ ├──01_pnio	PROFINET サンプルアプリケーション
│ ├──02_eip	EtherNet/IP サンプルアプリケーション
│ ├──03_ecat	EtherCAT サンプルアプリケーション
│ ├──04_pnio_largesize	PROFINET Large Size サンプルアプリケーション
│ ├──05_eip_largesize	EtherNet/IP Large Size サンプルアプリケーション
│ ├──06_ecat_largesize	EtherCAT Large Size サンプルアプリケーション
│ ├──07_modbus_tcp_slave	Modbus TCP サンプルアプリケーション
│ ├──10_multi_protocol	Multi [01_pnio, 02_eip, 03_ecat, 07_modbus] サンプルアプリケーション
│ ├──11_pnio_http	01_pnio サンプルに web サーバ機能，ホストマイコン FW 更新機能 拡張
│ ├──12_eip_http	02_eip に web サーバ機能，ホストマイコン FW 更新機能 拡張
│ ├──13_ecat_http	03_ecat に web サーバ機能，ホストマイコン FW 更新機能 拡張
│ └──17_fwup_bootloader	ホストマイコン FW 更新機能用 bootloader
│	
├──plat	デバイス依存コンポーネント(OS 依存部、ボード仕様、ドライバ群)
├──projects	ユーザアプリケーションと対になるプロジェクトファイル一式
│	
├──ugoal	uGOAL(Generic Open Abstraction Layer *)のメイン部
├──rpc	NW プロトコルや MCTC を含む RPC(Remote Procedure Call)に関連した機能部
├──sapi	Simple API
└──ext	外部ソフトウェアコンポーネント

* uGOAL の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

2.1.2 ユーザーコーディング部

サンプルソフトをユーザの開発プラットフォームへ移植するときに対応が必要な部分を示します。

Table 2-1 User cording part

appl	ユーザアプリケーション
plat	ボード依存部分と各機能のドライバ ボード依存部：ボードの初期化および設定 ドライバ：I2C, SPI, UART などデバイスドライバの初期化および設定

Table 2-2 Independent part (No need cording)

uGOAL middleware	
ugoal	uGOAL のメイン制御部
rpc	NW プロトコルや MCTC を含む RPC(Remote Procedure Call)に関連した機能部
sapi	Simple API
ext	外部ソフトウェアコンポーネント

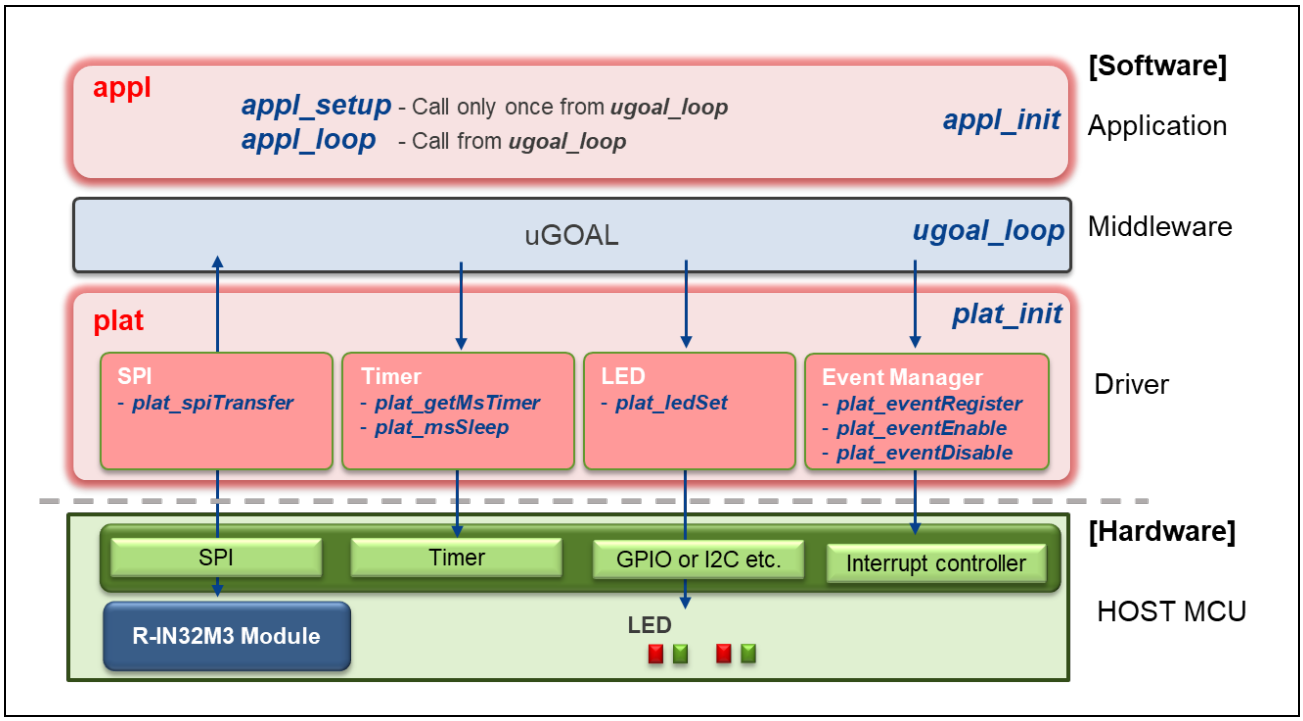

Figure 2-2 software structure

2.1.3 概略フロー

uGOAL 全体の概略フローは以下の通りです。main ルーチンにて、初期化処理として plat_init()関数、som_init()関数、appl_init()関数を呼び、ループ処理として ugoal_loop()関数を呼びます。

ユーザ側のプログラムに uGOAL システムを組み込む際は、初期化処理にこれらの 3 つの関数、ループ処理に ugoal_loop()関数を実装してください。

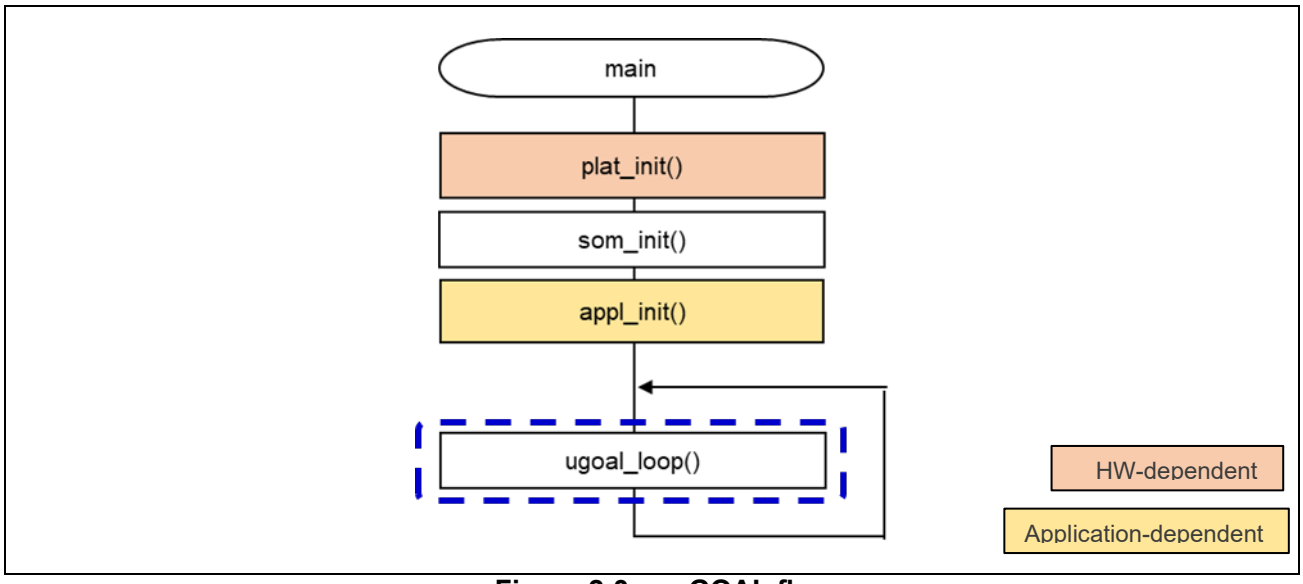

Figure 2-3 uGOAL flow

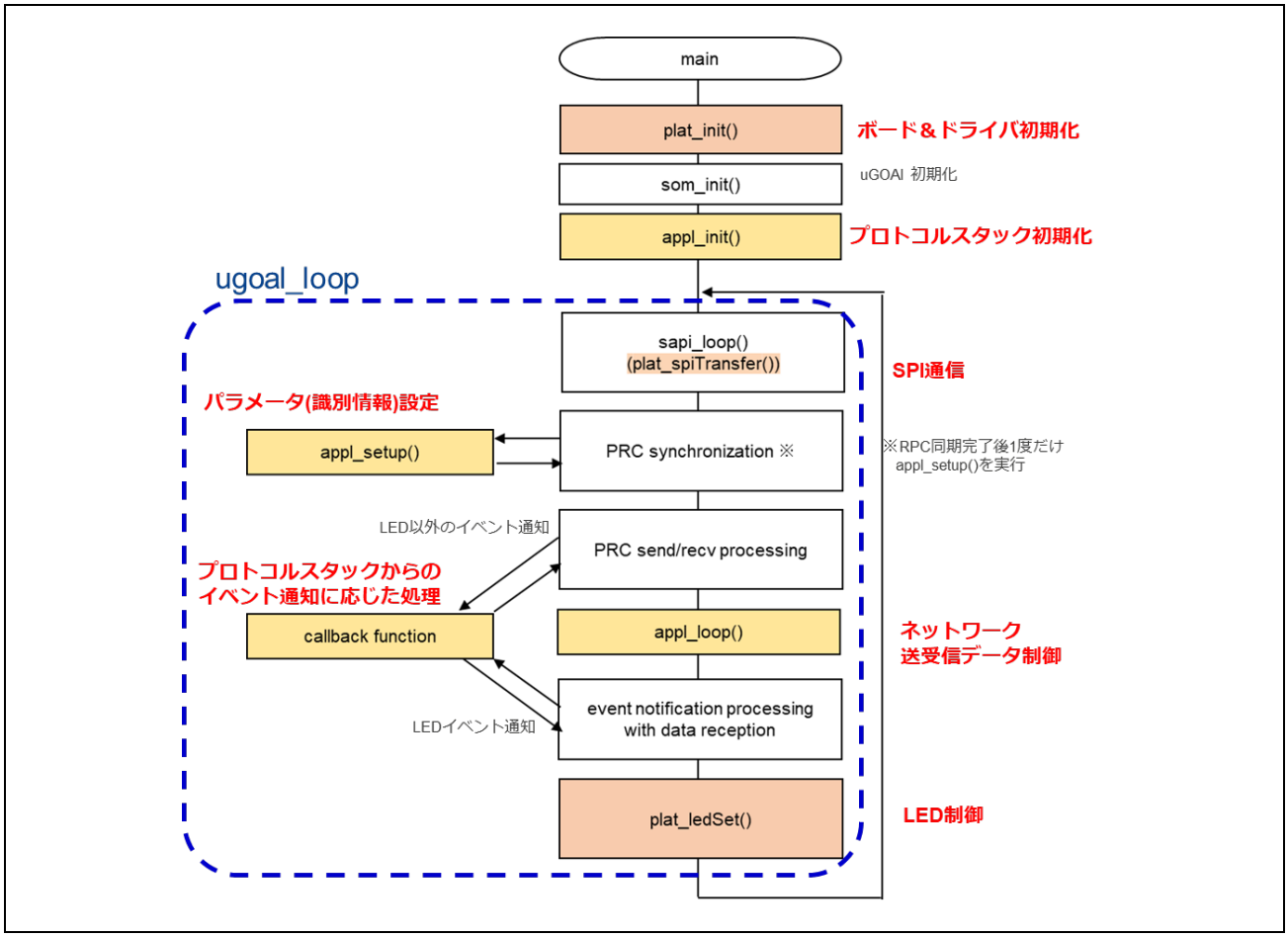

Figure 2-4 uGOAL flow (detailed)

Figure 2-4 に示すように、初期化処理とループ処理から呼ばれる主なユーザ実装部分は以下になります。

Table 2-3 Required user implementation part

part	User implementation	File	functions call
- appl	appl_init() [2.2.1、2.3.1、2.4.1 参照] - 使用するプロトコルスタック、uGOAL の初期化処理	goal_appl.c	Initialization processing
	appl_setup() [2.2.2、2.3.2、2.4.2 参照] - 使用するプロトコルのプロファイル(ベンダ ID 等)、Input/Output データの設定		ugoal_loop()
	appl_loop() [2.2.3、2.3.3、2.4.3 参照] - uGOAL ループ周期(約 1ms)で呼ばれる、ループ処理		
	callback function [2.2.3、2.3.3、2.4.3 参照] - プロトコル毎のイベントに応じた制御		
plat	plat_init() [3.1(a)参照] - ハード依存部分の初期化処理使用する各ドライバをオープン(利用可能)にする初期化処理	ra6m4ek \ plat.c	Initialization processing
	plat_spiTransfer() [3.2.3.2(1)(a)参照] - uGOAL ループ周期(約 1ms)で呼ばれる SPI 通信	ra6m4ek \ plat.c	ugoal_loop()
	plat_ledSet() [3.3.1.1(2)(a)参照] - uGOAL ループ周期(約 1ms)で呼ばれる I2C を介した LED 更新		

uGOAL にはユーザアプリケーション用関数として appl_init()、appl_setup()、appl_loop()、プロトコル固有のコールバックの関数が用意されています。
Figure 2-4 に示す通り、uGOAL の初期化処理では appl_init()が実行され、ループ処理である ugoal_loop()では appl_setup()が一度実行された後、appl_loop()が周期的に実行され、プロトコル毎のイベントに応じてコールバック関数が実行される構成となっています。

以下に、産業 Ethernet プロトコル毎のユーザアプリケーションについて説明します。

2.2 PROFINET ユーザアプリケーション

PROFINET サンプルアプリケーションについて説明します。

対象サンプルを Table 2-4 に示します。

Table 2-4 PROFINET Sample software

Sample software	Overview
01_pnio	サイクリック通信 サンプル
04_pnio_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル
11_pnio_http	01_pnio に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

2.2.1 appl_init() - 初期化

PROFINET プロトコルスタックの起動、及び、使用する uGOAL の各モジュールの初期化を行います。ユーザ側で処理を追加する必要はありません。

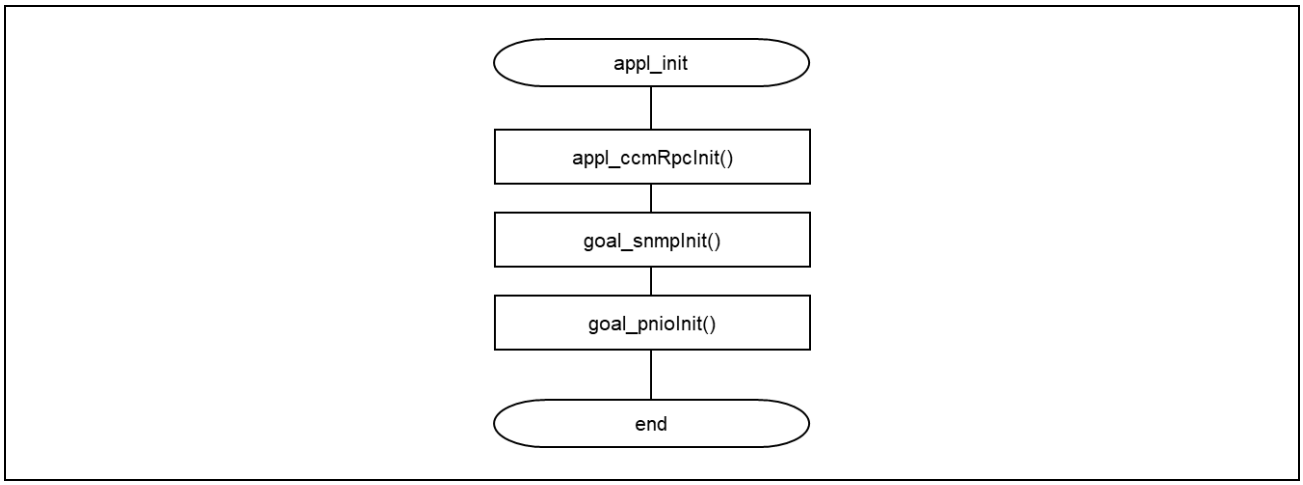

Figure 2-5 PROFINET appl_init() flow

Table 2-5 appl_init

Function	Overview
appl_ccmRpcInit()	uGOAL RPC interface の初期化
goal_snmplint()	SNMP function の初期化
goal_pnioInit()	PROFINET protocol の初期化

2.2.2 appl_setup() - パラメータ設定

LED の初期化、プロトコルのプロファイル設定(ベンダ ID 設定等)、及び、Input/Output データ設定を行います。そのため、4.1、4.2 章を参考に、ユーザ仕様に応じてこれらの設定を行ってください。

また、goal_pnioNew()関数にコールバック関数(appl_pnioCb())を登録することで、プロトコルスタックから通知されるイベントに対する準備を行います。

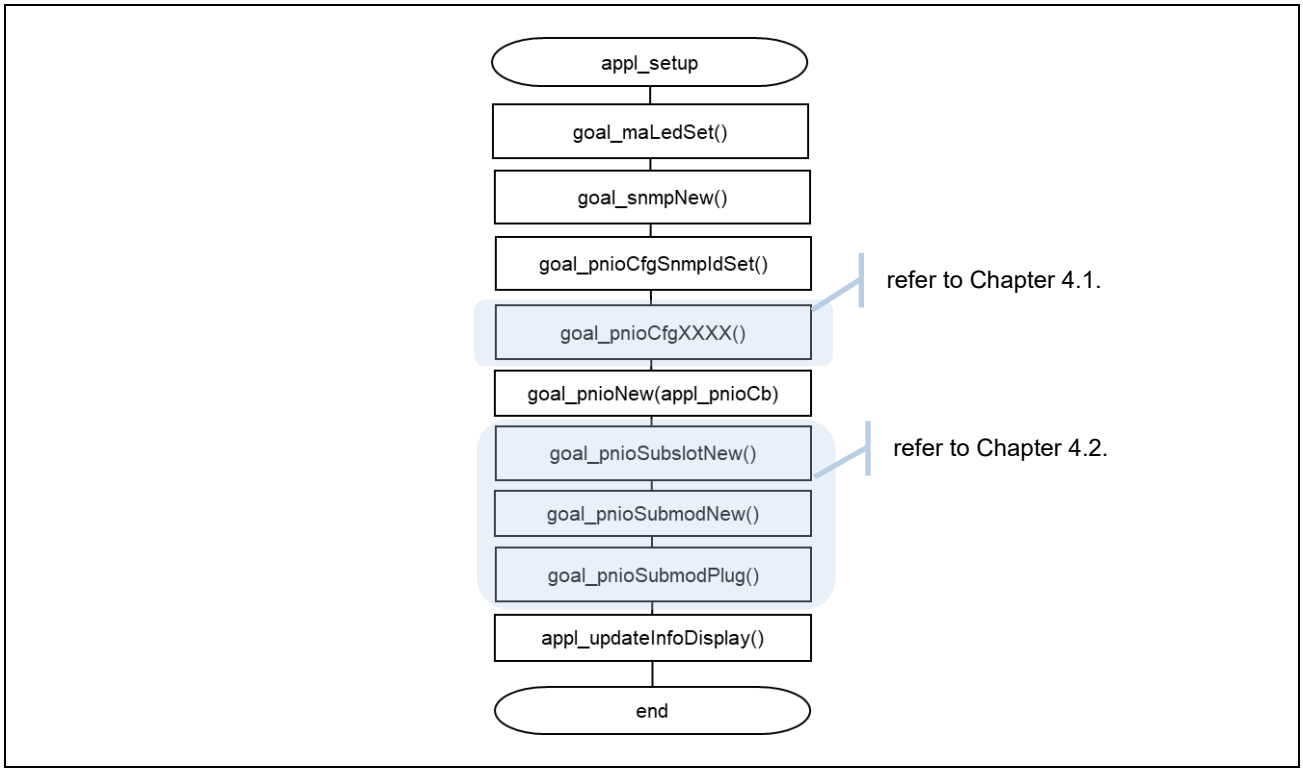

Figure 2-6 PROFINET appl_setup() flow

2.2.3 appl_loop() - データ送受信

appl_loop()でユーザアプリケーションを実装します。4.3 章を参考にデータ送受信を構成してください。なお、データ送受信の更新周期は 1ms で行っています*。

[* 注意] 1ms 周期は 01_pnio サンプルでの設定です。

更新周期は、データ転送方式を考慮する必要があります。参照：C-データサイズ上限

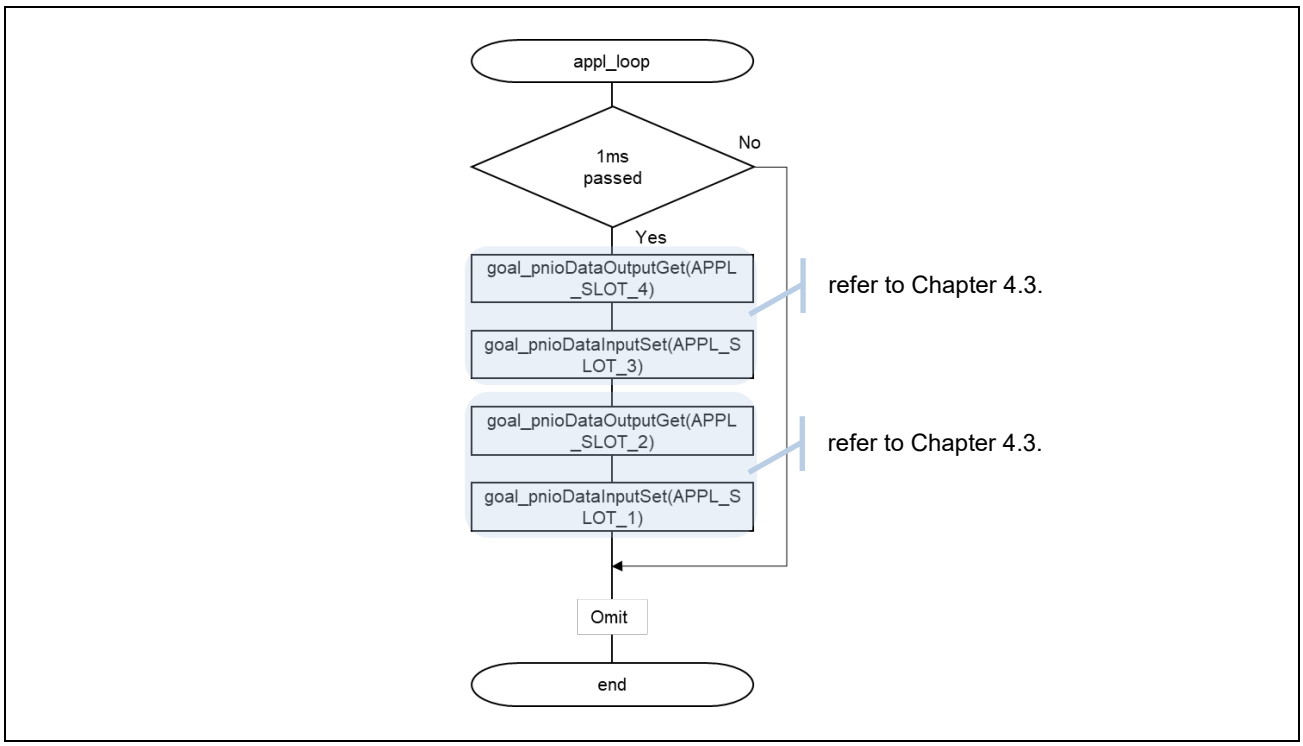

Figure 2-7 PROFINET appl_loop() flow

2.2.4 callback

コールバック関数(appl_pnioCb())では、プロトコルスタックから報告される状態に応じた処理を行います。ユーザ仕様に応じて処理を実装してください。

なお、プロトコルスタックから報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

この状態に応じた処理の 1 つとして LED ステータス(3.3 章参照)を成形しています。この成形したステータスは plat_ledSet()の引数に用いられ、plat_ledSet()が呼ばれるタイミングで各 LED の状態を更新します。

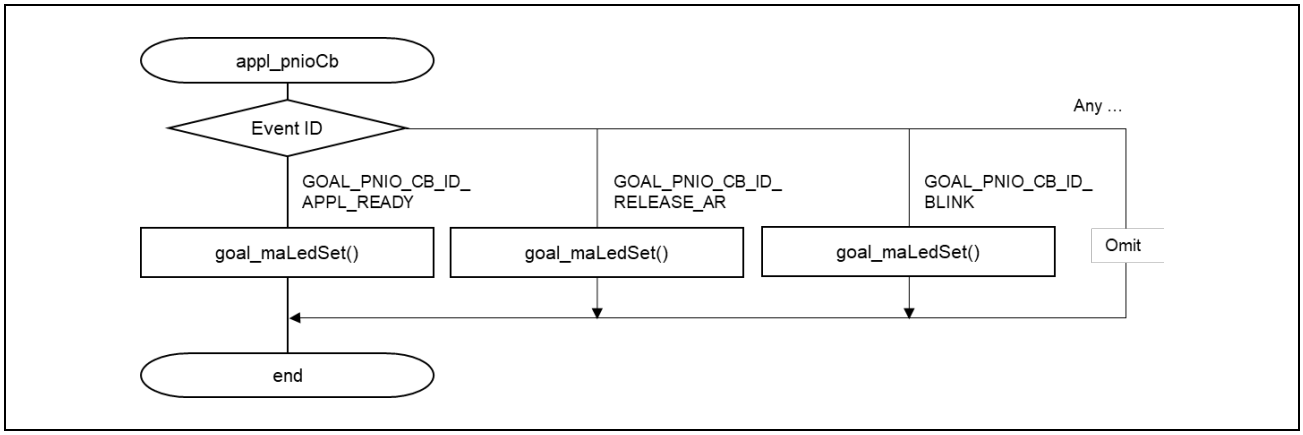

Figure 2-8 PROFINET appl_pnioCb() flow

2.3 EtherNet/IP ユーザアプリケーション

EtherNet/IP サンプルアプリケーションについて説明します。

対象サンプルを Table 2-6 に示します。

Table 2-6 EtherNet/IP Sample software

Sample software	Overview
02_eip	サイクリック通信 サンプル
05_eip_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル
12_eip_http	02_eip に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

2.3.1 appl_init() - 初期化

EtherNet/IP プロトコルスタックの起動、及び、使用する uGOAL の各モジュールの初期化を行います。ユーザ側で処理を追加する必要はありません。

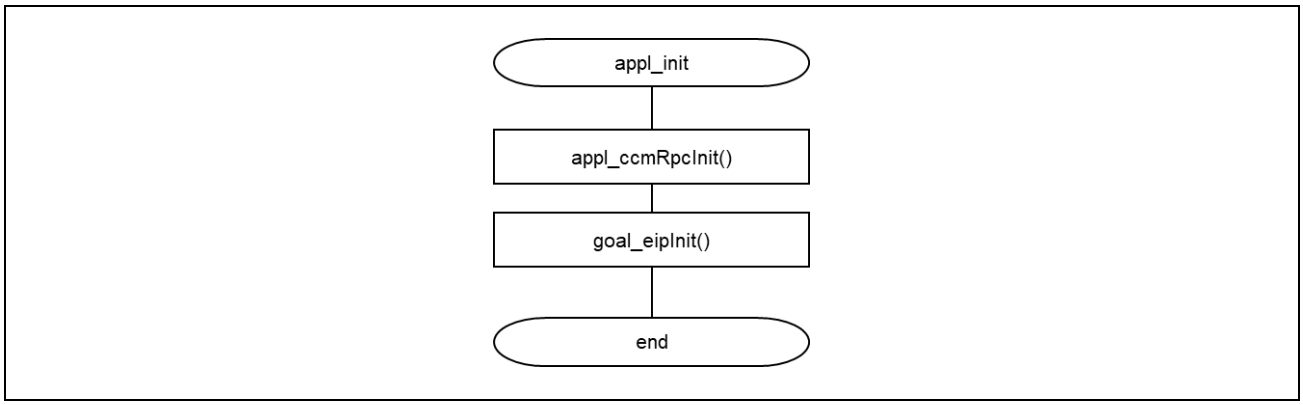

Figure 2-9 EtherNet/IP appl_init() flow

2.3.2 appl_setup() - パラメータ設定

LED の初期化、プロトコルのプロファイル設定(ベンダ ID 設定等)、及び、Input/Output データ設定を行います。そのため、0、5.2 章を参考に、ユーザ仕様に応じてこれらの設定を行ってください。

また、goal_eipNew()関数にコールバック関数(main_eipCallback())を登録することで、プロトコルスタックから通知されるイベントに対する準備を行います。

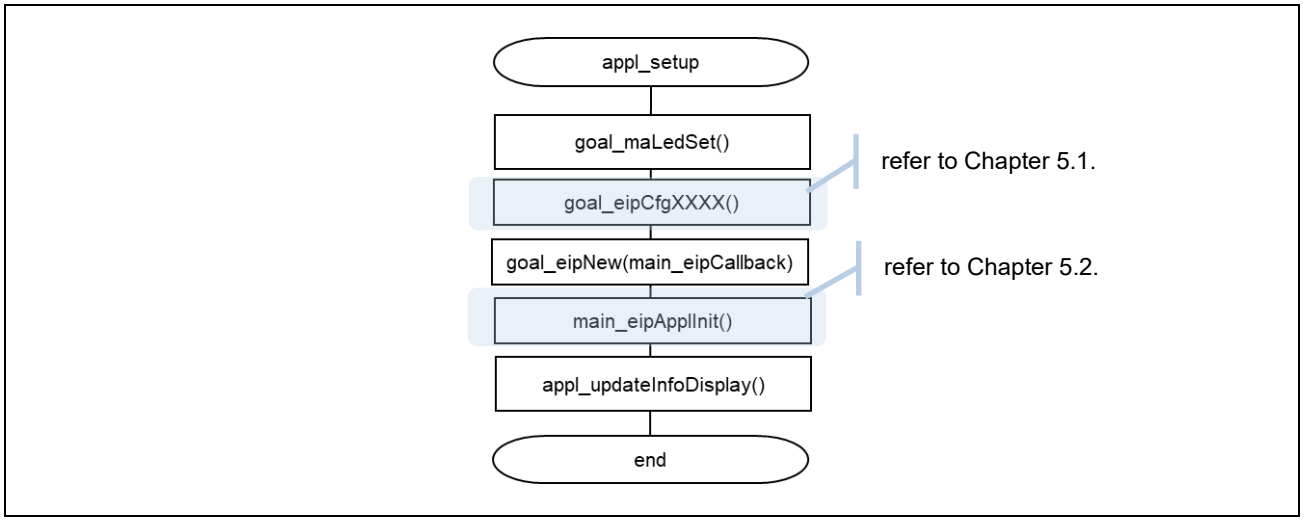

Figure 2-10 EtherNet/IP appl_setup() flow

2.3.3 appl_loop() - データ送受信

appl_loop()でユーザアプリケーションを実装します。5.3 章を参考にデータ送受信を構成してください。なお、データ送受信の更新周期は 1ms で行っています*。

[* 注意] 1ms 周期は 01_pnio サンプルでの設定です。

扱うデータサイズにより更新周期を考慮する必要があります。参照：C-データサイズ上限

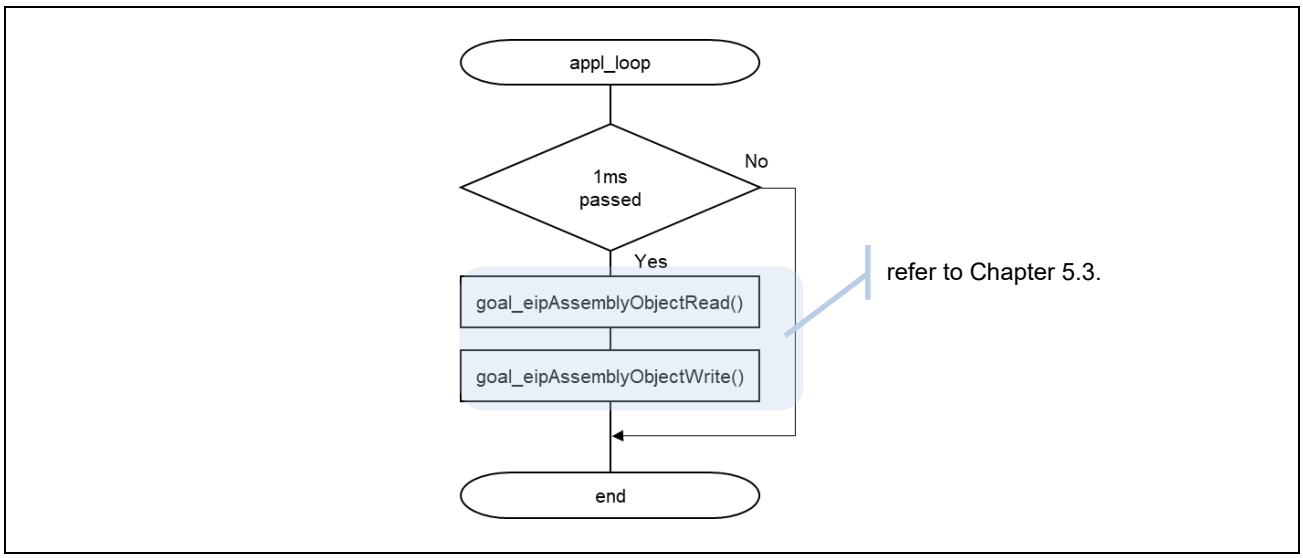

Figure 2-11 EtherNet/IP appl_loop() flow

2.3.4 callback

コールバック関数(main_eipCallback())では、プロトコルスタックから報告される状態に応じた処理を行います。ユーザ仕様に応じて処理を実装してください。

なお、プロトコルスタックから報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

この状態に応じた処理の 1 つとして LED ステータス(3.3 章参照)を成形しています。この成形したステータスは plat_ledSet()の引数に用いられ、plat_ledSet()が呼ばれるタイミングで各 LED の状態を更新します。

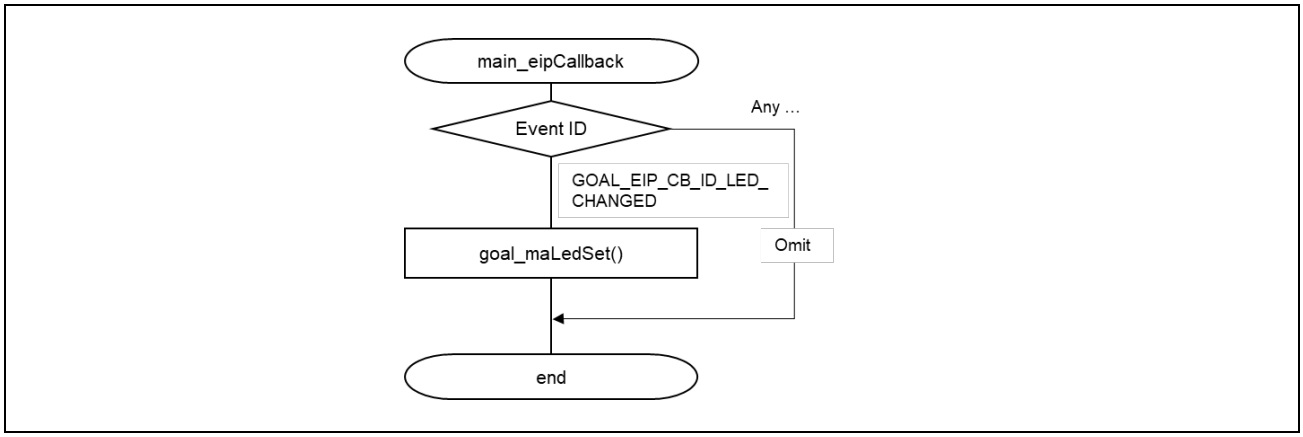

Figure 2-12 EtherNet/IP main_eipCallback() flow

2.4 EtherCAT ユーザアプリケーション

EtherCAT サンプルアプリケーションについて説明します。

対象サンプルを Table 2-7 に示します。

Table 2-7 EtherCAT Sample software

Sample software	Overview
03_ecat	サイクリック通信 サンプル
06_ecat_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル
13_ecat_http	03_ecat に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

2.4.1 appl_init() - 初期化

EtherCAT プロトコルスタックの起動、及び、使用する uGOAL の各モジュールの初期化を行います。ユーザ側で処理を追加する必要はありません。

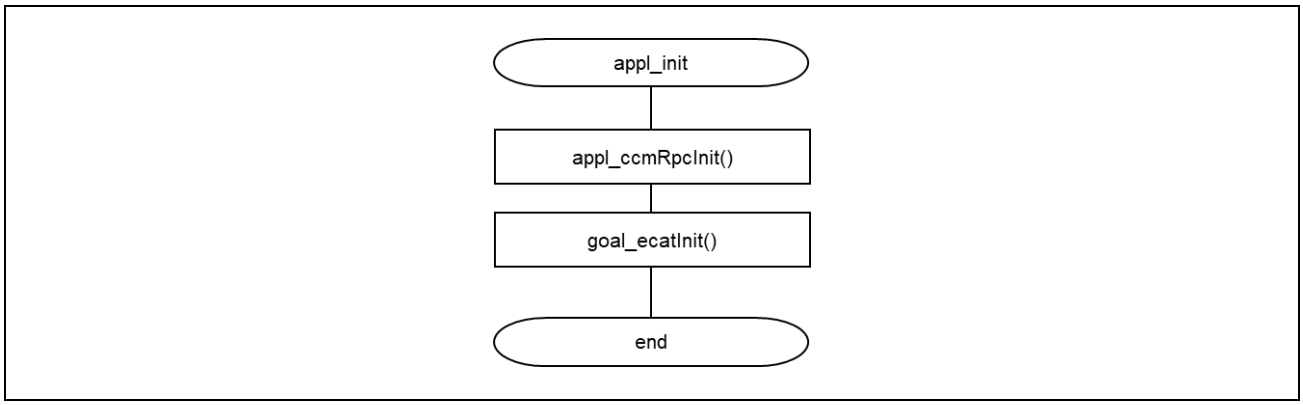

Figure 2-13 EtherCAT appl_init() flow

2.4.2 appl_setup() - パラメータ設定

LED の初期化、プロトコルのプロファイル設定(ベンダ ID 設定等)、及び、Input/Output データ設定を行います。そのため、6.1、6.2 章を参考に、ユーザ仕様に応じてこれらの設定を行ってください。

また、goal_ecatNew()関数にコールバック関数(appl_ecatCallback())を登録することで、プロトコルスタックから通知されるイベントに対する準備を行います。

その他、DC(Distributed Clock)機能をサポートする場合には、goal_maEventOpen()関数を用いて IRQ 割込みを使用できる状態にします。

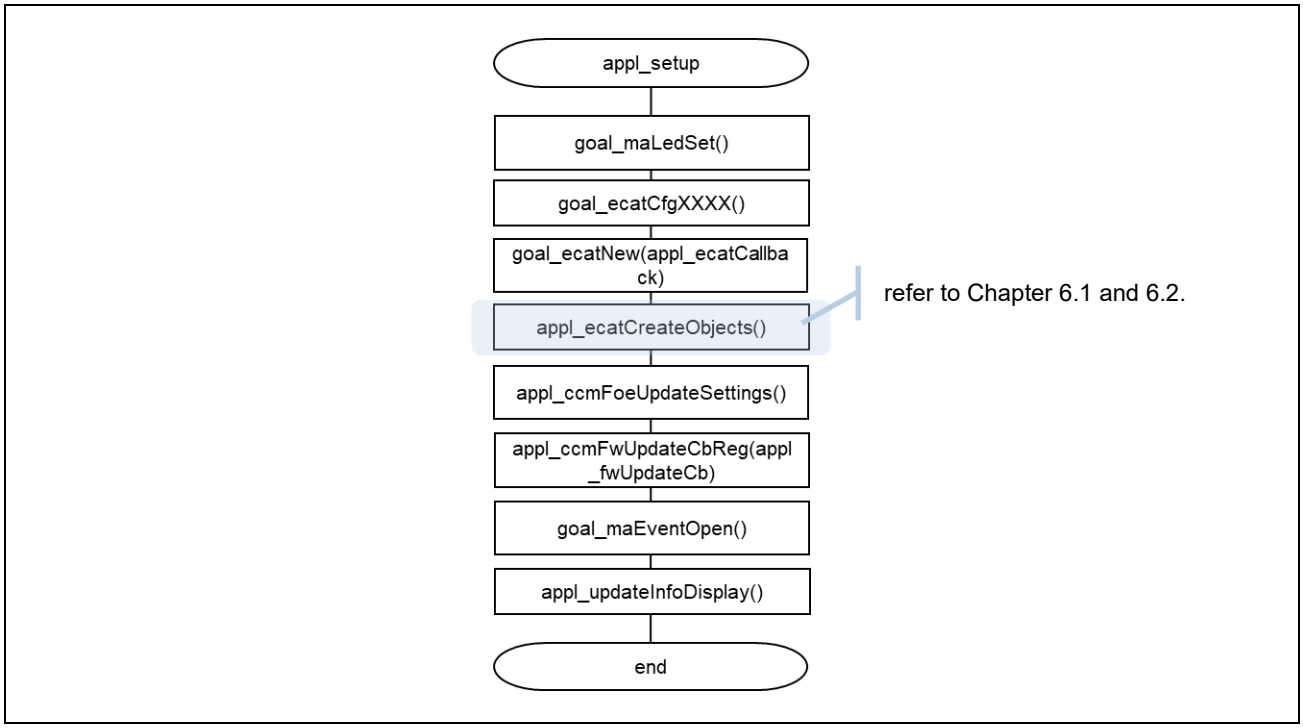

Figure 2-14 EtherCAT appl_setup() flow

2.4.3 appl_loop() - データ送受信

appl_loop()でユーザアプリケーションを実装します。6.3 章を参考にデータ送受信を構成してください。なお、データ送受信の更新周期は 1ms で行っています*。

[* 注意] 1ms 周期は 03_ecat サンプルでの設定です。

扱うデータサイズにより更新周期を考慮する必要があります。参照：C-データサイズ上限

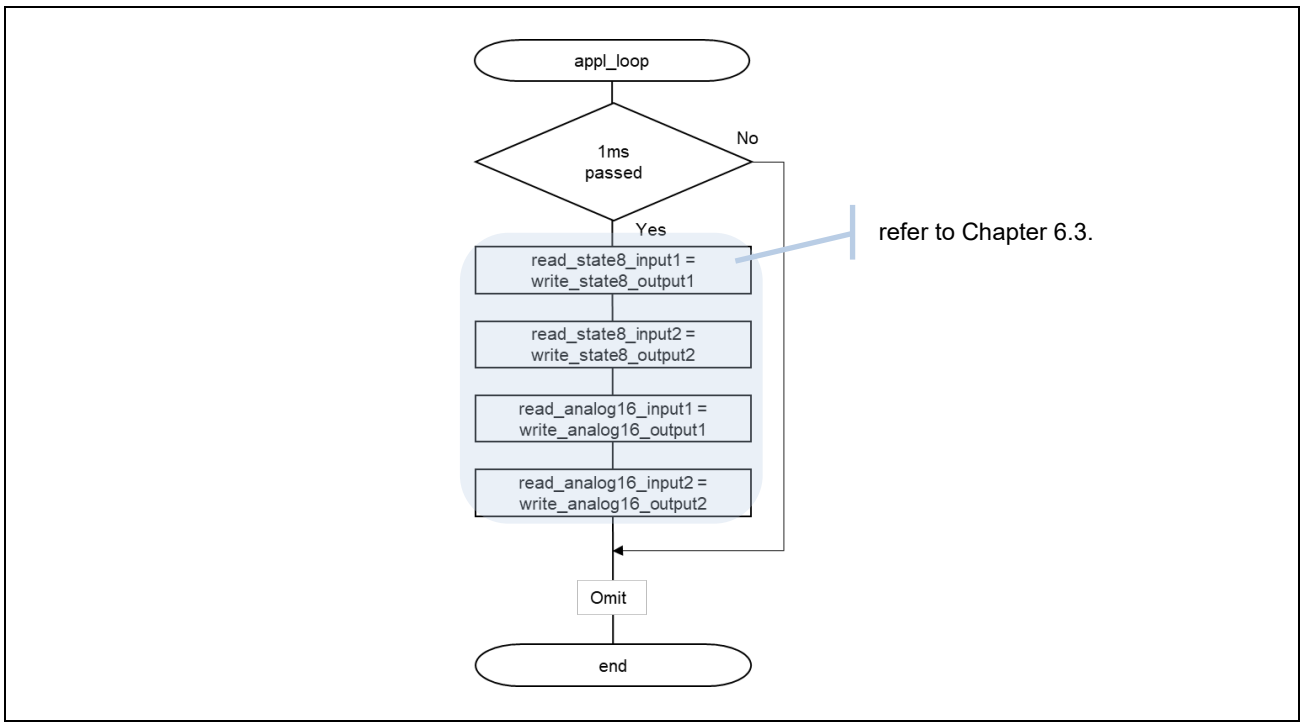

Figure 2-15 EtherCAT appl_loop() flow

2.4.4 callback

コールバック関数(appl_ecatCallback())では、プロトコルスタックから報告される状態に応じた処理を行います。ユーザ仕様に応じて処理を実装してください。

なお、プロトコルスタックから報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

この状態に応じた処理の 1 つとして LED ステータス(3.3 章参照)を成形しています。この成形したステータスは plat_ledSet()の引数に用いられ、plat_ledSet()が呼ばれるタイミングで各 LED の状態を更新します。

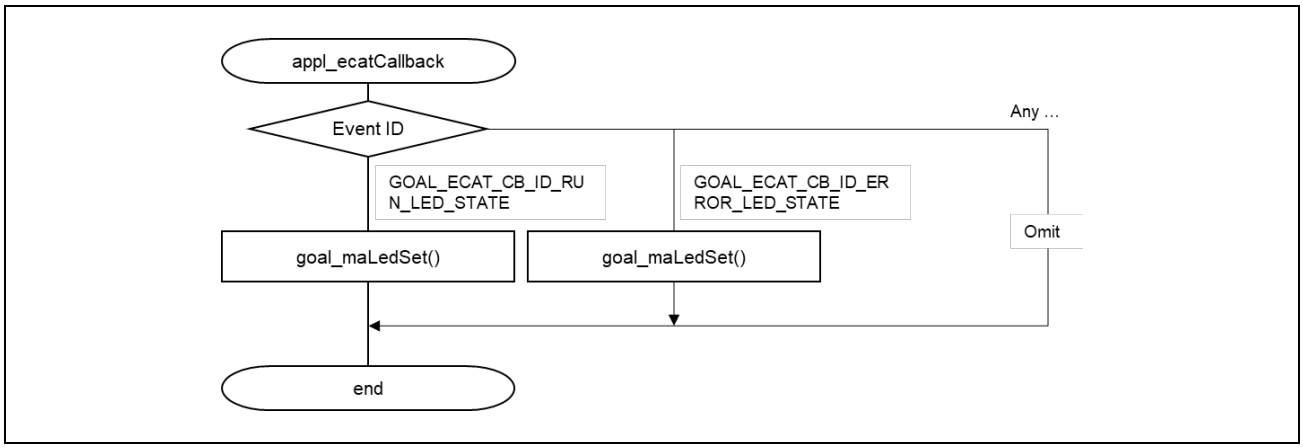

Figure 2-16 EtherCAT appl_ecatCallback() flow

3. ハードウェア周辺

本章では、R-IN32M3 Module を使用するためのハードウェア設計ガイドを含む、ハードウェア周辺のドライバおよびソフトウェアの実装ガイドについて説明します。
推奨回路やレイアウト注意事項等ハードウェアの詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ハードウェア編 (R19UH0122JJ****)』 を参照してください。

以降、それぞれの実装例について説明します。

3.1 Board 初期化 & ドライバ初期化

ホストコントローラを搭載した基板の仕様に合わせて初期設定を行います。
uGOAL では、ハード依存部分の初期化用 API（plat_init）が用意されており、当該 API にて初期設定することを想定しています。ホスト CPU の端子設定など、ホスト CPU として必要な初期化処理を追加してください。また、当該 API にて R-IN32M3 Module のリセット解除を行うようにしてください。
また、以降で示す I2C, SPI, UART の各ドライバに対する初期設定も行います。

以下に RA6M4 版での実装例を示します。

対象ファイル：

plat\ra6m4ek\plat.c

対象 API：

(a) plat_init

・周辺モジュール初期化
RA6M4 版サンプルでは、スマート・コンフィグレータ上で使用する周辺モジュールが管理されているため、ユーザ側で初期化する必要はありません。

・I/O ポート設定
RA6M4 版サンプルでは、スマート・コンフィグレータ上で使用する I/O ポートが管理されているため、ユーザ側で初期化する必要はありません。

・**R-IN32M3 Module リセット解除**

下記コードにて、リセット端子に割り付けられた I/O ポートを操作してリセットを解除しています。

```
/* reset AC */
platResetPeer(NULL);
```

```
static GOAL_STATUS_T platResetPeer(
    struct GOAL_MI_MCTC_INST_T *pMiMctc          /**< MI MCTC instance */
)
{
    UNUSEDARG(pMiMctc);

    g_ioport.p_api->pinWrite(g_ioport.p_ctrl, GOAL_IEM_RST_PIN, GOAL_IEM_RST_SELECT);
    g_ioport.p_api->pinWrite(g_ioport.p_ctrl, GOAL_IEM_RST_PIN, GOAL_IEM_RST_DESELECT);
#if GOAL_CONFIG_RESET_DELAY == 1
    {
        bsp_io_level_t value;
        do
        {
            g_ioport.p_api->pinRead(g_ioport.p_ctrl, GOAL_IEM_RST_PIN, &value);
        }
        while (value != GOAL_IEM_RST_DESELECT);
    }
#endif

    return GOAL_OK;
}
```

・**システムタイマの設定**

システムタイマとして、スマート・コンフィグレータ上で組み込んだ汎用 PWM タイマ（GPT）モジュールのチャネル 0 の動作を開始しています。本サンプルでは、goal_tgtCommonSystickHandler をコールバック関数として登録し、１ミリ秒毎に呼び出されるよう設定しています。

```
    /* start timer */
    g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg);
    g_timer0.p_api->start(g_timer0.p_ctrl);
```

・**I2C, SPI, UART ドライバの設定**

スマート・コンフィグレータ上で登録した I2C, SPI, UART チャネルをオープンし、使用できる状態にします。

```
    /* I2C */
    g_i2c_master0.p_api->open(g_i2c_master0.p_ctrl, g_i2c_master0.p_cfg);
    g_i2c_master0.p_api->abort(g_i2c_master0.p_ctrl);

    /* SPI */
    g_spi0.p_api->open(g_spi0.p_ctrl, g_spi0.p_cfg);

    /* UART */
    g_uart0.p_api->open(g_uart0.p_ctrl, g_uart0.p_cfg);
```

3.2 R-IN32M3 Module 接続

R-IN32M3 Module とホスト CPU の接続について説明します。

3.2.1 ホスト CPU 条件

Host CPU の推奨条件を以下に記します。 メモリ容量は R-IN32M3 Module サンプルソフトウェアおよび EK-RA6M3 評価環境でのメモリ容量を一例として示しています。

ROM 容量	: 64KB 以上
RAM 容量	: 32KB 以上
SPI パケット転送サイズ	: 128 バイト（8 ビット×128 回）一括データ転送

<注意>

- ✓ メモリ容量は 開発環境、マイコンプラットフォーム、コンパイラによって異なります。
- ✓ ルネサス製 CPU を使用する場合、RSPI は最大一括データ転送量が 32 バイトのため、簡易 SPI（SCI）を使用して下さい。

3.2.2 ハードウェア接続例

R-IN32M3 Module とホスト CPU を接続した場合の接続例を、Figure 3-1 に示します。

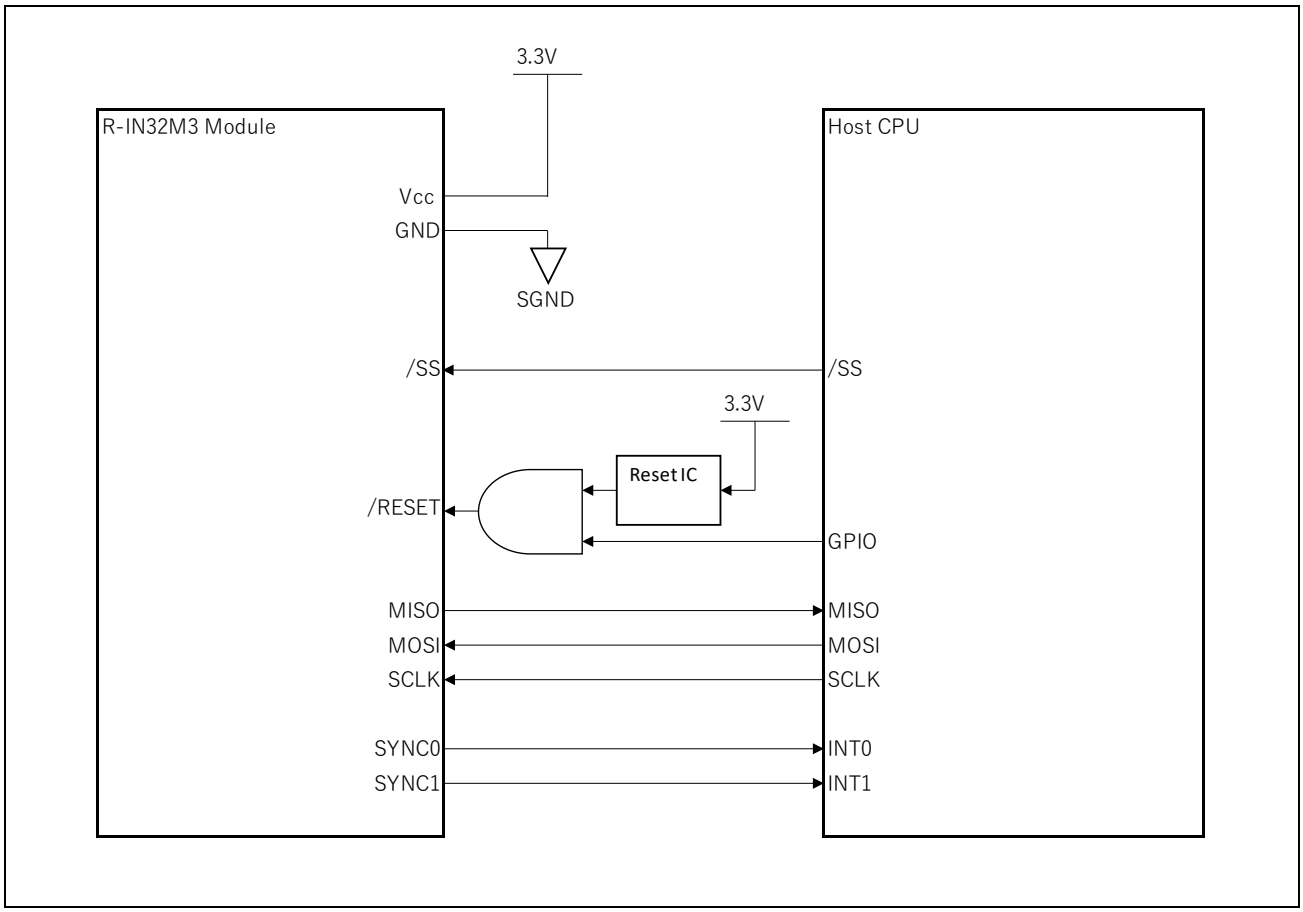

Figure 3-1 Connection to a Host CPU

3.2.2.1 ピン機能

R-IN32M3 Module ピンは、電源、スレーブインタフェースである SPI との連結、リセットの適用、およびクロック信号として機能します。

Table 3-1 Pin Description

ピン	信号	I/O	機能
1	V_{CC}	—	3.3V±0.15V DC 電源
2	GND	—	グランド
3	/SS	I	スレーブ選択： アクティブ Low でスレーブデバイスを有効にする
4	/RESET	I	R-IN32M3 Module 全体のリセット： アクティブ Low
5	MISO	O	マスターイン、スレーブアウト： スレーブからマスターへのデータ出力
6	MOSI	I	マスターアウト、スレーブイン： マスターからスレーブへのデータ入力
7	SCLK	I	シリアルクロック： 本クロックに同期して、データが出力されます。
8	SYNC0	O	ディストリビュートクロック用の EtherCAT 同期信号
9	SYNC1	O	ディストリビュートクロック用の EtherCAT 同期信号

注： ディストリビュートクロック用 EtherCAT 同期信号である、ピン 8 およびピン 9 は、EtherCAT プロトコルでのみ使用されます。

各ピンの内部回路を Figure 3-2 に示します。

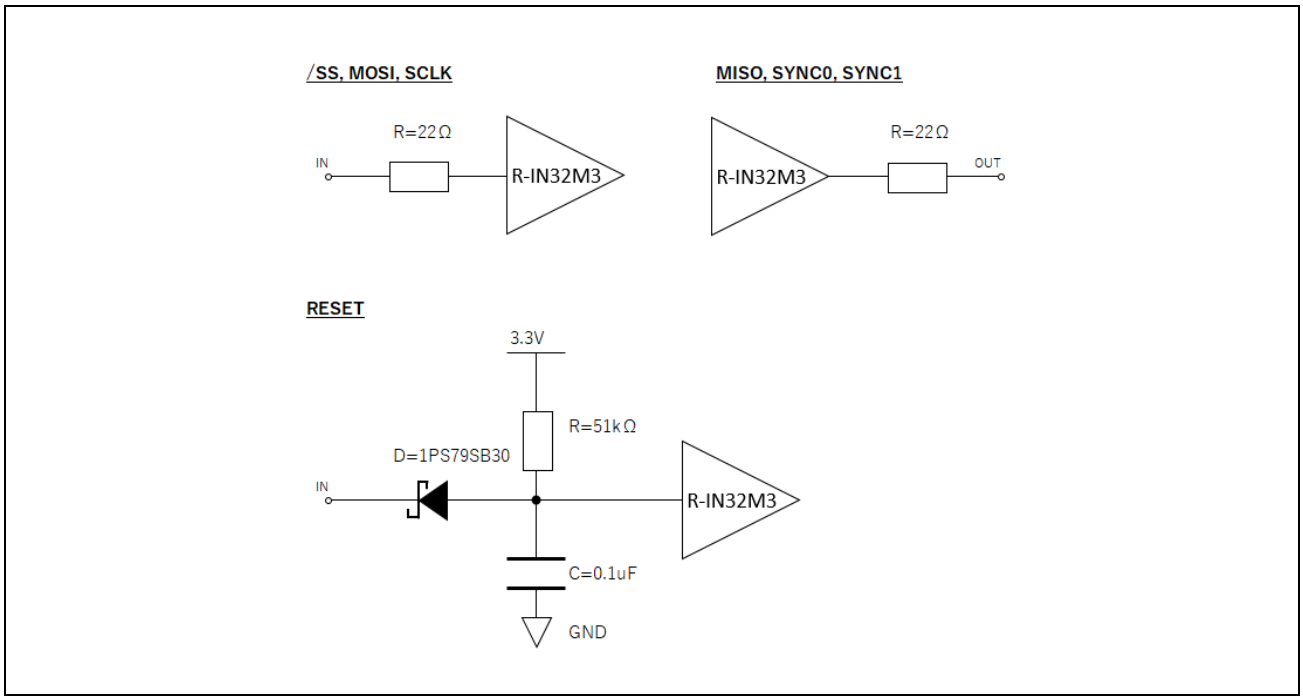

Figure 3-2 R-IN32M3 Module internal circuits

3.2.2.2 電源、リセット

Figure 3-3 に R-IN32M3 Module への電源の供給と遮断の推奨シーケンス、およびリセット信号のタイミング仕様を示します。

R-IN32M3 Module の供給電圧は、3.3V DC±0.15V DC（3.15V～3.45V）として下さい。

R-IN32M3 Module の最大消費電力は約 2.0W であるため、外部電源は 1.0A（またはそれ以上）を供給できることを推奨します。

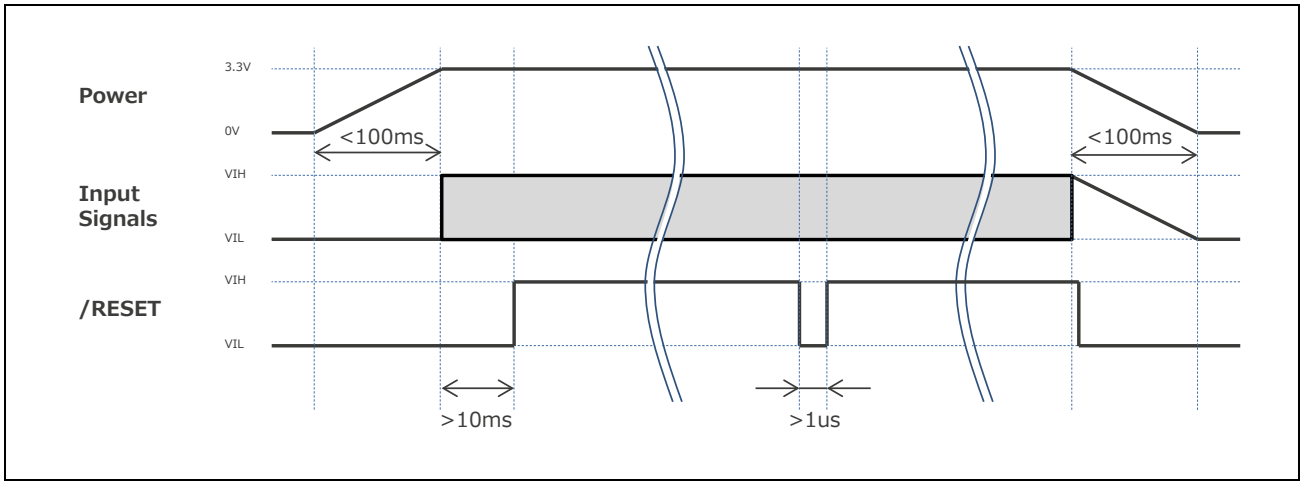

Figure 3-3 Power and Reset sequence

R-IN32M3 Module のリセット端子は内部でプルアップされているため(Figure 3-2)、外部にプルアップ抵抗は必要ありません。Figure 3-3 のリセット仕様を満足するためのリセット回路の例を Figure 3-4 に示します。この場合、3.3V スーパーバイザ (ISL88011H529Z-TK) に接続するホスト CPU からの/RESET 信号はオープンドレイン出力設定(NCODR=1) とします。また、ISL88011H529Z-TK を使用した場合はコンパイルオプション GOAL_CONFIG_RESET_DELAY を有効にして、R-IN32M3 Module への/RESET 解除の完了を待ってから次の処理に進むようにしてください。

ホスト CPU によるリセット解除は Board 初期化(plat_init) において行っております。詳しくは、3.1(a)章をご参照下さい

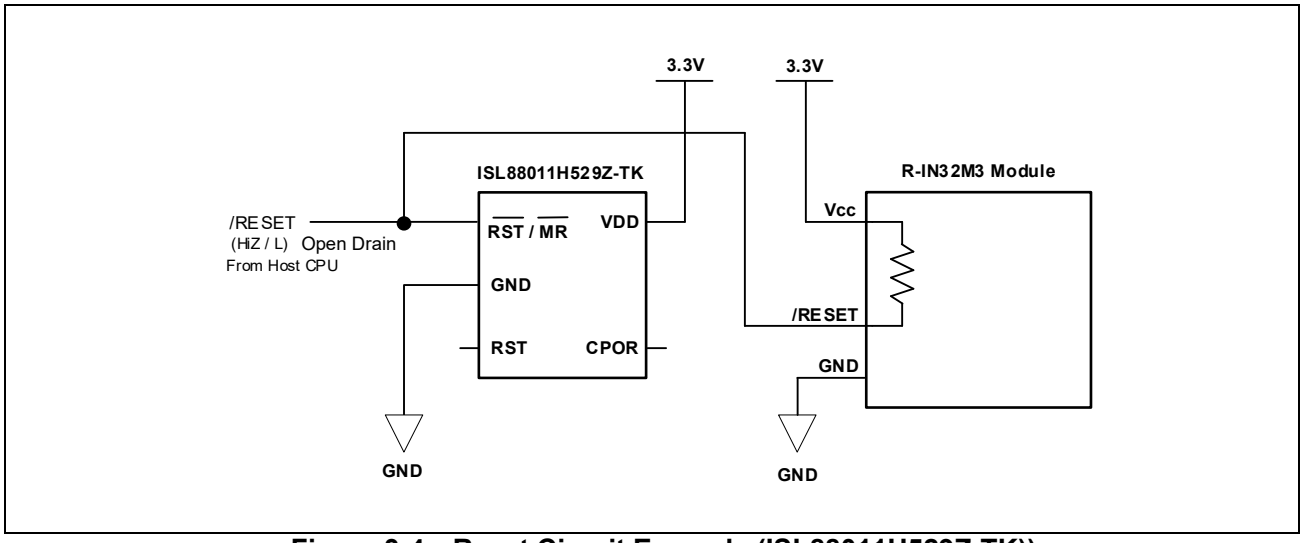

Figure 3-4 Reset Circuit Example (ISL88011H529Z-TK))

3.2.2.3 レイアウト設計ガイドライン

Figure 3-5 は、R-IN32M3 Module の実装に必要なフットプリントを示しています。

- ✓ 灰色の領域はスルーホールを示し、赤色の領域はランドパターンを示します。
- ✓ R-IN32M3 Module の取り付け面は、配線のリードアウト部分を除いて、確実にアースする必要があります。
- ✓ P.C.B の内層に制限はありません。P.C.B の推奨の厚みは、1.6mm です。
- ✓ R-IN32M3 Module の裏側に部品を取り付けることは禁止されています。

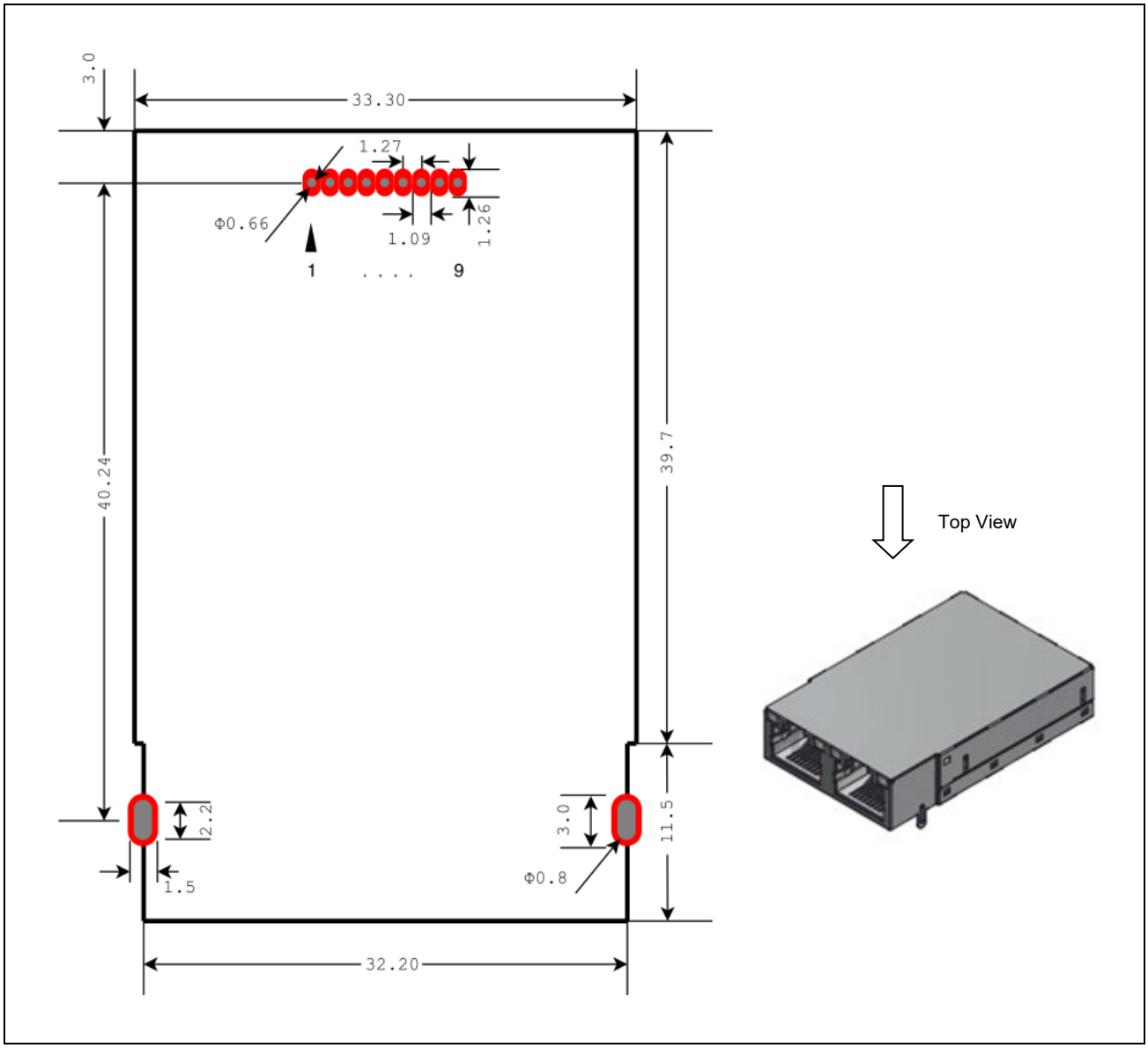

Figure 3-5 Footprint Drawing (Top View)

3.2.3 SPI

3.2.3.1 SPI 仕様

R-IN32M3 Module には、R-IN32M3-EC コントローラでサポートされるシリアルペリフェラルインタフェースがあります。この通信インタフェースは、Table 3-2 に示すように 4 本の信号線を使用します。

R-IN32M3 Module は常に**スレーブモード**で動作し、設定やプロセスデータをホスト CPU に送信します。

Table 3-2 SPI Signal Description

信号	機能
SCLK	シリアルクロック入力（マスターからの出力）
MOSI	マスター出力スレーブ入力、またはマスターアウト、スレーブイン（マスターからのデータ出力）
MISO	マスター入力スレーブ出力、またはマスターイン、スレーブアウト（スレーブからのデータ出力）
/SS	スレーブ選択（アクティブ Low、マスターからの出力）

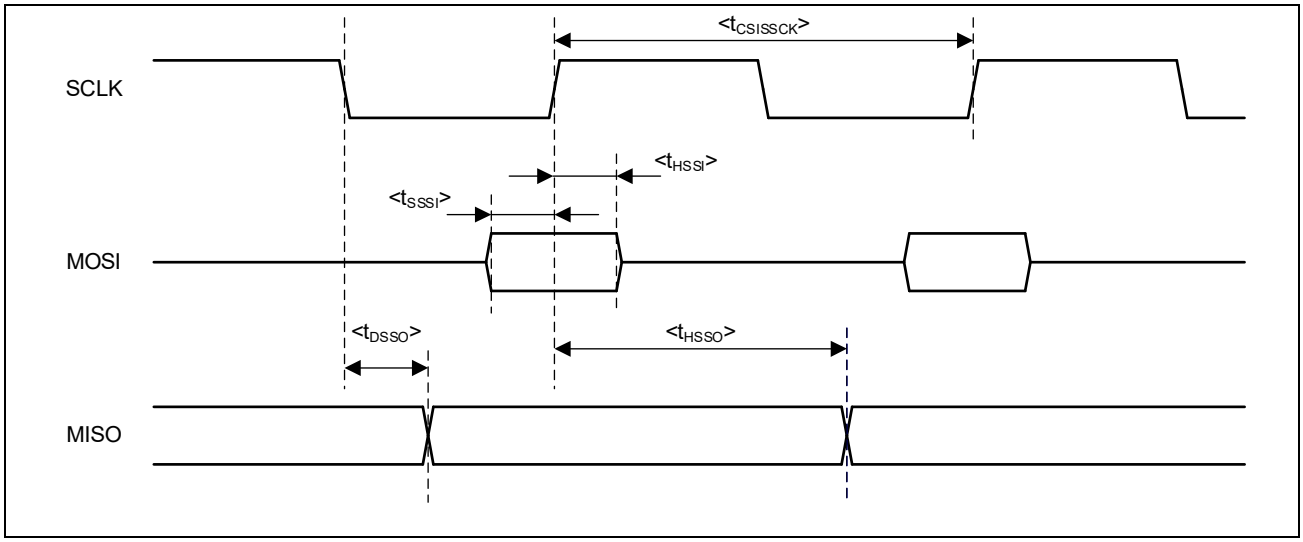

Figure 3-6 SPI Timing Chart

Table 3-3 SPI Specifications（V_{CC}＝3.3±0.15V、T_a＝−40～+70°C）

項目	略号	条件	Min.	Max.	単位
SCLK 入力サイクル	$t_{CSISSCK}$	—	60	—	ns
SCLK 出力 High レベル幅	t_{WSKH}	—	$t_{CSIMSCK}$×0.5−5.0	—	ns
SCLK 出力 Low レベル幅	t_{WSKL}	—	$t_{CSIMSCK}$×0.5−5.0	—	ns
MOSI 入力セットアップ時間（対 CSISCKn↑）	t_{SSSI}	—	10	—	ns
MOSI 入力セットアップ時間（対 CSISCKn↓）	t_{SSSI}	—	10	—	ns
MOSI 入力ホールド時間（対 CSISCKn↑）	t_{HSSI}	—	15	—	ns
MOSI 入力ホールド時間（対 CSISCKn↓）	t_{HSSI}	—	15	—	ns
MISO 出力遅延時間（対 CSISCKn↑）	t_{DSSO}	CL=15pF	—	10	ns
MISO 出力遅延時間（対 CSISCKn↓）	t_{DSSO}		—	10	ns
MISO 出力ホールド時間（対 CSISCKn↑）	t_{HSSO}		$t_{CSISSCK}$×0.5−5.0	—	ns
MISO 出力ホールド時間（対 CSISCKn↓）	t_{HSSO}		$t_{CSISSCK}$×0.5−5.0	—	ns

3.2.3.2 SPI 通信

(1) ソフトウェア実装

R-IN32M3 Module との通信には、シリアルペリフェラルインタフェース（SPI）を使用し、Table 3-4 で示すフレーム構造にて行われる必要があります。uGOAL では、SPI 通信制御用にドライバ(plat/ra6m4ek)が用意されており、当該ドライバにハードウェア制御部分を実装することにより機能を実現します。

ユーザ側で SPI 通信部分の実装を行う際、当該ドライバにおいて 128 バイト単位で指定されたバッファにデータが格納されるよう設計を行ってください。（通信用バッファには送信用 128 バイト(sendData[])と受信用 128 バイト(receiveData[])のバッファがあり、plat_spiTransfer の引数に用いられます。）

Figure 3-7 のように 128Byte の SPI 通信中の/SS 信号は Low として下さい。本サンプルソフトでは、plat_spiTransfer でアサート(Low)、goal_drvSpiSynCb でネゲート(High) にしています。

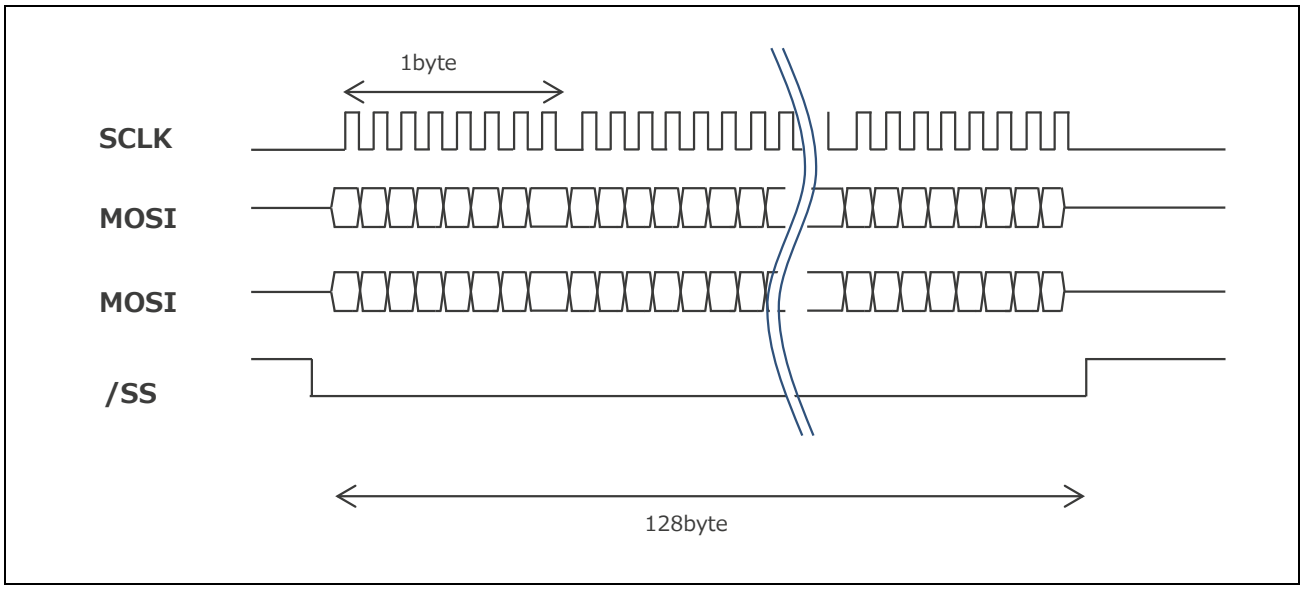

Figure 3-7 SPI Frame Timing

Table 3-4 で、Cyclic Data フレームは SPI フレーム毎に更新した周期通信データを転送します。一方、RPC Data フレームは SPI フレーム毎に更新はされませんが、分割して大容量のデータを転送することができます。このため RPC は主に非同期通信に用いますが、大容量の同期通信にも用いることができます。(Appendix C)

Table 3-4 SPI Frame Structure

Bytes 0..1	Byte 2	Byte 3	Bytes 4..76	Bytes 77..127
Fletchar-16 Checksum with offset 0x0007 (little endian)	Sequence	Data length	Cyclic Data	RPC Data

本サンプルにおける SPI 通信の実装は、uGOAL の SPI 通信制御用ドライバに、SCI モジュールの API を組み込む形式で実装しています。以下、RA6M4 版での実装例を示します。
尚、SPI 通信サイクルの詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

対象ファイル：

plat\ra6m4ek\plat.c

対象 API：

本サンプルでは、スマート・コンフィグレータ上から SPI 通信用 SCI モジュール（r_spi）を組み込み、FSP ドライバ経由で制御しています。

(a) plat_spiTransfer

本 API は、SPI 通信を管理する処理になります。SPI 通信は uGOAL のループ周期をトリガに当処理が起動され、通信が開始されます。本サンプルでは、SPI 通信の CS 端子(/SS)の操作(アサート)および SPI 通信のリードライト処理を SCI モジュールのドライバ API で実装しています。

```
1.          /* Execute write */
2.          g_ioport.p_api->pinWrite(g_ioport.p_ctrl, GOAL_DRV_SPI_CS_PIN, GOAL_DRV_SPI_CS_SELECT);
3.
    … omit …
4.
5.          /* transfer */
6.          fsp_res = g_spi0.p_api->writeRead(g_spi0.p_ctrl,
7.                                            (uint8_t *)txBuf,
8.                                            (uint8_t *)rxBuf,
9.                                            size,
10.                                           SPI_BIT_WIDTH_8_BITS);
11.         if (fsp_res != FSP_SUCCESS)
12.         {
13.             /* disable CS */
14.             g_ioport.p_api->pinWrite(g_ioport.p_ctrl, GOAL_DRV_SPI_CS_PIN,
      GOAL_DRV_SPI_CS_DESELECT);
15.             return GOAL_ERR_SPECIFIC;
16.         }
```

(b) goal_drvSpiSynCb

本 API は、SPI 通信完了時に呼び出されるコールバック関数になります。本サンプルでは、SPI 通信の CS 端子(/SS)の操作(ネゲート)を SCI モジュールのドライバ API で実装しています。

```
1.          /* Deselect CS signal */
2.          g_ioport.p_api->pinWrite(g_ioport.p_ctrl, GOAL_DRV_SPI_CS_PIN, GOAL_DRV_SPI_CS_DESELECT);
```

3.3 プロトコル別 HW 制御 (LED, ID Selector)

本章では、産業 Ethernet プロトコル別に必要な回路設計要件を解説します。

R-IN32M3 Module には、各 Ethernet ポートに 2 つのインジケータ LED（RL45-LED 0G,0Y,1G,1Y）付きの RJ45 メスコネクタが搭載されています。緑色の LED はリンク状態を示し、黄色の LED はネットワークアクティビティに応答して点灯する仕様となっています。(EtherCAT の場合は、Table 3-9 を参照してください。)

これら 2 つのインジケータ LED に関しては、R-IN32M3 Module 側で制御するため、ユーザ側で設計する必要がありません。

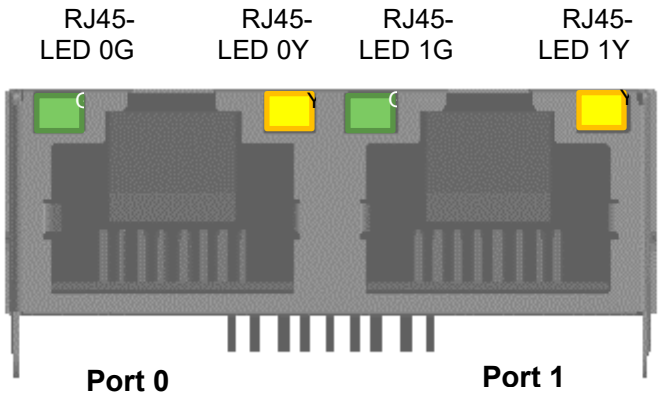

Figure 3-8 RJ45-LED of R-IN32M3 Module

産業 Ethernet に準拠するためには、Table 3-5 に示すステータス LED を対応プロトコルごとに追加する必要があります。R-IN32M3 Module から受信した LED ステータスに応じて、ホスト CPU は LED を制御する必要があります。

Table 3-5 State Indication

産業 Ethernet 規格	ステータス LED1		ステータス LED2	
PROFINET 注 1	システム障害(SF)	赤	バス障害(BF)	赤
	Connection (接続確立)	緑	DCP インジケータ	緑
EtherNet/IP 注 2	モジュール（MS）	緑／赤	ネットワーク（NS）	緑／赤
EtherCAT 注 3	RUN	緑	ERR	赤

注1. PROFINET Diagnosis Guideline V1.4 Chapter 6.7

注2. The CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP

注3. EtherCAT Indicator and Labeling ETG.1300S(R) Vx.x.x

以降、プロトコル毎のハードウェアおよびソフトウェアの実装について説明します。

3.3.1 PROFINET

3.3.1.1 LED 制御

(1) ハードウェア設計

PROFINET では、DCP (Discovery and Configuration Protocol) プロトコルで IP アドレスと装置名を割り当てるときのインジケータが一つ必要とされております。また、必須ではありませんが経験的に幾つかの LED を追加することが推奨されており、R-IN32M3 Module では以下に対応しております。

BF(バス障害)
SF(システム障害)
Connection (接続確立)
DCP (DCP 点滅シグナル)

Figure 3-9 は PROFINET LED 接続の回路例を示します。

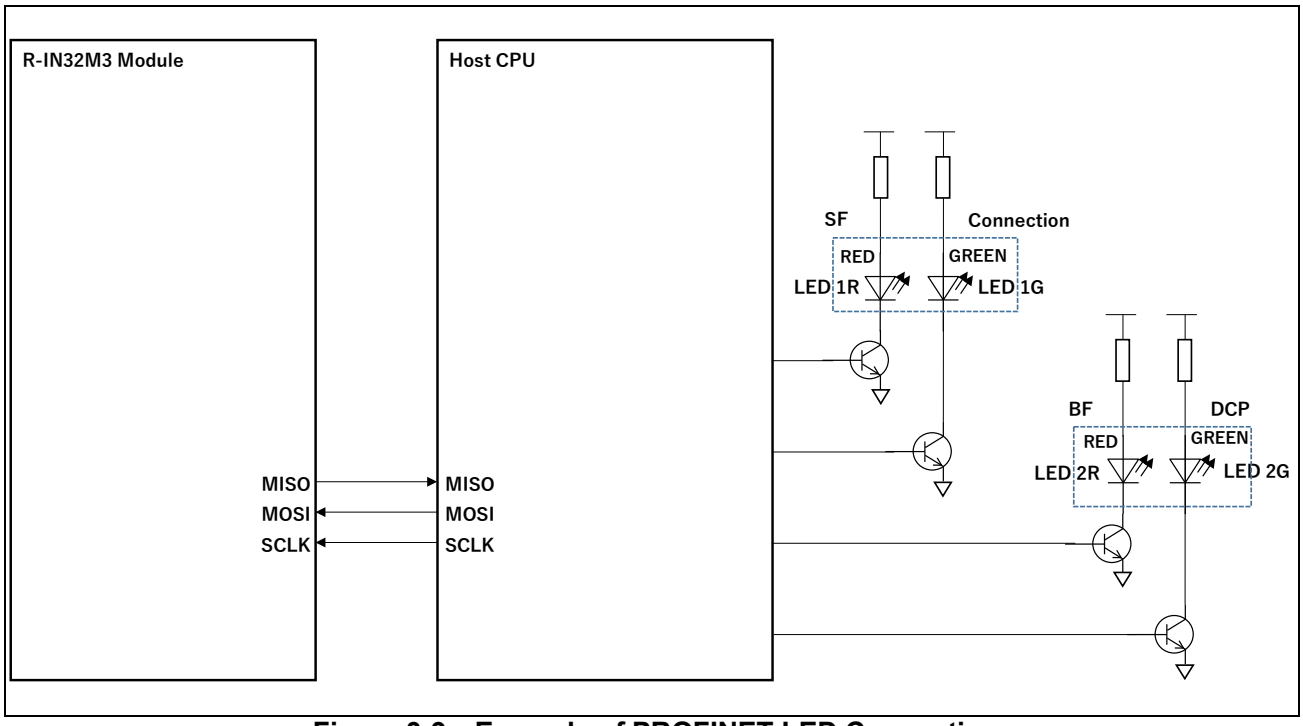

Figure 3-9 Example of PROFINET LED Connections

(2) ソフトウェア実装

Figure 3-9 に示した LED は R-IN32M3 Module とは別に実装し、ホストコントローラで制御する必要があります。R-IN32M3 Module から受信した LED ステータスに応じて、ホスト CPU は LED を制御します。詳細は、以下の Table 3-6 を参照してください。

Table 3-6 PROFINET State Indication by the host controller

LED	名称	色	状態	機能
1R	SF	赤	オン	メンテナンス必要。少なくとも一つの診断エラーがあります。
			オフ	正常時
1G	Connection	緑	オン	PROFINET 通信が確立しています。
			オフ	PROFINET 通信が確立していません。
2R	BF	赤	オン	PROFINET 通信エラー発生。アラーム受信。
			オフ	通常時
2G	DCP	緑	点滅	DCP 点滅表示
			オフ	通常時

uGOAL には、ホストコントローラによる LED 制御用に LED ドライバ(plat/ra6m4ek)が用意されており、当該ドライバにハードウェア制御部分を実装することにより機能を実現します。

RA6M4 版サンプルソフトでは、Arduino 接続を介した I2C 通信を用いて、R-IN32M3 Module 搭載アダプタボード上に実装された LED（LED 1RG, LED 2RG）を制御します。以下に RA6M4 版での実装例を示します。

対象ファイル:

plat\ra6m4ek\plat.c

対象 API :

本サンプルでは、LED を I2C ドライバ経由で制御しているため、スマート・コンフィグレータ上で I2C マスタドライバ（I2C Master on IIC）を組み込み、FSP ドライバ経由で制御しています。

(a) plat_ledSet

uGOAL 上で保持している各 LED の状態を、I2C 通信経由で更新します。

```
1.          /* set new value for LED states */
2.          g_i2c_master0.p_api->slaveAddressSet(g_i2c_master0.p_ctrl, GOAL_DRV_IIC_LED_ADDR,
        I2C_MASTER_ADDR_MODE_7BIT);
3.
4.          ledState[0] = 0x16;                                  /* register offset */
5.          ledState[1] = (state >> 0) & 0xFF;
6.          ledState[2] = (state >> 8) & 0xFF;
7.          ledState[3] = (state >> 16) & 0xFF;
8.          goal_drvIicClearXferStatus();
9.          g_i2c_master0.p_api->write(g_i2c_master0.p_ctrl, (uint8_t *) &ledState[0], 4, false);
10.         goal_drvIicCheckXferStatus(I2C_MASTER_EVENT_TX_COMPLETE);
```

R-IN32M3 Module から受信した LED ステータスは、アプリケーション層で成形されて当該 API の引数に渡されます。成形された引数の LED 情報(4byte)の各 bit には、LED を ON/OFF させる情報が割り当てられています。該当 bit が 1 時に LED が ON になります。

例)

PROFINET 接続確立：0x0404

PROFINET 接続遮断：0x0410

LED を制御する I2C 通信では 4byte データを使用し、以下のように引数の LED 情報のうち 3byte を割り当てます。

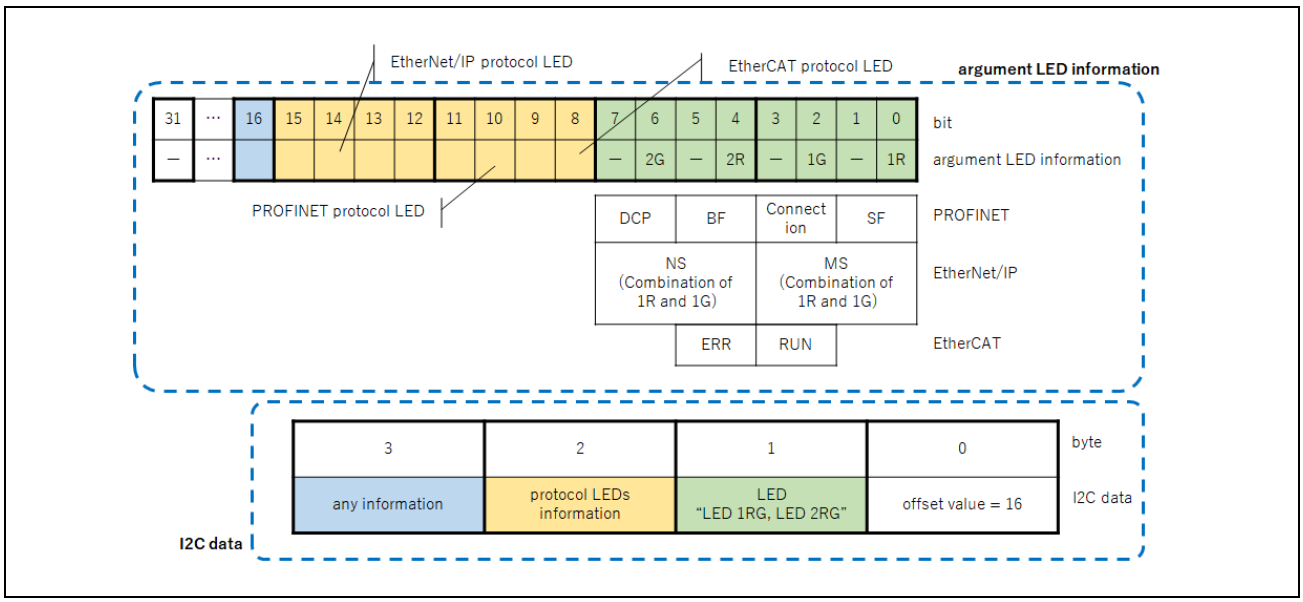

Figure 3-10 Relationship between argument information and I2C data

なお、汎用 I/O ポートを用いて LED（LED 1RG, LED 2RG）を制御する場合は、以降の例を参照ください。

(b) plat_ledSet (for GPIO)

uGOAL 上で保持している各 LED の状態を、汎用 I/O ポート経由で更新します。

```
1.  g_ioport.p_api->pinWrite(LedPinSetting(state, LED1_RED));
2.  g_ioport.p_api->pinWrite(LedPinSetting(state, LED1_GREEN));
3.  g_ioport.p_api->pinWrite(LedPinSetting(state, LED2_RED));
4.  g_ioport.p_api->pinWrite(LedPinSetting(state, LED2_GREEN));
5.  g_ioport.p_api->pinWrite(LedPinSetting(state, ETHERCAT));
6.  g_ioport.p_api->pinWrite(LedPinSetting(state, ETHERNETIP));
7.  g_ioport.p_api->pinWrite(LedPinSetting(state, PROFINET));
```

3.3.2 EtherNet/IP

3.3.2.1 LED 制御

(1) **ハードウェア設計**

Table 3-5 に補足して、EtherNet/IP には、より詳細な LED 制御ガイドラインがあります。EtherNet/IP 通信には、2 種類の 2 色 LED ディスプレイが必要です。

MS（モジュール状態インジケータ）
NS（ネットワーク状態インジケータ）

Figure 3-11 は、EtherNet/IP LED 接続の回路例を示します。

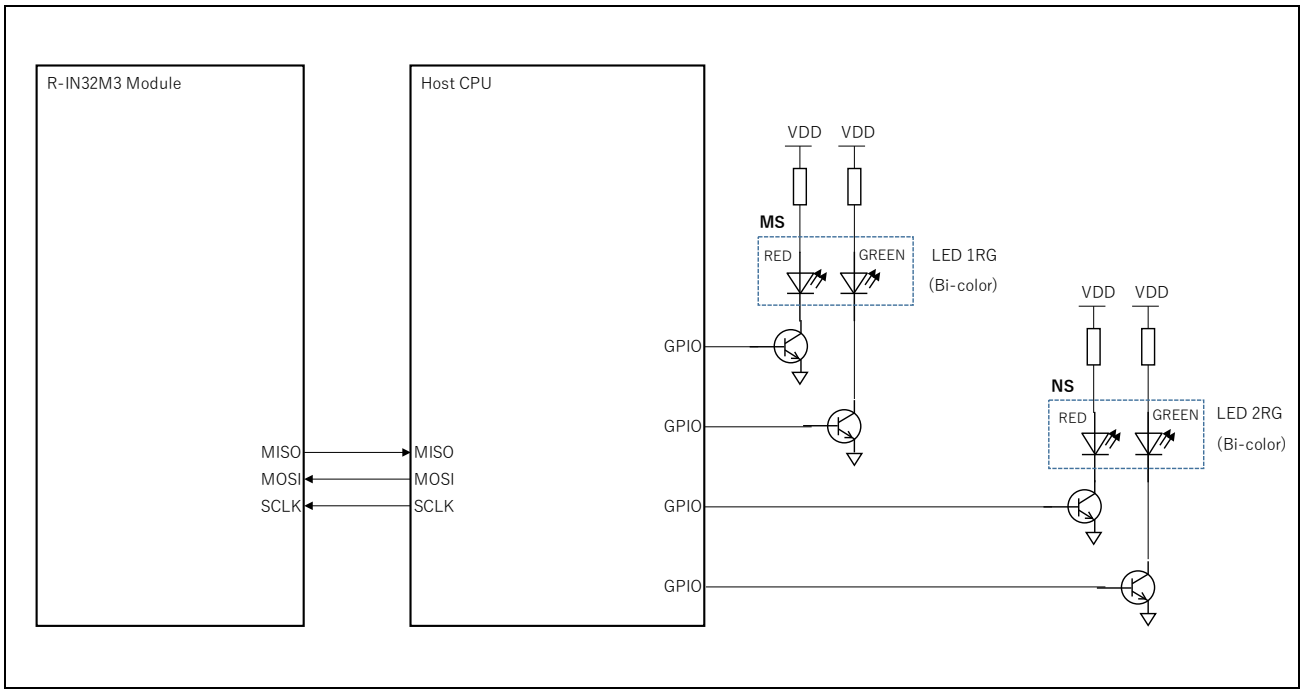

Figure 3-11 Example of EtherNet/IP LED Connections

(2) ソフトウェア実装

MS、NS のインジケータは R-IN32M3 Module とは別に実装し、アプリケーションコントローラで制御する必要があります。R-IN32M3 Module から受信した LED ステータスに応じて、ホスト CPU は LED（LED 1RG, LED 2RG）を制御する必要があります。LED 制御方法に関しては、PROFINET と同じく uGOAL の LED ドライバを使用します。詳細は、3.3.1.1(2) 章を参照してください。

詳細は、Table 3-7 と Table 3-8 を参照してください。

Table 3-7 MS (Module Status Indicator LED 1RG)

インジケータ状態	概要	機能
定常 OFF	電源オフ	デバイスに電源が供給されていない場合は、モジュール状態インジケータは定常オフになります。
定常緑	デバイス動作	デバイスが正常に動作している場合、モジュール状態インジケータは定常で緑色になります。
点滅した緑	スタンドバイ	デバイスが設定されていない場合、モジュール状態インジケータは緑色に点滅します。
点滅した赤	重大な回復可能な障害	デバイスが、重大な回復可能な障害を検出した場合、モジュール状態インジケータは赤色に点滅します。 注. 正しくない設定や一貫性のない設定は、重大な回復可能な障害とみなされます。
定常赤	重大な回復不能な障害	デバイスが、重大な回復不能な障害を検出した場合、モジュール状態インジケータは定常で赤色になります。
点滅した緑/赤	セルフテスト	デバイスが電源供給テストを実行している間、モジュール状態インジケータは、以下に説明するテストシーケンスを適用します。 • モジュール状態インジケータは、約 0.25 秒間緑色になり、約 0.25 秒間赤色になり、その後緑色になり、電源供給テストが完了するまでその状態を保持します。 • モジュール状態インジケータとネットワーク状態インジケータ、どちらもある場合、モジュール状態インジケータのテストシーケンスは、ネットワーク状態インジケータのテストシーケンスの前または同時に発生する必要があります。複数のネットワーク状態インジケータが存在する場合、各ネットワーク状態インジケータのテストシーケンスは、連続して、または同時に進行します。 • この電源供給テストの完了後、インジケータは通常の動作状態を表します。

Table 3-8 NS (Network Status Indicator LED 2RG)

インジケータ状態	概要	機能
定常オフ	電源オフ、 IP アドレスなし	デバイスの電源がオフ、またはオンだが、IP アドレスが設定されていません（TCP/IP インタフェースオブジェクトのインタフェース設定属性）。
点滅した緑	接続なし	IP アドレスは設定されていますが、CIP（共通産業プロトコル）接続[注1]は確立しておらず、独占オーナー接続[注2]はタイムアウトしていません。
定常緑	接続あり	IP アドレスは設定されており、少なくとも 1 つは CIP 接続（どのような転送クラスであれ）が確立しており、独占オーナー接続はタイムアウトしていません。
点滅した赤	接続タイムアウト	IP アドレスが設定されており、本デバイスがターゲットである独占オーナー接続がタイムアウトしました。タイムアウトしたすべての独占オーナー接続が再確立されると、ネットワーク状態インジケータは、定常で緑色の状態に戻ります。 単一の独占オーナー接続をサポートするデバイスでは、その次の独占オーナー接続が確立されると、定常で緑色の状態になります。 複数の独占オーナー接続をサポートするデバイスでは、独占オーナー接続がタイムアウトしたとき、O→T（オリジネーターからターゲットへの）接続パス情報を保持する必要があります。以前にタイムアウトになった O→T 接続ポイントへのすべての接続が再確立された場合にのみ、ネットワーク状態インジケータは、赤の点滅から定常で緑色の状態に移行します。 独占オーナー接続以外の接続がタイムアウトしても、インジケータが赤く点滅することはありません。 赤く点滅する状態は、ターゲット接続にのみ適用されます。オリジネーターと CIP ルーターが原因で、LED がこの状態になることはありません。
点滅した緑／赤	セルフテスト	デバイスが電源供給テストを実行している間、ネットワーク状況インジケータは、Table 3-7 に記載されているテストシーケンスを実行します。

注1. CIP（共通産業プロトコル）は、オープンアプリケーション層プロトコルであり、EtherNet/IP はこのプロトコルをアプリケーション層で使用します。詳細については、EtherNet/IP 仕様を参照してください。

注2. 独占オーナー接続は、モジュールの出力を制御するために使用され、その他の条件には依存していません。モジュールに対してオープンになる独占オーナー接続は 1 つのみです。詳細については、EtherNet/IP 仕様を参照してください。

3.3.3 EtherCAT

3.3.3.1 LED 制御

(1) **ハードウェア設計**

Table 3-5 で示したことに補足して、EtherCAT には、より詳細な LED 制御ガイドラインがあります。EtherCAT 通信には、4 種類の LED 表示が必要です。

L/A IN（リンク/アクティビティ IN）

L/A OUT（リンク/アクティビティ OUT）

RUN（デバイス状態インジケータ）

ERR（エラー状態インジケータ）

EtherCAT の規格としてデイジーチェーントポロジやリングトポロジなど、複数台を接続する時は、通信ポート 0 を IN、通信ポート 1 を OUT として接続する必要があります。各通信ポートと IN と OUT の関係は、EtherCAT の規格上、ハードウェア的に固定となり、機器に IN と OUT を明記する必要があります。

Figure 3-12 は EtherCAT LED 接続、ID セレクタの例を示します。例では、RUN（LED 1G）及び ERR（LED 2R）の LED は、独立した LED としておりますが、1 つの 2 色 LED（STATUS LED）として置き換えることも可能です。

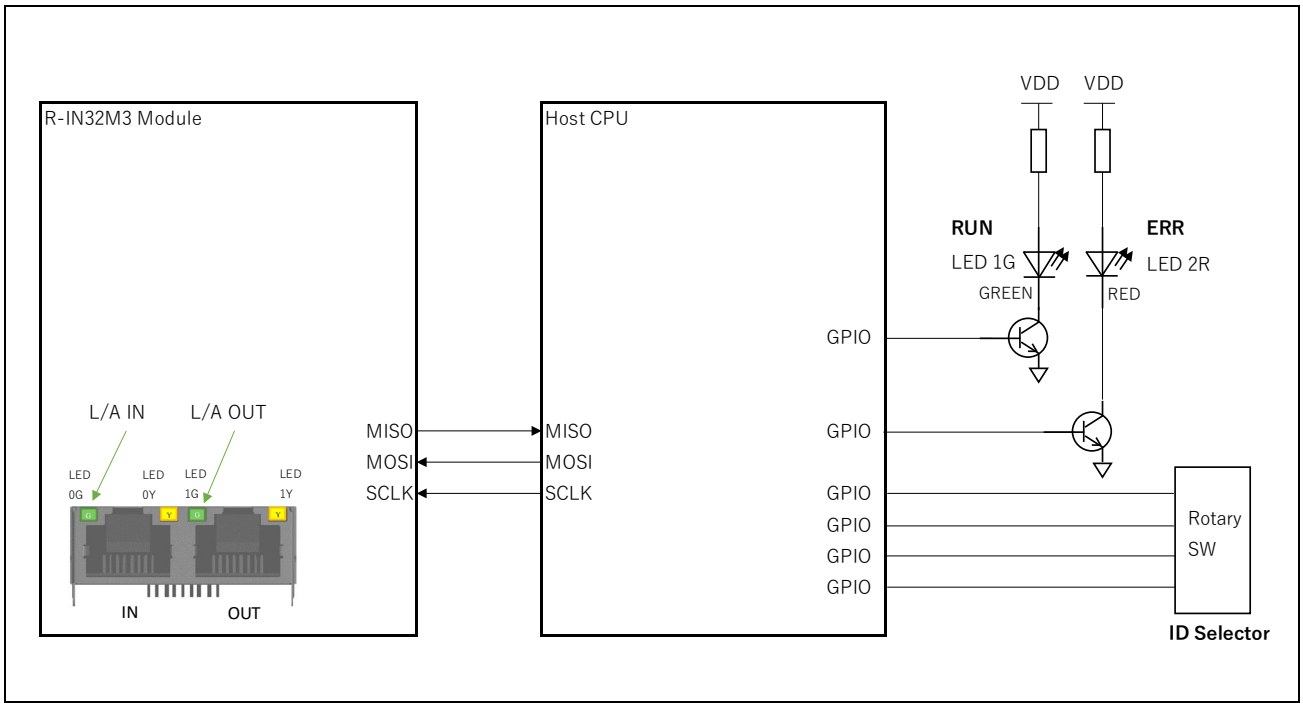

Figure 3-12 Example of EtherCAT LED and ID-selector connection

L/A IN および L/A OUT 用の LED は、R-IN32M3 Module に含まれており、R-IN32M3 Module によって制御されます。このため、ユーザ側で設計する必要がありません。詳細は、Table 3-9 を参照してください。

Table 3-9 LED status display for L /A IN and L/ A OUT by R-IN32M3Module

RJ45-LED	LED 名称	色	状態	機能
0G, 1G	L/A IN 及び L/A OUT インジケータ	緑	OFF	Link 未確立
			Flickering	Link 確立、データ送受信あり。
			ON	Link 確立、データ送受信なし。

(2) ソフトウェア実装

RUN 用 LED および ERR 用 LED は、R-IN32M3 Module とは別に実装し、ホストコントローラで制御する必要があります。R-IN32M3 Module から受信した LED ステータスに応じて、ホスト CPU は LED を制御する必要があります。LED 制御方法に関しては、PROFINET と同じく uGOAL の LED ドライバを使用します。詳細は、3.3.1.1(2) 章を参照してください。

Figure 3-12 の例における LED 制御使用を Table 3-10 に示します。

Table 3-10 EtherCAT status indicator by host controller

LED	LED 名称	色	状態	機能
1G	RUN	緑	OFF	デバイスは初期状態です。
			Blinking	デバイスはプリ・オペレーション状態です。
			Single flash	デバイスは、セーフ・オペレーション状態です。
			ON	デバイスは、オペレーション状態です。
2R	ERR	赤	OFF	エラー無しであり、EtherCAT 通信状態です。
			Blinking	通信設定異常です。
			Single flash	同期イベント異常です。
			Double flash	同期マネージャーウォッチドッグタイムアウト。
			Flickering	初期化異常です。
			ON	PDI 異常です。

3.3.3.2 ID Selector

(1) ハードウェア設計

EtherCAT では、明示的なデバイス ID セレクタが必須となります。ID セレクタの形については、ロータリースイッチ、コントロール付きディスプレイなどどのタイプでも構いません。

Figure 3-12 に ID セレクタの接続例を示します。

RA6M4 版サンプルソフトでは、R-IN32M3 Module 搭載アダプタボードを Arduino 接続した EK-RA6M4 ボードの Pmod2 コネクタの 1-6pin に Digilent 社製 Pmod SWT ボードを接続した状態で、実装および動作確認を行っております。

(2) ソフトウェア実装

uGOAL では、ID セレクタによるデバイス ID の取り込み用に ID ドライバ(plat/ra6m4ek)が用意されており、当該ドライバにハードウェア制御部分を実装することにより機能を実現します。

以下に RA6M4 版での実装例を示します。

対象ファイル：

plat\ra6m4ek\plat.c

対象 API：

(a) goal_drvIdSynGet

Pmod2 コネクタ上段(1-6pin)に接続された汎用 I/O ポート（P413, P411, P410, P412）の値を、デバイス ID として返却します。

```
1.      if (NULL != pId) {
2.          g_ioport.p_api->pinRead(g_ioport.p_ctrl, GOAL_PMOD_ECAT_ID1, &value);
3.          *pId = value;
4.          g_ioport.p_api->pinRead(g_ioport.p_ctrl, GOAL_PMOD_ECAT_ID2, &value);
5.          *pId |= (value << 1);
6.          g_ioport.p_api->pinRead(g_ioport.p_ctrl, GOAL_PMOD_ECAT_ID3, &value);
7.          *pId |= (value << 2);
8.          g_ioport.p_api->pinRead(g_ioport.p_ctrl, GOAL_PMOD_ECAT_ID4, &value);
9.          *pId |= (value << 3);
10.     }
```

3.3.3.3 DC 制御

(1) ハードウェア設計

EtherCAT のディストリビュートクロック (Distributed Clocks : DC) 同期モードを使用する場合は、Figure 3-1 の接続例のように、R-IN32M3 Module の SYNC0 および SYNC1 信号を、ホスト CPU の外部割り込み端子に接続する必要があります。

<注意>

R-IN32M3 Module 搭載アダプタボード(YCONNECT-IT-I-RJ4501) を使用する場合は、接続するホスト CPU ボードに応じて、該当する Arduino コネクタの端子を適宜ショートする必要があります。

EK-RA6M4 ボード、および、RL78/G14 Fast Prototyping Board 用サンプルソフトをご使用の際は、Arduino コネクタ J10 の 3 ピンと 6 ピン、および 4 ピンと 7 ピン をそれぞれショートしてください。詳細は、『YCONNECT-IT-I-RJ4501 ユーザーズマニュアル (R12UZ0094JJ****) 2.5 章』をご参照ください。

---

(2) ソフトウェア実装

ディストリビュートクロック(DC)機能を実装する場合、uGOAL では R-IN32M3 Module で発生する SYNC0/SYNC1 イベントをホスト CPU 側で外部割り込みとして受信し、uGOAL 内のイベントに変換して処理します。イベント処理用にドライバ(plat/ra6m4ek)が用意されており、当該ドライバにハードウェア制御部分を実装することにより機能を実現します。

以下に RA6M4 版での実装例を示します。

対象ファイル :

plat\ra6m4ek\plat.c

対象 API :

本サンプルでは、外部割り込みを IRQ 割り込みとして制御します。スマート・コンフィグレータ上で割り込み制御ユニットドライバ（r_icu）を組み込み、FSP ドライバ経由で制御しています。

(a) plat_eventRegister

本 API は、IRQ 割り込みを使用できる状態にする際に呼び出されます。本サンプルでは、ICU モジュールのオープン処理 goal_maEventOpen()を呼び出し、スマート・コンフィグレータ上で登録した IRQ 割り込みが使用できる状態にしています。

```
1.  result = R_ICU_ExternalIrqOpen(mEventCfg[cnt].pInst->p_ctrl, mEventCfg[cnt].pInst->p_cfg);
2.  if (FSP_SUCCESS != result) {
3.      return GOAL_ERR_NULL_POINTER;
4.  }
5.  if (GOAL_TRUE == flgAutoStart){
6.      R_ICU_ExternalIrqEnable(mEventCfg[cnt].pInst->p_ctrl);
7.  }
```

(b) plat_eventEnable

IRQ 割り込みを有効化します。

```
1.  R_ICU_ExternalIrqEnable(mEventCfg[cnt].pInst->p_ctrl);
```

(c) plat_eventDisable

IRQ 割り込みを無効化します。

```
1.  R_ICU_ExternalIrqDisable(mEventCfg[cnt].pInst->p_ctrl);
```

3.4 OS

uGOAL は OS 依存の機能を使用しない OS レスの環境のため、ユーザ側で実装する必要はありません。

3.5 タイマ

uGOAL ではシステムタイマとしてハードウェアタイマを 1 チャネル使用します。システムタイマで 1 ティック経過毎にティックカウンタ（g_ulTickCounter）をカウントアップするように実装してください。デフォルトでは、1 ティックあたり 1 ミリ秒の設定（GOAL_TGT_TIMER_TICKS_PER_SECOND）となっています。

本サンプルでは、スマート・コンフィグレータにてタイマモジュールを実装したのち、uGOAL の初期化 API（plat_init）にて当該タイマを起動しています。具体的な実装については、3.1 章を参照してください。

3.6 UART

(1) ソフトウェア実装

PC とホスト CPU 間で UART 通信する際に使用します。UART 通信は、ホスト CPU 側のデバッグ機能を使用する際に必要となります。デバッグ機能は、下記のコンパイルマクロを変更することで有効になります。ただし、本サンプルではデバッグ機能をデフォルト無効としています。

Table 3-11 Compile macro for enabling debugging feature

コンパイルマクロ	デフォルト値
GOAL_UGOAL_CONFIG	0

デバッグ機能の有効化の詳細については、『RA サンプルアプリケーション (uGOAL 版) (R30AN0398JJ****)』を参照してください。

<注意>

- ✓ RL78/G14 版サンプルソフトでは、RL78/G14 のメモリ配置制限を理由に、デバッグ用途を目的としたログメッセージを出力する機能は使用できません。

---

uGOAL では、デバッグ機能のために plat_logInfo()で vprintf の標準入出力ライブラリを呼び出して文字列出力を行います。そのため、_write 関数を別途用意して、この vprintf をコンパイル時のオーバライドによって置き換えることで、UART ドライバ経由での PC ターミナルソフト出力を実現します。

また、uGOAL では、UART 通信制御用に UART ドライバ(plat/ra6m4ek)が用意されており、当該ドライバにハードウェア制御部分を実装することにより機能を実現します。

対象ファイル：

plat\ra6m4ek\plat.c

対象 API：

本サンプルでは、SCI モジュールの UART 機能を使って実装しています。スマート・コンフィグレータ上で SCI モジュールの UART ドライバ（r_sci_uart）を組み込み、各 API の呼び出しを行っています。

(a) _write

別途用意した本関数は、文字列の UART 通信書き込みを行う際に呼び出されます。本サンプルでは、SCI モジュールの SCI 通信ライト処理で実装しています。

```
for (index = 0; index < len; index++)
{
    if (*ptr == '\n')
    {
        cr = '\r';
        flg_rawtrans_done = GOAL_FALSE;
        g_uart0.p_api->write(g_uart0.p_ctrl, &cr, 1);
        while (flg_rawtrans_done != GOAL_TRUE) {
        }
    }
    flg_rawtrans_done = GOAL_FALSE;
    g_uart0.p_api->write(g_uart0.p_ctrl, ptr++, 1);
    while (flg_rawtrans_done != GOAL_TRUE) {
    }
}
```

(b) goal_drvSciUartSynCallback

本 API は、UART 通信書き込み完了時に呼び出されるコールバック関数になります。本サンプルでは、転送完了を示す flg_rawtrans_done のセットを行います。

```
flg_rawtrans_done = GOAL_TRUE;
```

4. PROFINET

本サンプルにて、ユーザが設定するプロファイル設定を下記に示します。

スレーブのプロファイル情報は、GSDML (General Station Description Markup Language)ファイルでデバイスに関する機器情報をマスター機器に提供します。そのため、スレーブが保持する機器情報と GSDML ファイルの情報は一致していなければいけません。

ここでは、スレーブが保持する機器情報と GSDML ファイルのプロファイル設定について説明します。

スレーブが保持する機器情報: appl\ 01_pnio\goal_appl.h

GSDML ファイル: appl \01_pnio\gsdml\GSDML-V2.4-Renesas-irj45-20211014.xml

GSDML ファイルに記述される機器情報の一部

- ベンダープロファイル
- データ構成
- 利用可能なデータ通信の種類
- コンフィグレーションパラメータ

GSDML ファイルは XML エディタを使用して作成、変更することが可能です。PROFIBUS & PROFINET International (PI) が提供する PROFINET GSD Checker を利用して XML ファイルを修正、登録情報の正当性確認することが可能です。

profinet-gsd-checker - www.profibus.com

以降で示す API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

4.1 Device Identity

ベンダ ID やベンダ名、プロダクトコード等について説明します。

これらのスレーブが保持する機器情報はプログラムファイル内にマクロで定義されています。

Table 4-1 PROFINET Device Identify

Item	Macro	Default Value [01_pnio]
ベンダ ID	APPL_PNIO_VENDOR_ID	0x02c7
ベンダ名	APPL_PNIO_VENDOR	Renesas Electronics
デバイス ID	APPL_PNIO_DEVICE_ID	0x0300
IP アドレス	MAIN_APPL_IP	192.168.0.100

4.1.1 ベンダ ID

ベンダ ID は PI によって割り当てられます。デフォルト値は 0x02C7 (ルネサスエレクトロニクス)です。

(1) スレーブが保持する機器情報

”APPL_PNIO_VENDOR_ID” の値を変更します。

```
1.  #ifndef APPL_PNIO_VENDOR_ID
2.  # define   APPL_PNIO_VENDOR_ID    (0x02c7)              /**< vendor id */
3.  #endif
```

(2) GSDML ファイル

VendorID の値を変更します。

```
1.      <DeviceIdentity VendorID="0x02C7" DeviceID="0x0300">
2.          <InfoText TextId="TOK_DevIdent_InfoText"/>
3.          <VendorName Value="Renesas Electronics"/>
4.      </DeviceIdentity>
```

4.1.2 ベンダ名

ベンダ名はベンダ ID に紐づいている必要があります。デフォルト値は”Renesas Electronics” (ルネサスエレクトロニクス)です。

(1) スレーブが保持する機器情報

” APPL_PNIO_VENDOR” の値を変更します。

```
#ifndef APPL_PNIO_VENDOR
# define   APPL_PNIO_VENDOR      Renesas Electronics     /**< vendor name */
#endif
```

(2) GSDML ファイル

VendorName Value の値を変更します。

```
        <DeviceIdentity VendorID="0x02C7" DeviceID="0x0300">
            <InfoText TextId="TOK_DevIdent_InfoText"/>
            <VendorName Value="Renesas Electronics"/>
        </DeviceIdentity>
```

4.1.3 デバイス ID

任意の製品コードを設定します。デフォルト値は 0x0300 です。

(1) スレーブが保持する機器情報

” APPL_PNIO_DEVICE_ID” の値を変更します。

```
#ifndef APPL_PNIO_DEVICE_ID
# define   APPL_PNIO_DEVICE_ID      (0x0300)              /**< device id */
#endif
```

(2) GSDML ファイル

DeviceID の値を変更します。

```
        <DeviceIdentity VendorID="0x02C7" DeviceID="0x0300">
            <InfoText TextId="TOK_DevIdent_InfoText"/>
            <VendorName Value="Renesas Electronics"/>
        </DeviceIdentity>
```

4.1.4 IP アドレス

評価用途のために任意の IP アドレスの設定が可能です。

IP アドレスの値は、以下のファイルにマクロで定義されています。設定方法の詳細は Appendix A 章を参照ください。

(1) スレーブが保持する機器情報

```
1.  #define MAIN_APPL_IP          GOAL_NET_IPV4(192, 168, 0, 100)
2.  #define MAIN_APPL_NM          GOAL_NET_IPV4(255, 255, 255, 0)
3.  #define MAIN_APPL_GW          GOAL_NET_IPV4(0, 0, 0, 0)
```

GSDML ファイルへの設定は不要です。

4.2 Data Model

PROFINET の通信プロセスで用いられる Module と Slot の構成について説明します。

PROFINET の I/O デバイスでは、DAP(Device Access Point)という Ethernet のアクセスポイント情報、及び、1 つ又は複数の Module を各 Slot にマッピングした構成で、通信プロセスが実行されます。

- Module: 入出力データの一塊、1 つ又は複数の Submodule を持つ
- Submodule: Module 内に定義される実際の入出力データの情報(入出力設定、データ長 etc)
- Slot: 通信する際の Module の格納場所、1 つ又は複数の Subslot を持つ
- Subslot: Slot 内に定義される実際の入出力のインタフェース

例として、01_pnio サンプルで使用される Module と Slot の構成を以下に示します。

それぞれの設定は、4.2.3 Output データ、4.2.4 Input データを参照ください。

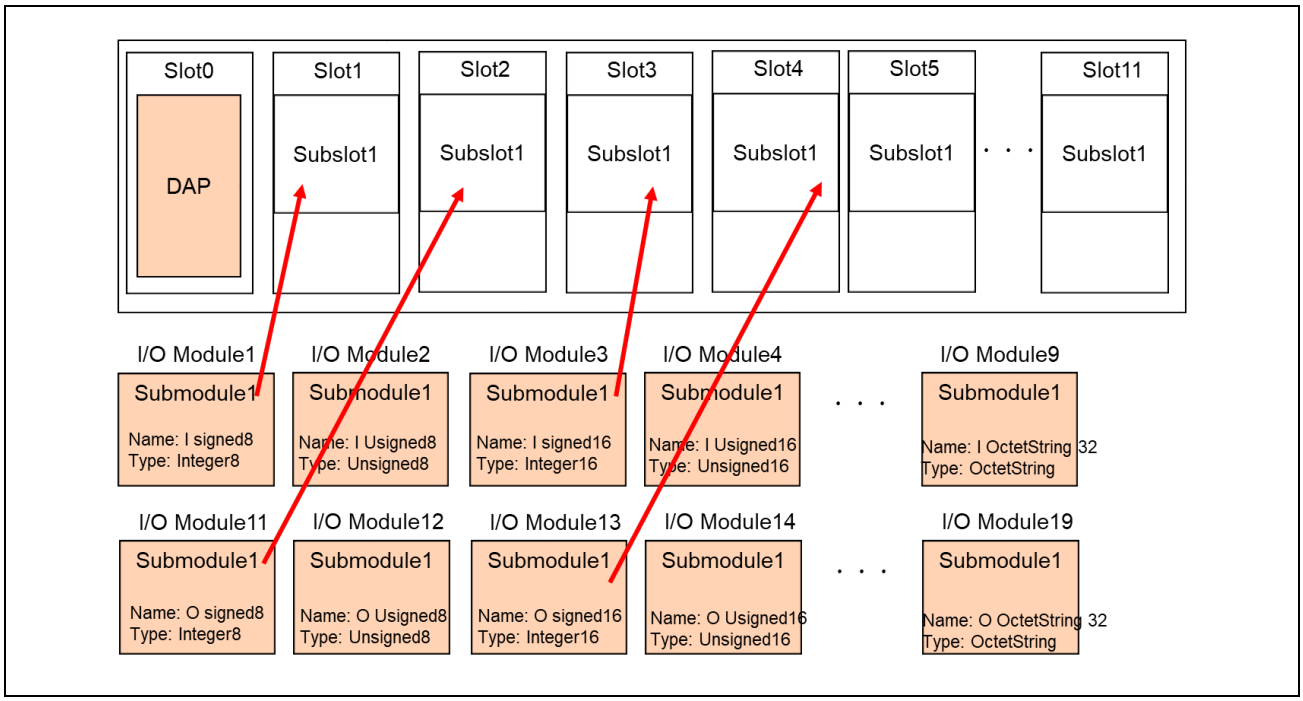

Figure 4-1 Module and Slot configuration [01_pnio]

4.2.1 Slot 最大数

設定値は以下のファイルにマクロで定義されています。

```
1.  #define  APPL_SLOT_COUNT      (12)          /**< maximum number of slots */
```

4.2.2 Subslot 最大数

設定値は以下のファイルにマクロで定義されています。

```
1.  #define  APPL_SUBSLOT_COUNT   (4)           /**< maximum number of subslots */
```

4.2.3 Output データ

マスター機器から受信するデータについて、サンプルソフトの構成を示します。

Table 4-2 Module and Slot Output configuration

sample		Output variable	Module	Submodule	Slot	Subslot	Size
04_pnio_large	01_pnio	dataDm_1	APPL_MOD_11 (201)	APPL_MOD_11_SUB_1 (1)	APPL_SLOT_02 (2)	APPL_SLOT_02_SUB_1 (1)	APPL_SIZE_11_SUB_1_OUT (1)
		dataDm_2	APPL_MOD_13 (203)	APPL_MOD_13_SUB_1 (1)	APPL_SLOT_04 (4)	APPL_SLOT_04_SUB_1 (1)	APPL_SIZE_13_SUB_1_OUT (16)
	-	dataRpc_1	APPL_MOD_15 (205)	APPL_MOD_15_SUB_1 (1)	APPL_SLOT_06 (6)	APPL_SLOT_06_SUB_1 (1)	APPL_SIZE_15_SUB_1_OUT (32)
		dataRpc_2	APPL_MOD_16 (206)	APPL_MOD_16_SUB_1 (1)	APPL_SLOT_08 (8)	APPL_SLOT_08_SUB_1 (1)	APPL_SIZE_16_SUB_1_OUT (32)
		dataRpc_3	APPL_MOD_17 (207)	APPL_MOD_17_SUB_1 (1)	APPL_SLOT_10 (10)	APPL_SLOT_10_SUB_1 (1)	APPL_SIZE_17_SUB_1_OUT (32)

Output データを変更するためには、スレーブが保持する機器情報のプログラムファイル並びにマスターが参照する GSDML ファイルを変更する必要があります。

(1) スレーブが保持する機器情報

1. Output データに使用する Module とその Submodule の型(bool, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32 等)に合わせて、サイズ用のマクロを変更します。

```
1.  #define  APPL_SIZE_11_SUB_1_OUT    (1)      /**< size for module 11 sub 1 output */
    … omit …
2.  #define  APPL_SIZE_13_SUB_1_OUT    (16)     /**< size for module 13 sub 1 output */
    … omit …
```

```
1.  #define  APPL_MOD_11               (0x201)  /**< module 11 */
2.  #define  APPL_MOD_11_SUB_1         (0x01)   /**< submodule for module 11 */
    … omit …
3.  #define  APPL_MOD_13               (0x203)  /**< module 13 */
4.  #define  APPL_MOD_13_SUB_1         (0x01)   /**< submodule for module 13 */
    … omit …
```

2. Output データに使用する Slot と Subslot に合わせて、識別子(ID)用のマクロを変更します。

```
1.  #define  APPL_SLOT_02              (2)      /**< slot 2 */
2.  #define  APPL_SLOT_02_SUB_1        (1)      /**< subslot for slot 2 */
    … omit …
3.  #define  APPL_SLOT_04              (4)      /**< slot 4 */
4.  #define  APPL_SLOT_04_SUB_1        (1)      /**< subslot for slot 4 */
    … omit …
```

3. Output データに使用する Submodule の数に合わせて、Submodule を生成する goal_pnioSubmodNew() の関数コールを変更します。引数には、上記 1. 2. で設定した Module と Submodule のマクロを用います。

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …

5.      /* create submodules */
    … omit …                                                          Module11

6.      res = goal_pnioSubmodNew(pPnio, APPL_MOD_11, APPL_MOD_11_SUB_1,
    GOAL_PNIO_MOD_TYPE_OUTPUT, 0, APPL_SIZE_11_SUB_1_OUT, GOAL_PNIO_FLG_AUTO_GEN);
7.      if (GOAL_RES_ERR(res)) {
8.          goal_logErr("failed to add submodule");
9.          return res;
10.     }
11.
    … omit …                                                          Module13
12.
13.     res = goal_pnioSubmodNew(pPnio, APPL_MOD_13, APPL_MOD_13_SUB_1,
    GOAL_PNIO_MOD_TYPE_OUTPUT, 0, APPL_SIZE_13_SUB_1_OUT, GOAL_PNIO_FLG_AUTO_GEN);
14.     if (GOAL_RES_ERR(res)) {
15.         goal_logErr("failed to add submodule");
16.         return res;
17.     }
18.
    … omit …
```

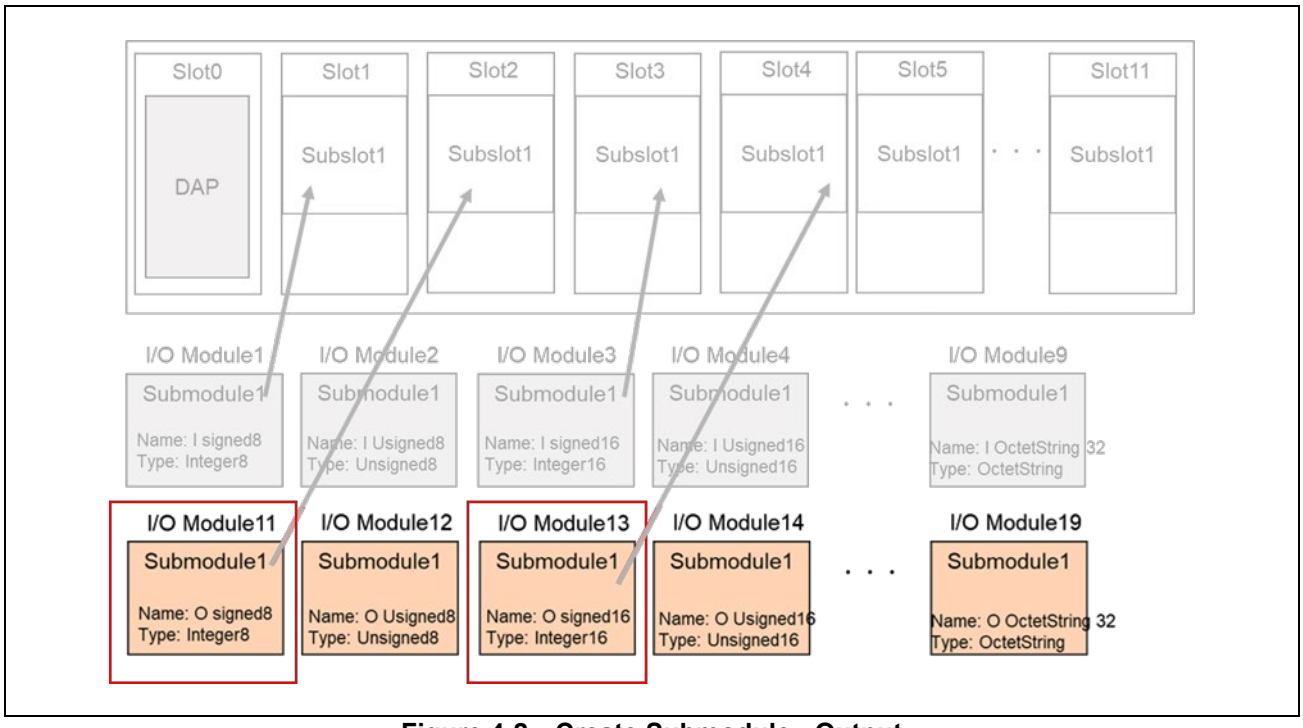

Figure 4-2 Create Submodule - Output

4. Output データに使用する Subslot の数に合わせて、Subslot を生成する goal_pnioSubslotNew()の関数コールを変更します。引数には、上記で設定した Slot と Subslot のマクロを用います。

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …
5.      /* create subslots */
    … omit …
6.
7.      res = goal_pnioSubslotNew(pPnio, APPL_API, APPL_SLOT_02, APPL_SLOT_02_SUB_1,
    GOAL_PNIO_FLG_AUTO_GEN);
8.      if (GOAL_RES_ERR(res)) {
9.          goal_logErr("failed to add subslot");
10.         return res;
11.     }
12.
    … omit …
13.
14.     res = goal_pnioSubslotNew(pPnio, APPL_API, APPL_SLOT_04, APPL_SLOT_04_SUB_1,
    GOAL_PNIO_FLG_AUTO_GEN);
15.     if (GOAL_RES_ERR(res)) {
16.         goal_logErr("failed to add subslot");
17.         return res;
18.     }
19.
    … omit …
```

Slot2(Subslot1)

Slot4(Subslot1)

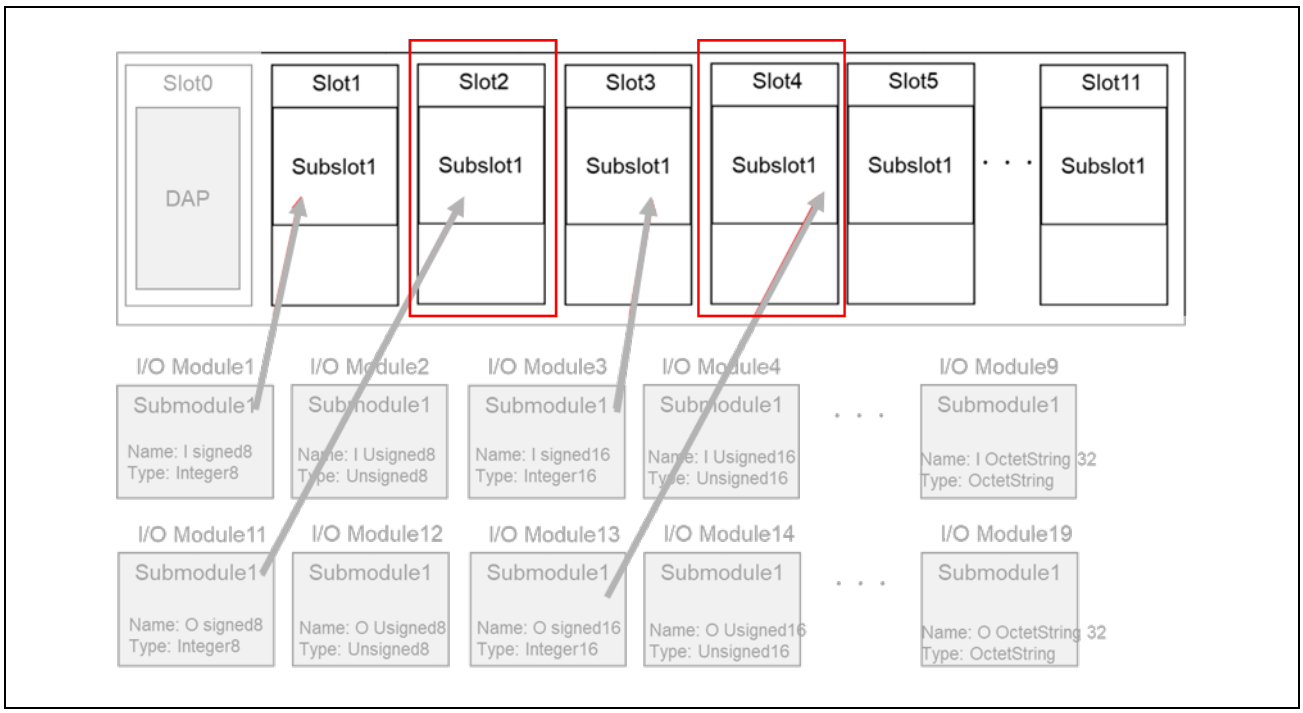

Figure 4-3 Create Subslot - Output

5. Output データに使用する Module と Slot の構成に合わせて、Module と Slot をマッピングする goal_pnioSubmodPlug() の関数コールを変更します*。引数には、上記で設定した Module と Submodule のマクロ、及び、Slot と Subslot のマクロを用います。
 * RPC で送受信する場合は goal_pnioRpcSubmodPlug() を関数コールします

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …
5.      /* plug modules into slots */
                                                    Module11 - Slot2(Subslot1)
    … omit …
6.      res = goal_pnioSubmodPlug(pPnio, APPL_API, APPL_SLOT_02, APPL_SLOT_02_SUB_1,
    APPL_MOD_11, APPL_MOD_11_SUB_1);
7.      if (GOAL_RES_ERR(res)) {
8.          goal_logErr("failed to plug submodule");
9.          return res;
10.     }
11.
    … omit …                                        Module13 – Slot4(Subslot1)
12.     res = goal_pnioSubmodPlug(pPnio, APPL_API, APPL_SLOT_04, APPL_SLOT_04_SUB_1,
    APPL_MOD_13, APPL_MOD_13_SUB_1);
13.     if (GOAL_RES_ERR(res)) {
14.         goal_logErr("failed to plug submodule");
15.         return res;
16.     }
17.
    … omit …
```

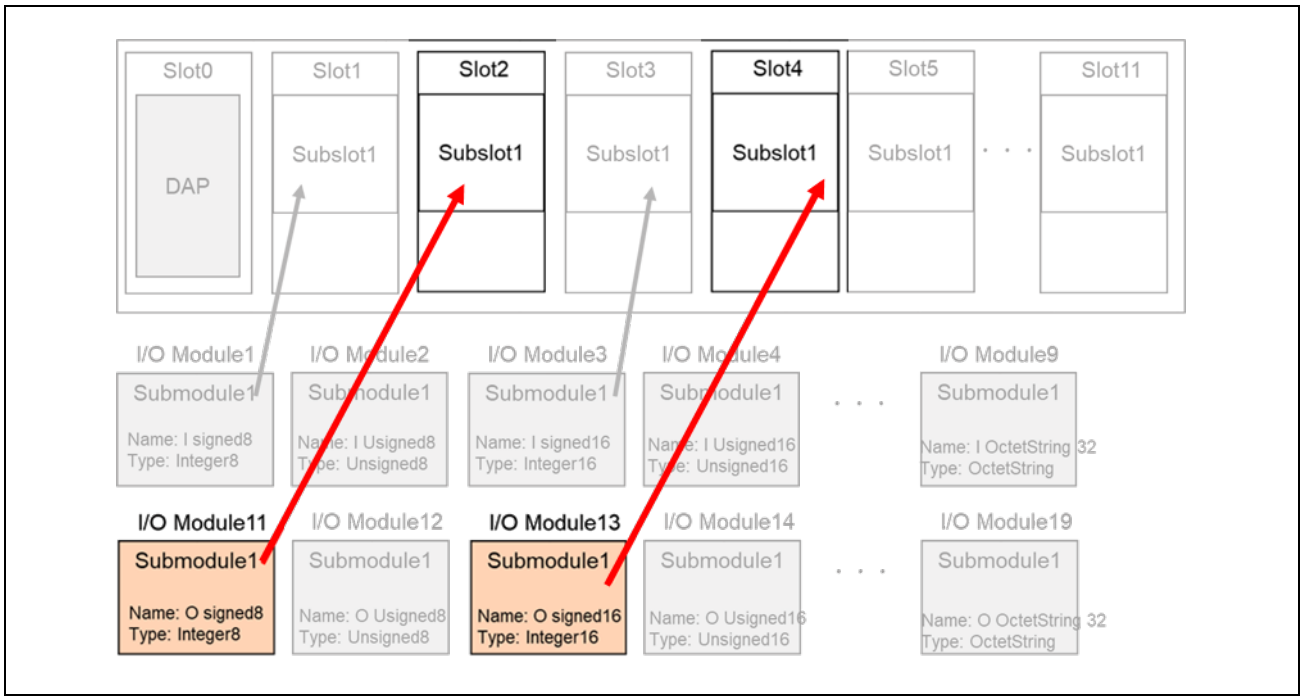

Figure 4-4 Plugin - Output

(2) GSDML ファイル

1. Output データに使用する Module と Slot のマッピングに合わせて、UseableModules の内容を変更します。以下は サンプルソフト 01_pnio での定義例になります。

```
1.  <UseableModules>
      … omit …

2.      <ModuleItemRef ModuleItemTarget="ID_Mod_201" FixedInSlots="2"/>
3.      <ModuleItemRef ModuleItemTarget="ID_Mod_202"/>
4.      <ModuleItemRef ModuleItemTarget="ID_Mod_203" FixedInSlots="4"/>
5.      <ModuleItemRef ModuleItemTarget="ID_Mod_204"/>
6.      <ModuleItemRef ModuleItemTarget="ID_Mod_205"/>
7.      <ModuleItemRef ModuleItemTarget="ID_Mod_206"/>
8.      <ModuleItemRef ModuleItemTarget="ID_Mod_207"/>
9.      <ModuleItemRef ModuleItemTarget="ID_Mod_208"/>
10.     <ModuleItemRef ModuleItemTarget="ID_Mod_209"/>
11.     <ModuleItemRef ModuleItemTarget="ID_Mod_210"/>
12. </UseableModules>
```

2. Output データに使用する Module/Submodule の数、データ型、ID、文字列情報に合わせて、ModuleList の内容を変更します。

```
<ModuleList>
  … omit …

    <ModuleItem ID="ID_Mod_201" ModuleIdentNumber="0x00000201">
        <ModuleInfo>
            <Name TextId="TOK_TextId_Module_ID201"/>
            <InfoText TextId="TOK_InfoTextId_Module_ID201"/>
            <HardwareRelease Value="1.0"/>
            <SoftwareRelease Value="1.0"/>
        </ModuleInfo>
        <VirtualSubmoduleList>
          <VirtualSubmoduleItem ID="ID_Mod_201_Sub_1" SubmoduleIdentNumber="0x0001" API="0"
MayIssueProcessAlarm="false">
            <IOData>
                <Output Consistency="All items consistency">
                    <DataItem DataType="Unsigned8" TextId="TOK_Output_DataItem_Unsigned8"/>
                </Output>
            </IOData>
            <ModuleInfo>
                <Name TextId="TOK_TextId_Module_ID201"/>
                <InfoText TextId="TOK_InfoTextId_Module_ID201"/>
            </ModuleInfo>
          </VirtualSubmoduleItem>
        </VirtualSubmoduleList>
    </ModuleItem>
    <ModuleItem ID="ID_Mod_202" ModuleIdentNumber="0x00000202">
        <ModuleInfo>
… omit …
```

文字列情報は、TextId によって紐づけられ、GSDML 末尾の ExternalTextList にリストで管理されています。

```
<ExternalTextList>
  <PrimaryLanguage>
    <!--english-->
  … omit …

    <!--module name-->
  … omit …
    <Text TextId="TOK_TextId_Module_ID201" Value="O LED Request"/>
    <Text TextId="TOK_TextId_Module_ID202" Value="O Unsigned8"/>
    <Text TextId="TOK_TextId_Module_ID203" Value="O OctedString 16 bytes"/>
    <Text TextId="TOK_TextId_Module_ID204" Value="O Unsigned16"/>
    <Text TextId="TOK_TextId_Module_ID205" Value="O OctedString 32 bytes"/>
  … omit …

    <!--module info name-->
  … omit …
    <Text TextId="TOK_InfoTextId_Module_ID201" Value="Output module (LED request)"/>
    <Text TextId="TOK_InfoTextId_Module_ID202" Value="Output module (Unsigned8)"/>
    <Text TextId="TOK_InfoTextId_Module_ID203" Value="Output module (Octed String 16) "/>
    <Text TextId="TOK_InfoTextId_Module_ID204" Value="Output module (Unsigned16) "/>
    <Text TextId="TOK_InfoTextId_Module_ID205" Value="Output module (Octed String 32) "/>
  … omit …
```

4.2.4 Input データ

マスター機器から受信するデータについて、サンプルソフトの構成を示します。

Input データを変更するためには、スレーブが保持する機器情報のプログラムファイル並びにマスターが参照する GSDML ファイルを変更する必要があります。

Table 4-3 Module and Slot Input configuration

sample		Input variable	Module	Submodule	Slot	Subslot	Size
04_pnio_large	01_pnio	dataDm_1	APPL_MOD_01 (101)	APPL_MOD_01_SUB_1 (1)	APPL_SLOT_01 (1)	APPL_SLOT_01_SUB_1 (1)	APPL_SIZE_01_SUB_1_IN (1)
		dataDm_2	APPL_MOD_03 (103)	APPL_MOD_03_SUB_1 (1)	APPL_SLOT_03 (3)	APPL_SLOT_03_SUB_1 (1)	APPL_SIZE_03_SUB_1_IN (16)
	-	dataRpc_1	APPL_MOD_05 (105)	APPL_MOD_05_SUB_1 (1)	APPL_SLOT_05 (5)	APPL_SLOT_05_SUB_1 (1)	APPL_SIZE_05_SUB_1_IN (32)
		dataRpc_2	APPL_MOD_06 (106)	APPL_MOD_06_SUB_1 (1)	APPL_SLOT_07 (7)	APPL_SLOT_07_SUB_1 (1)	APPL_SIZE_06_SUB_1_IN (32)
		dataRpc_3	APPL_MOD_07 (107)	APPL_MOD_07_SUB_1 (1)	APPL_SLOT_09 (9)	APPL_SLOT_09_SUB_1 (1)	APPL_SIZE_07_SUB_1_IN (32)

(1) スレーブが保持する機器情報

1. Input データに使用する I/O Module とその Submodule の型(bool, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32 等)に合わせて、サイズ用のマクロを変更します。

```
1.  #define  APPL_SIZE_01_SUB_1_IN     (1)     /**< size for module 1 sub 1 input */
    … omit …
2.  #define  APPL_SIZE_03_SUB_1_IN     (16)    **< size for module 3 sub 1 input */
3.
    … omit …
```

```
1.  #define  APPL_MOD_01               (0x101) /**< module 1 */
2.  #define  APPL_MOD_01_SUB_1         (0x01)  /**< submodule for module 1 */
    … omit …
3.  #define  APPL_MOD_03               (0x102) /**< module 3 */
4.  #define  APPL_MOD_03_SUB_1         (0x01)  /**< submodule for module 3 */
    … omit …
```

2. Input データに使用する Slot と Subslot に合わせて、識別子(ID)用のマクロを変更します。

```
1.  #define  APPL_SLOT_01              (1)     **< slot 1 */
2.  #define  APPL_SLOT_01_SUB_1        (1)     /**< subslot for slot 1 */
    … omit …
3.  #define  APPL_SLOT_03              (3)     /**< slot 3 */
4.  #define  APPL_SLOT_03_SUB_1        (1)     /**< subslot for slot 3 */
    … omit …
```

3. Input データに使用する Submodule の数に合わせて、Submodule を生成する goal_pnioSubmodNew() の関数コールを変更します。引数には、上記 1. 2.で設定した Module と Submodule のマクロを用います。

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …

5.      /* create submodules */
    … omit …                                                          Module1

6.      res = goal_pnioSubmodNew(pPnio, APPL_MOD_01, APPL_MOD_01_SUB_1,
    GOAL_PNIO_MOD_TYPE_INPUT, APPL_SIZE_01_SUB_1_IN, 0, GOAL_PNIO_FLG_AUTO_GEN);
7.      if (GOAL_RES_ERR(res)) {
8.          goal_logErr("failed to add submodule");
9.          return res;
10.     }
11.
    … omit …                                                          Module3
12.
13.     res = goal_pnioSubmodNew(pPnio, APPL_MOD_03, APPL_MOD_03_SUB_1,
    GOAL_PNIO_MOD_TYPE_INPUT, APPL_SIZE_03_SUB_1_IN, 0, GOAL_PNIO_FLG_AUTO_GEN);
14.     if (GOAL_RES_ERR(res)) {
15.         goal_logErr("failed to add submodule");
16.         return res;
17.     }
18.
    … omit …
```

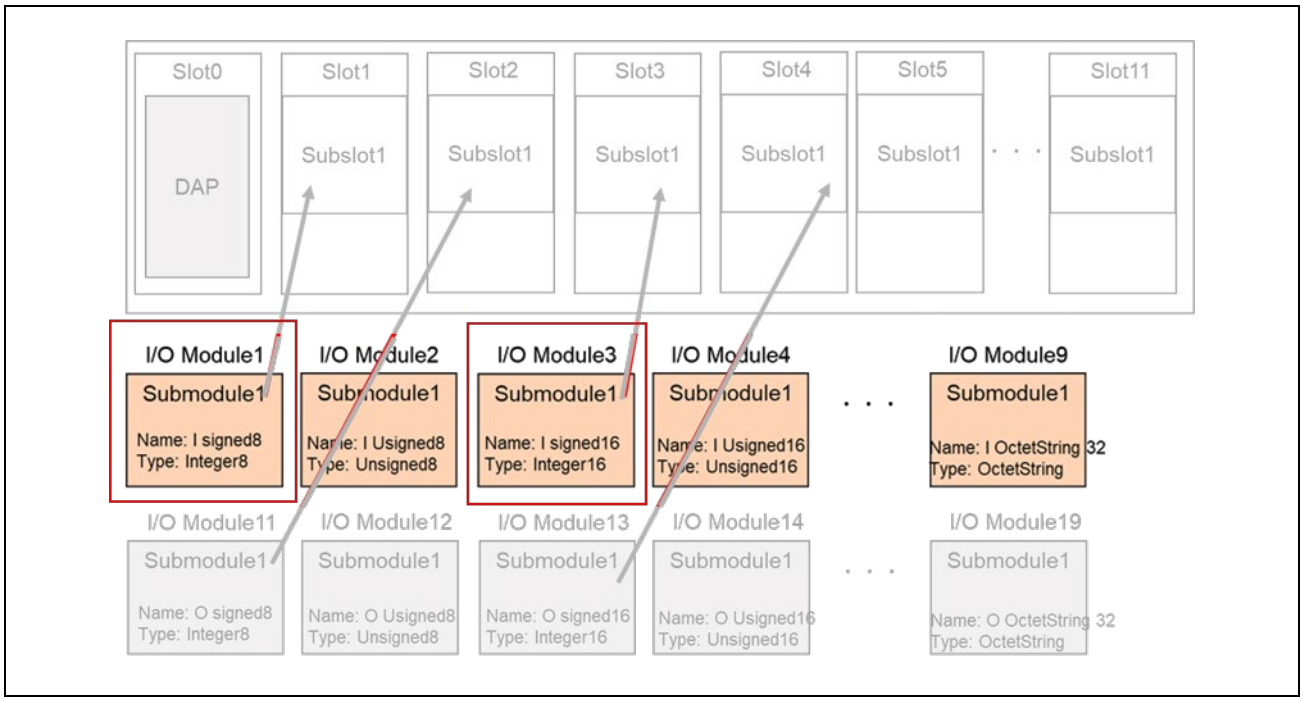

Figure 4-5 Create Submodule - Input

4. Input データに使用する Subslot の数に合わせて、Subslot を生成する goal_pnioSubslotNew()の関数コールを変更します。引数には、上記で設定した Slot と Subslot のマクロを用います。

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …
5.      /* create subslots */
    … omit …                                                    Slot1(Subslot1)
6.
7.      res = goal_pnioSubslotNew(pPnio, APPL_API, APPL_SLOT_01, APPL_SLOT_01_SUB_1,
    GOAL_PNIO_FLG_AUTO_GEN);
8.      if (GOAL_RES_ERR(res)) {
9.          goal_logErr("failed to add subslot");
10.         return res;
11.     }
12.
    … omit …                                                    Slot3(Subslot1)
13.
14.     res = goal_pnioSubslotNew(pPnio, APPL_API, APPL_SLOT_03, APPL_SLOT_03_SUB_1,
    GOAL_PNIO_FLG_AUTO_GEN);
15.     if (GOAL_RES_ERR(res)) {
16.         goal_logErr("failed to add subslot");
17.         return res;
18.     }
19.
    … omit …
```

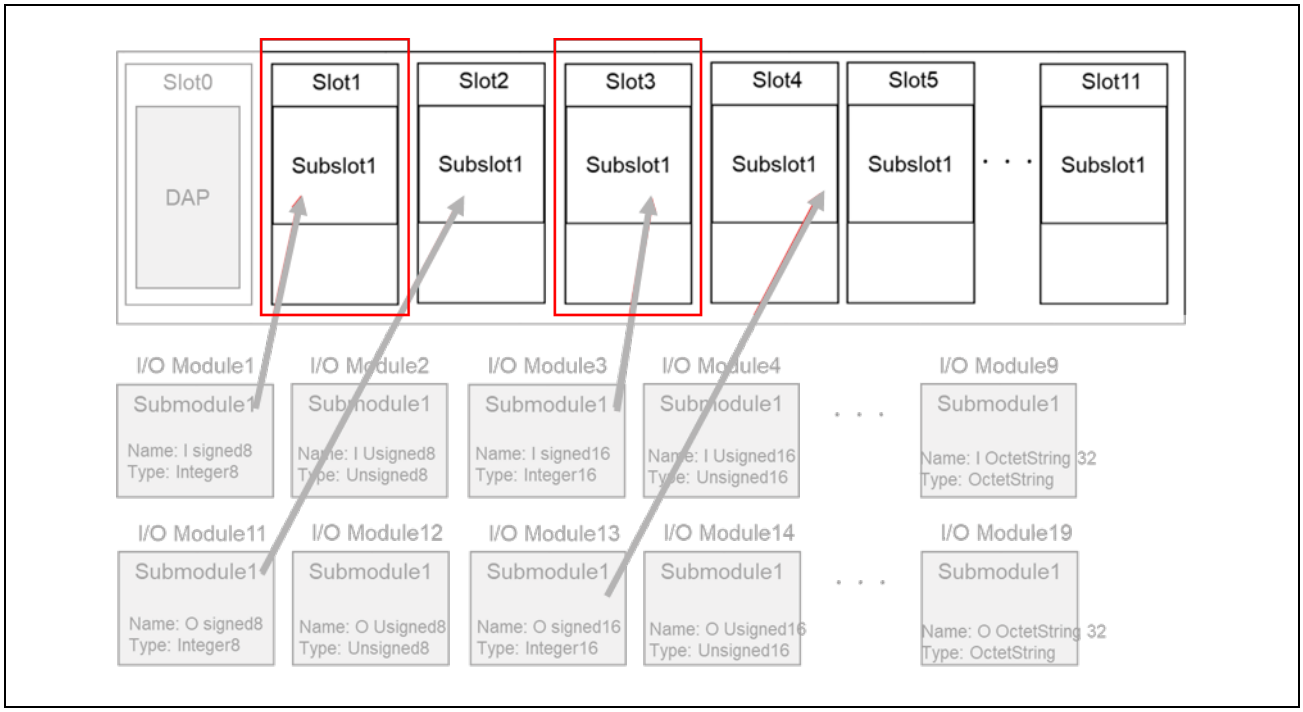

Figure 4-6 Create Subslot - Input

5. Input データに使用する Module と Slot の構成に合わせて、Module と Slot をマッピングする goal_pnioSubmodPlug()の関数コールを変更します*。引数には、上記で設定した Module と Submodule のマクロ、及び、Slot と Subslot のマクロを用います。
* RPC で送受信する場合は goal_pnioRpcSubmodPlug() を関数コールします

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …
5.
6.      /* plug modules into slots */
7.
    … omit …                                          Module1 – Slot1(Subslot1)
8.
9.      res = goal_pnioSubmodPlug(pPnio, APPL_API, APPL_SLOT_01, APPL_SLOT_01_SUB_1,
    APPL_MOD_01, APPL_MOD_01_SUB_1);
10.     if (GOAL_RES_ERR(res)) {
11.         goal_logErr("failed to plug submodule");
12.         return res;
13.     }
14.
    … omit …                                          Module3 – Slot3(Subslot1)
15.
16.     res = goal_pnioSubmodPlug(pPnio, APPL_API, APPL_SLOT_03, APPL_SLOT_03_SUB_1,
    APPL_MOD_03, APPL_MOD_03_SUB_1);
17.     if (GOAL_RES_ERR(res)) {
18.         goal_logErr("failed to plug submodule");
19.         return res;
20.     }
21.
    … omit …
```

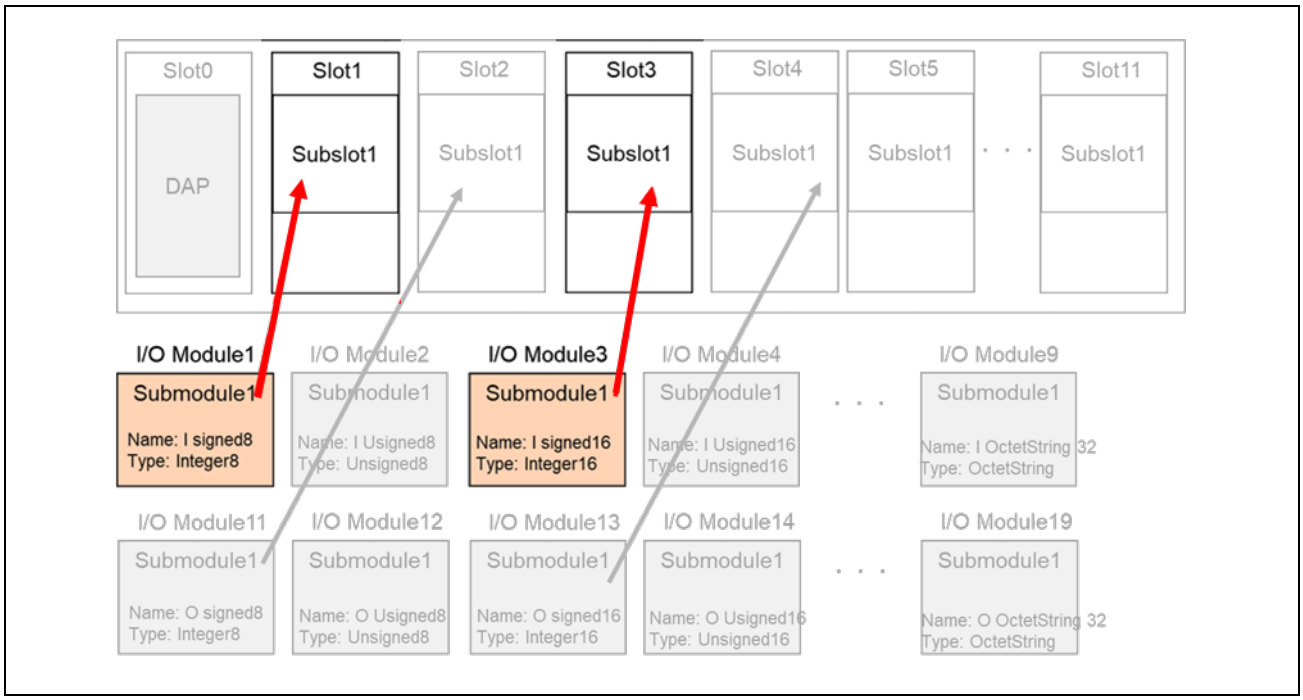

Figure 4-7 Plugin - Input

(2) **GSDML ファイル**

1. Input データに使用する Module と Slot のマッピングに合わせて、UseableModules の内容を変更します。以下は サンプルソフト 01_pnio での定義例になります。

```
1.    <UseableModules>
2.        <ModuleItemRef ModuleItemTarget="ID_Mod_101" FixedInSlots="1"/>
3.        <ModuleItemRef ModuleItemTarget="ID_Mod_102"/>
4.        <ModuleItemRef ModuleItemTarget="ID_Mod_103" FixedInSlots="3"/>
5.        <ModuleItemRef ModuleItemTarget="ID_Mod_104"/>
6.        <ModuleItemRef ModuleItemTarget="ID_Mod_105"/>
7.        <ModuleItemRef ModuleItemTarget="ID_Mod_106"/>
8.        <ModuleItemRef ModuleItemTarget="ID_Mod_107"/>
9.        <ModuleItemRef ModuleItemTarget="ID_Mod_108"/>
10.       <ModuleItemRef ModuleItemTarget="ID_Mod_109"/>
11.       <ModuleItemRef ModuleItemTarget="ID_Mod_110"/>

      … omit …
12.   </UseableModules>
```

2. Input データに使用する Module/Submodule の数、データ型、ID、文字列情報に合わせて、以下の ModuleList の内容を変更します。以下は **dataDm_1** の定義例になります。

```
<ModuleList>
    <ModuleItem ID="ID_Mod_101" ModuleIdentNumber="0x00000101">
        <ModuleInfo>
            <Name TextId="TOK_TextId_Module_ID101"/>
            <InfoText TextId="TOK_InfoTextId_Module_ID101"/>
            <HardwareRelease Value="1.0"/>
            <SoftwareRelease Value="1.0"/>
        </ModuleInfo>
    <VirtualSubmoduleList>
        <VirtualSubmoduleItem ID="ID_Mod_101_Sub_1" SubmoduleIdentNumber="0x0001" API="0"
MayIssueProcessAlarm="false">
            <IOData>
                <Output Consistency="All items consistency">
                    <DataItem DataType="Unsigned8" TextId="TOK_Input_DataItem_Unsigned8"/>
                </Output>
            </IOData>
            <ModuleInfo>
                <Name TextId="TOK_TextId_Module_ID101”/>
                <InfoText TextId=”TOK_InfoTextId_Module_ID101"/>
            </ModuleInfo>
        </VirtualSubmoduleItem>
    </VirtualSubmoduleList>
    </ModuleItem>
    <ModuleItem ID="ID_Mod_102" ModuleIdentNumber="0x00000102">
        <ModuleInfo>
    … omit …
```

文字列情報は、TextId によって紐づけられ、GSDML 末尾の ExternalTextList にリストで管理されています。

```
<ExternalTextList>
  <PrimaryLanguage>
    <!--english-->
… omit …

    <!--module name-->
    <Text TextId="TOK_TextId_Module_ID101" Value="I Switch Status"/>
    <Text TextId="TOK_TextId_Module_ID102" Value="I Unsigned8"/>
    <Text TextId="TOK_TextId_Module_ID103" Value="I OctedString 16 bytes"/>
    <Text TextId="TOK_TextId_Module_ID104" Value="I Unsigned16"/>
    <Text TextId="TOK_TextId_Module_ID105" Value="I OctedString 32 bytes"/>
… omit …

    <!--module info name-->
    <Text TextId="TOK_InfoTextId_Module_ID101" Value="Input module (Switch status)"/>
    <Text TextId="TOK_InfoTextId_Module_ID102" Value="Input module (Unsigned8)"/>
    <Text TextId="TOK_InfoTextId_Module_ID103" Value="Input module (Octed String 16) "/>
    <Text TextId="TOK_InfoTextId_Module_ID104" Value="Input module (Unsigned16) "/>
    <Text TextId="TOK_InfoTextId_Module_ID105" Value="Input module (Octed String 32) "/>
  … omit …
```

4.3 Output/Input ユーザーアプリケーションデータ操作

データ送受信に用いる API とデータ用変数を示します。

サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- Remote-IO (LED/Switch)：評価ボードの LED 点灯制御および Switch 状態を送受信
- Mirror：マスターから受信したデータをミラーバック送信

Table 4-4 API and Sample application data variable

sample		Sample app.	Data variable	Purpose	API	reference
04_pnio_large	01_pnio	get LED Data [#3]	dataDm_1 (Slot 2)	Reception	goal_pnioDataOutputGet()	4.2.3 Output データ
		get Mirror Data [#1]	dataDm_2 (Slot 4)			
		set Switch Data [#4]	dataDm_1 (Slot 1)	Transmission	goal_pnioDataInputSet()	4.2.4 Input データ
		set Mirror Data [#2]	dataDm_2 (Slot 3)			
	-	get Mirror Data_1	dataRpc_1 (Slot 6)	Reception	goal_pnioDataOutputGet()	4.2.3 Output データ
		get Mirror Data_2	dataRpc_2 (Slot 8)			
		get Mirror Data_3	dataRpc_3 (Slot 10)			
		set Mirror Data_1	dataRpc_1 (Slot 5)	Transmission	goal_pnioDataInputSet()	4.2.4 Input データ
		set Mirror Data_2	dataRpc_2 (Slot 7)			
		set Mirror Data_3	dataRpc_3 (Slot 9)			

以下は サンプルソフト 01_pnio でのアプリケーション実装例になります。

```
1.  void appl_loop(
2.      void
3.  )
4.  {
    … omit …

5.      if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >= APPL_DM_TIMEOUT_TRIG_VAL))
    {
6.
7.  /* read data from output module */
8.    res = goal_pnioDataOutputGet(pPnio, APPL_API, APPL_SLOT_04, APPL_SLOT_04_SUB_1, dataDm_2,
    APPL_SIZE_13_SUB_1_OUT, &iops);
9.    if (GOAL_RES_ERR(res)) {
10.       return;
11.   }
12.   /* copy data to input module */
13.   res = goal_pnioDataInputSet(pPnio, APPL_API, APPL_SLOT_03, APPL_SLOT_03_SUB_1, dataDm_2,
    APPL_SIZE_3_SUB_1_IN , GOAL_PNIO_IOXS_GOOD);
14.   if (GOAL_RES_ERR(res)) {
15.       return;
16.   }
17.
18.   /* read data from output module */
19.   res = goal_pnioDataOutputGet(pPnio, APPL_API, APPL_SLOT_02, APPL_SLOT_02_SUB_1, dataDm_1,
    APPL_SIZE_11_SUB_1_OUT, &iops);
20.   if (GOAL_RES_ERR(res)) {
21.       return;
22.   }
23.
24.   /* sample application: set LED */
25.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE1, (dataDm_1[0] & 0x01) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
26.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE2, (dataDm_1[0] & 0x02) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
27.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE3, (dataDm_1[0] & 0x04) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
28.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE4, (dataDm_1[0] & 0x08) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
29.
30. /* sample application: get switch state */
31. dataDm_1[0] = 0;
32. for (uint8_t i = 0; i < 4; i++) {
33.     dataDm_1[0] |= (plat_remoteIoSwGet(i) << i);
34. }
35.
36. /* copy data to input module */
37.   res = goal_pnioDataInputSet(pPnio, APPL_API, APPL_SLOT_01, APPL_SLOT_01_SUB_1, dataDm_1,
    APPL_SIZE_1_SUB_1_IN, GOAL_PNIO_IOXS_GOOD);
38.   if (GOAL_RES_ERR(res)) {
39.       return;
40.   }
41.
    … omit …
42.
43. }
```

[#1] get Mirror Data

[#2] set Mirror Data

[#3] get LED Data

[#4] set Switch Data

アプリケーションのデータ更新は uGOAL のループ周期で呼ばれる appl_loop()関数の中で制御します。データ更新周期は、データ転送方式により異なります。

サンプルソフトでは、転送方式により Table 4-5 の更新周期となっています。

Table 4-5 sample soft update cycle

Data Transfer	Update cycle Define	Sample soft default [ms]	Sample	
			01	04
Cyclic	APPL_DM_TIMEOUT_TRIG_VAL	1	✓	✓
RPC	APPL_RPC_TIMEOUT_TRIG_VAL	310	-	✓

RPC を利用したデータ転送では、PROFINET の最大送受信データサイズ 69byte 以上のプロセスデータが送受信可能となりますが、アプリケーションの更新周期が増加します。Table 4-6 を参考に、扱うデータサイズにより更新周期を指定してください。

Table 4-6 Data size and update cycle comparison

Data Transfer	Data size [byte]	Average cycle [ms]
Cyclic	69	1
RPC	128	40
	256	70
	512	125
	1024	230
	1434	310

4.4 MRP

MRP(Media Redundancy Protocol)機能はデフォルト有効になっています。機能を無効にする場合は、以下の変更を行ってください。

(1) スレーブが保持する機器情報

goal_pnioCfgFlgMrpCfgSet()の引数を GOAL_TRUE から GOAL_FALSE に変更します。

```
1.  GOAL_STATUS_T appl_setup(
2.      void
3.  )
4.  {
    … omit …
5.
6.      /* enable MRP */
7.      res = goal_pnioCfgFlgMrpCfgSet(GOAL_TRUE);
8.      if (GOAL_RES_ERR(res)) {
9.          goal_logErr("failed to enable MRP");
10.         return res;
11.     }
12.
    … omit …
13. }
```

(2) GSDML ファイル

MediaRedundancy …から始まる 1 行を削除またはコメントアウトします。

```
1.  <SystemDefinedSubmoduleList>
      … omit …

2.      </ApplicationRelations>
3.        <MediaRedundancy AdditionalProtocolsSupported="true" SupportedRole="Client"/>
4.    </InterfaceSubmoduleItem>
```

5. EtherNet/IP

本サンプルにて、ユーザが設定するプロファイル情報を下記に示します。

アダプタのプロファイル情報は、EDS (Electric Data Sheet)ファイルでデバイスに関する機器情報をスキャナ機器に提供します。そのため、アダプタが保持する機器情報と EDS ファイルの情報は一致していなければいけません。

ここでは、アダプタが保持する機器情報と EDS ファイルのプロファイル設定について説明します。

スレーブが保持する機器情報: appl\02_eip\goal_appl.h

EDS ファイル: appl\02_eip\eds\eip_goal_renesas.eds

EDS ファイルに記述される機器情報の一部

- ベンダープロファイル
- データ構成と意味
- 利用可能なデータ通信の種類
- コンフィグレーションパラメータ

EDS ファイルはテキストエディタを使用して作成、変更することが可能ですが、Open DeviceNet Vendor Association, Inc. (ODVA) が提供する EZ-EDS を利用することで変更を容易に実施できます。

Order EZ-EDS | ODVA Network Specifications

以降で示す API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

5.1 Device Identity

ベンダ ID やベンダ名、プロダクトコード等について説明します。

これらのアダプタが保持する機器情報はプログラムファイル内にマクロで定義されています。

Table 5-1 EtherNet/IP Device Identify

Item	Macro	Default Value [02_eip]
Vendor ID	APPL_EIP_VENDOR_ID	1105
Vendor Name	APPL_EIP_VENDOR	Renesas Electronics
Product Code	APPL_EIP_PRODUCT_CODE	768
Product Name	APPL_EIP_PRODUCT	R-IN32M3 Module
IP address	MAIN_APPL_IP	192.168.0.100

5.1.1 ベンダ ID

ベンダ ID は ODVA によって割り当てられます。デフォルト値は 1105 (ルネサスエレクトロニクス)です。

(1) アダプタが保持する機器情報

”APPL_EIP_VENDOR_ID” の値を変更します。

```
1.  #ifndef APPL_EIP_VENDOR_ID
2.  # define   APPL_EIP_VENDOR_ID    (1105)              /**< vendor id */
3.  #endif
```

(2) EDS ファイル

“VendorCode” の値を変更します。

```
1.  [Device]
2.          VendCode = 1105;
3.          VendName = "Renesas Electronics";
4.          ProdType = 43;
5.          ProdTypeStr = "Generic Device";
6.          ProdCode = 768;
7.          MajRev = 1;
8.          MinRev = 1;
9.          ProdName = "R-IN32M3 Module";
```

< EZ-EDS 表示>

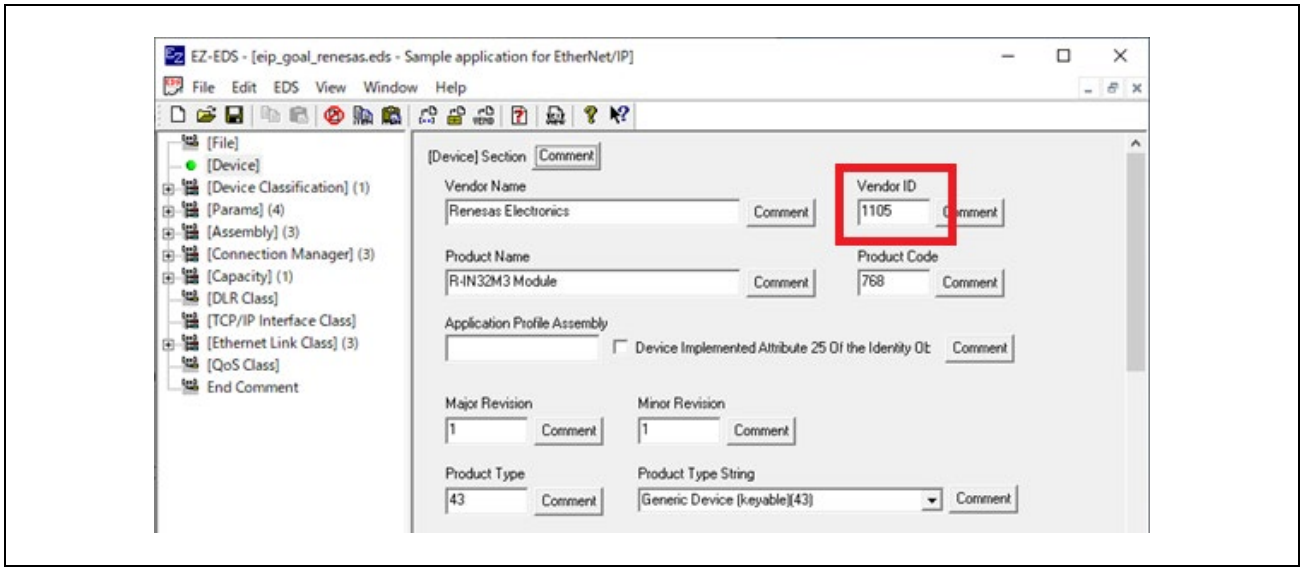

Figure 5-1 Vendor ID [EZ-EDS]

5.1.2 ベンダ名

ベンダ名はベンダ ID に紐づいている必要があります。デフォルト値は”Renesas Electronics” (ルネサスエレクトロニクス)です。

(1) アダプタが保持する機器情報

”APPL_EIP_VENDOR” の値を変更します。

```
#ifndef APPL_EIP_VENDOR
# define   APPL_EIP_VENDOR    Renesas Electronics      /**< vendor name */
#endif
```

(2) EDS ファイル

“VendorName” の値を変更します。

```
[Device]
        VendCode = 1105;
        VendName = "Renesas Electronics";
        ProdType = 43;
        ProdTypeStr = "Generic Device";
        ProdCode = 768;
        MajRev = 1;
        MinRev = 1;
        ProdName = "R-IN32M3 Module";
```

< EZ-EDS 表示 >

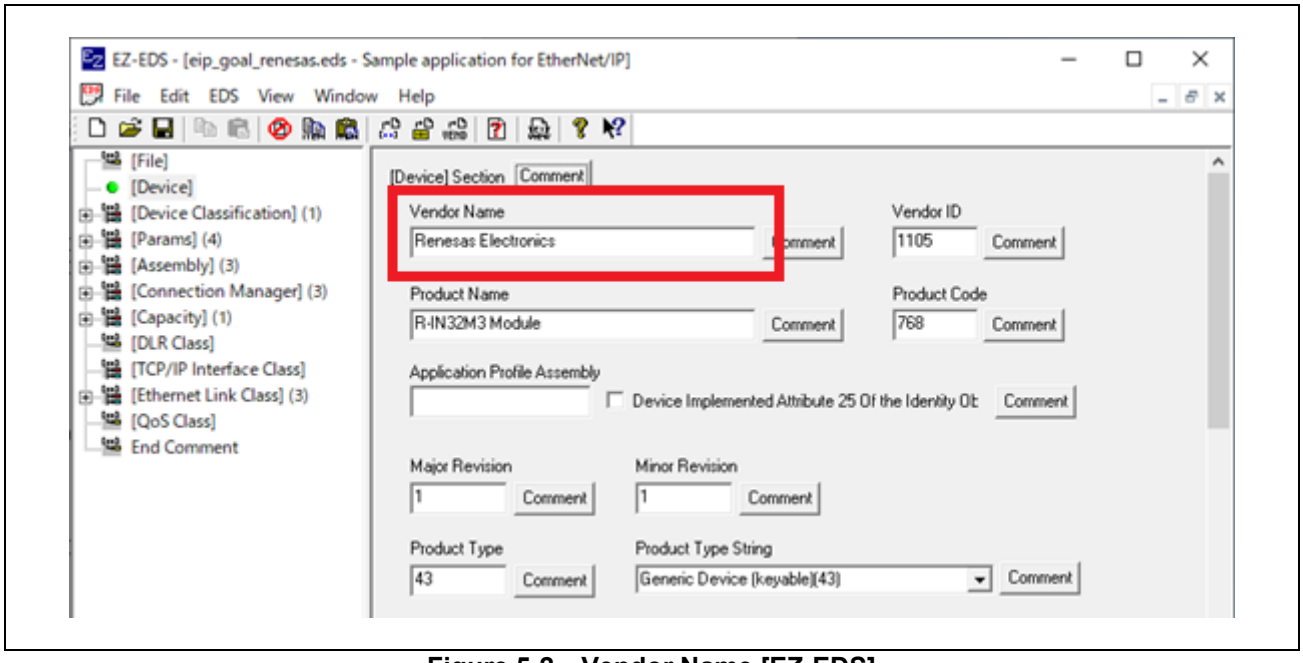

Figure 5-2 Vendor Name [EZ-EDS]

5.1.3 プロダクトコード

任意の製品コードを設定します。デフォルト値は 768 です。

(1) アダプタが保持する機器情報

” APPL_EIP_PRODUCT_CODE” の値を変更します。

```
#ifndef APPL_EIP_PRODUCT_CODE
# define   APPL_EIP_PRODUCT_CODE   (768)                /**< product code */
#endif
```

(2) EDS ファイル

” ProdCode” の値を変更します。

```
[Device]
        VendCode = 1105;
        VendName = "Renesas Electronics";
        ProdType = 43;
        ProdTypeStr = "Generic Device";
        ProdCode = 768;
        MajRev = 1;
        MinRev = 1;
        ProdName = "R-IN32M3 Module";
```

< EZ-EDS 表示 >

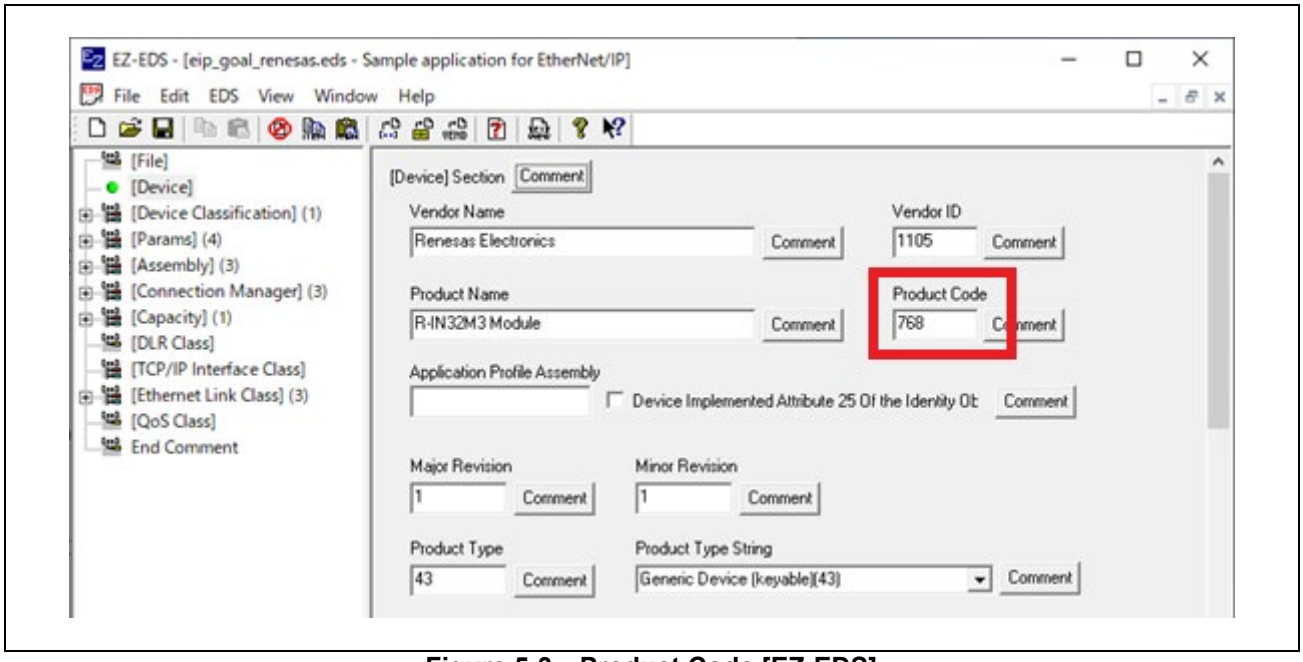

Figure 5-3 Product Code [EZ-EDS]

5.1.4 プロダクト名

任意の製品名を設定します。デフォルト値は”R-IN32M3_Module”です。

(1) アダプタが保持する機器情報

” APPL_EIP_PRODUCT” の値を変更します。

```
#ifndef APPL_EIP_PRODUCT
# define   APPL_EIP_PRODUCT   R-IN32M3_Module          /**< product name */
#endif
```

(2) EDS ファイル

” ProdName” の値を変更します。

```
[Device]
        VendCode = 1105;
        VendName = "Renesas Electronics";
        ProdType = 43;
        ProdTypeStr = "Generic Device";
        ProdCode = 768;
        MajRev = 1;
        MinRev = 1;
        ProdName = "R-IN32M3 Module";
```

< EZ-EDS 表示 >

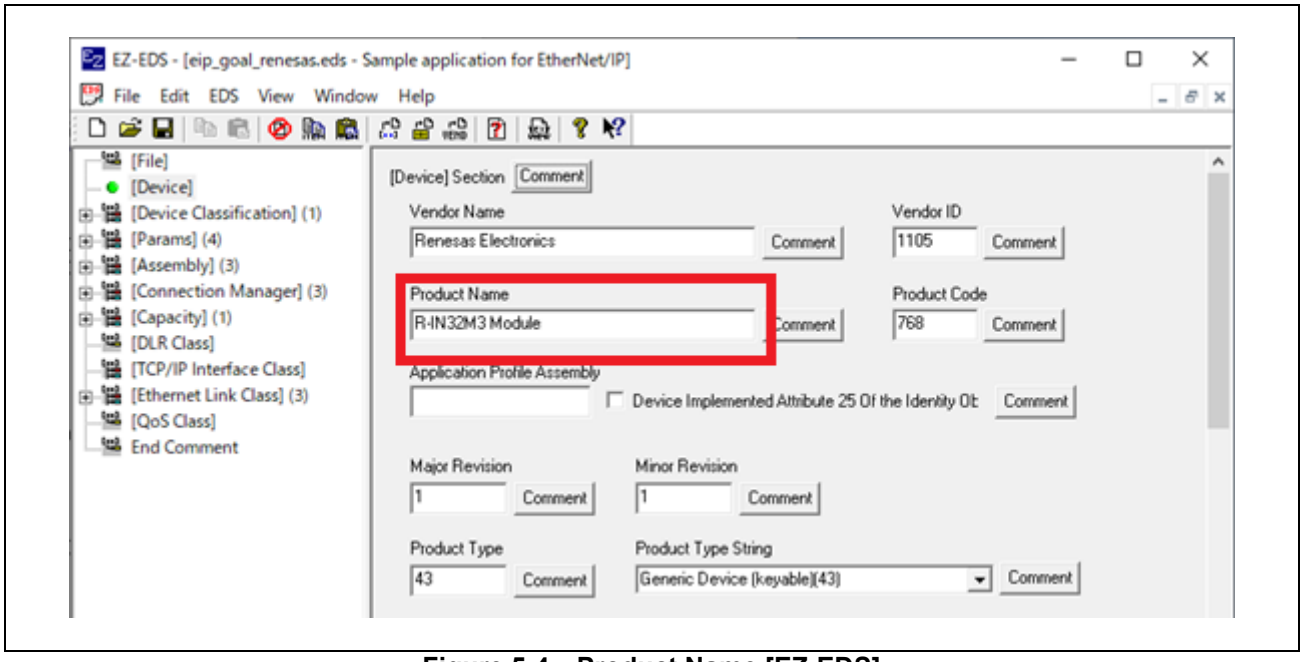

Figure 5-4 Product Name [EZ-EDS]

5.1.5 IP アドレス

評価用途のために任意の固定 IP アドレスの設定が可能です。

IP アドレスの値は、以下のファイルにマクロで定義されています。設定方法の詳細は Appendix A を参照ください。

(1) アダプタが保持する機器情報

```
1.  #define  MAIN_APPL_IP          GOAL_NET_IPV4(192, 168, 0, 100)
2.  #define  MAIN_APPL_NM          GOAL_NET_IPV4(255, 255, 255, 0)
3.  #define  MAIN_APPL_GW          GOAL_NET_IPV4(0, 0, 0, 0)
```

EDS ファイルへの設定は不要です。

5.2 Data Model

EtherNet/IP の通信プロセスで用いられる Input/Output データやサイズについて説明します。

5.2.1 Output データ

スキャナからアダプタ(本モジュール) が受信するデータを定義します。

Table 5-2 Output configuration

sample		variable	Assembly ID	size
05_eip_large	02_eip	ledRequest	GOAL_APP_ASM_ID_OUTPUT_DM1 (150)	GOAL_APP_ASM_SIZE_OUTPUT_DM1 (1)
		outData[]	GOAL_APP_ASM_ID_OUTPUT_DM2 (151)	GOAL_APP_ASM_SIZE_OUTPUT_DM2 (16)
	'	rpcData[]	GOAL_APP_ASM_ID_OUTPUT_RPC1 (152)	GOAL_APP_ASM_SIZE_OUTPUT_RPC (32)
		rpcData[]	GOAL_APP_ASM_ID_OUTPUT_RPC2 (153)	GOAL_APP_ASM_SIZE_OUTPUT_RPC (32)
		rpcData[]	GOAL_APP_ASM_ID_OUTPUT_RPC3 (154)	GOAL_APP_ASM_SIZE_OUTPUT_RPC (32)

Output データの構成やサイズを変更するためには、プログラムファイルと EDS ファイルを変更する必要があります。

(1) **アダプタが保持する機器情報**

” GOAL_APP_ASM_SIZE_OUTPUT_**” の値を変更します。

```
1.  #define  GOAL_APP_ASM_SIZE_OUTPUT_DM1      (1)          /**< Output Assembly Data */
```

(2) **EDS ファイル**

” Assem150” の値を変更します。

```
1.  [Assembly]
    … omit …

2.          Assem150 =
3.                  "Output Assembly",
4.                  "",
5.                  1,
6.                  0x0000,
7.                  ,,
8.                  8,Param150,
    … omit …
```

< EZ-EDS 表示 >

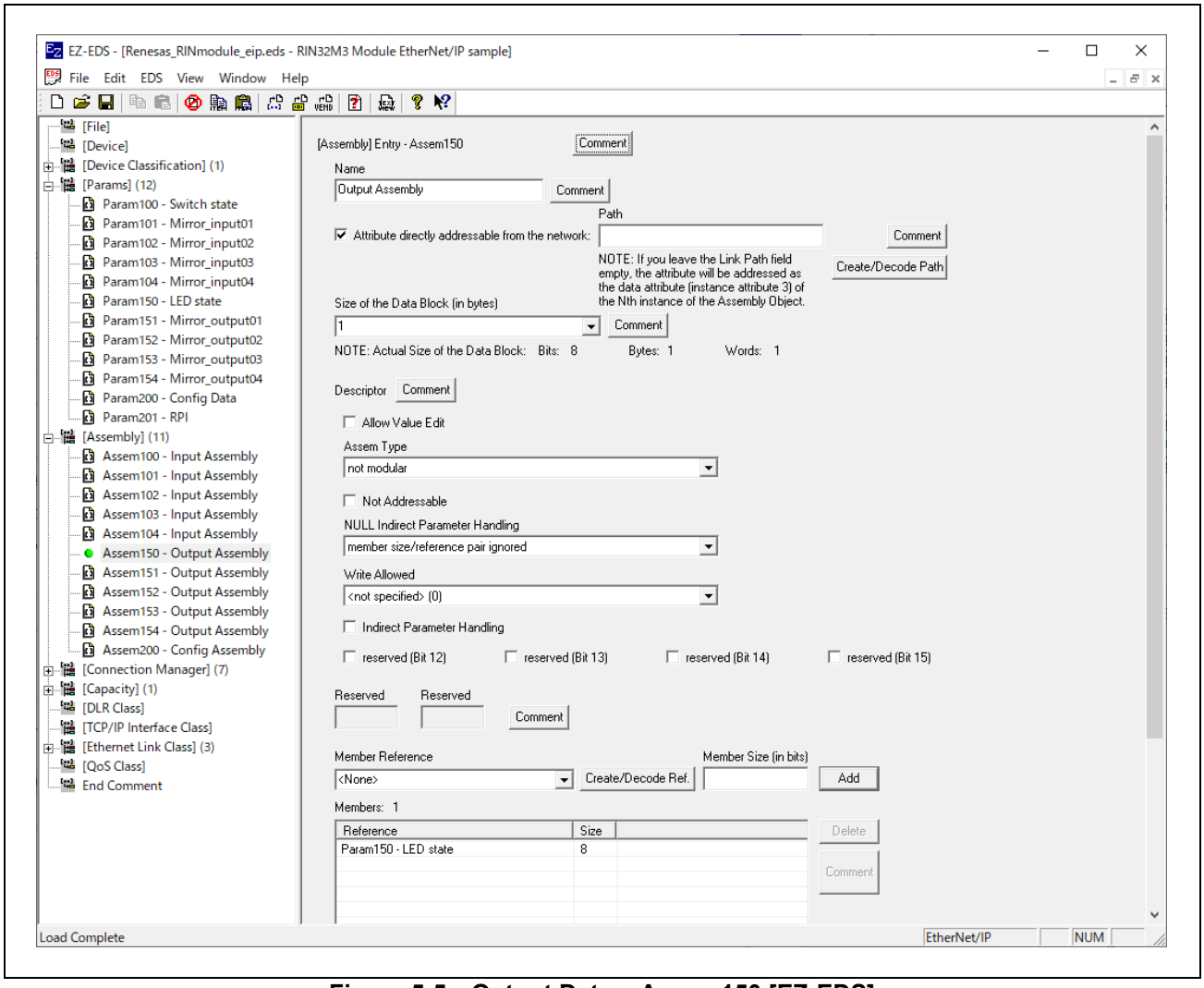

Figure 5-5 Output Data – Assem150 [EZ-EDS]

5.2.2 Input データ

アダプタ(本モジュール)がスキャナへ送信するデータを定義します。

Table 5-3 Input configuration

sample		variable	Assembly ID	size
05_eip_large	02_eip	ledRequest	GOAL_APP_ASM_ID_INPUT_DM1 (100)	GOAL_APP_ASM_SIZE_INPUT_DM1 (1)
		outData[]	GOAL_APP_ASM_ID_ INPUT_DM2 (101)	GOAL_APP_ASM_SIZE_ INPUT_DM2 (16)
	'	rpcData[]	GOAL_APP_ASM_ID_ INPUT_RPC1 (102)	GOAL_APP_ASM_SIZE_ INPUT_RPC (32)
		rpcData[]	GOAL_APP_ASM_ID_ INPUT_RPC2 (103)	GOAL_APP_ASM_SIZE_ INPUT_RPC (32)
		rpcData[]	GOAL_APP_ASM_ID_ INPUT_RPC3 (104)	GOAL_APP_ASM_SIZE_ INPUT_RPC (32)

Input データの構成やサイズを変更するためには、プログラムファイルと EDS ファイルを変更する必要があります。

(1) アダプタが保持する機器情報

” GOAL_APP_ASM_SIZE_INPUT_**” の値を変更します。

```
1.  #define  GOAL_APP_ASM_SIZE_INPUT_DM1      (1)      /**< Input Assembly Data */
```

(2) EDS ファイル

” Assem100” の値を変更します。

```
1.  [Assembly]
2.          Object_Name = "Assembly Object";
3.          Object_Class_Code = 0x04;
4.          Number_Of_Static_Instances = 6;
5.          Assem100 =
6.                  "Input Assembly",
7.                  "",
8.                  1,
9.                  0x0000,
10.                 ,,
11.                 8,Param100,
    … omit …
```

< EZ-EDS 表示 >

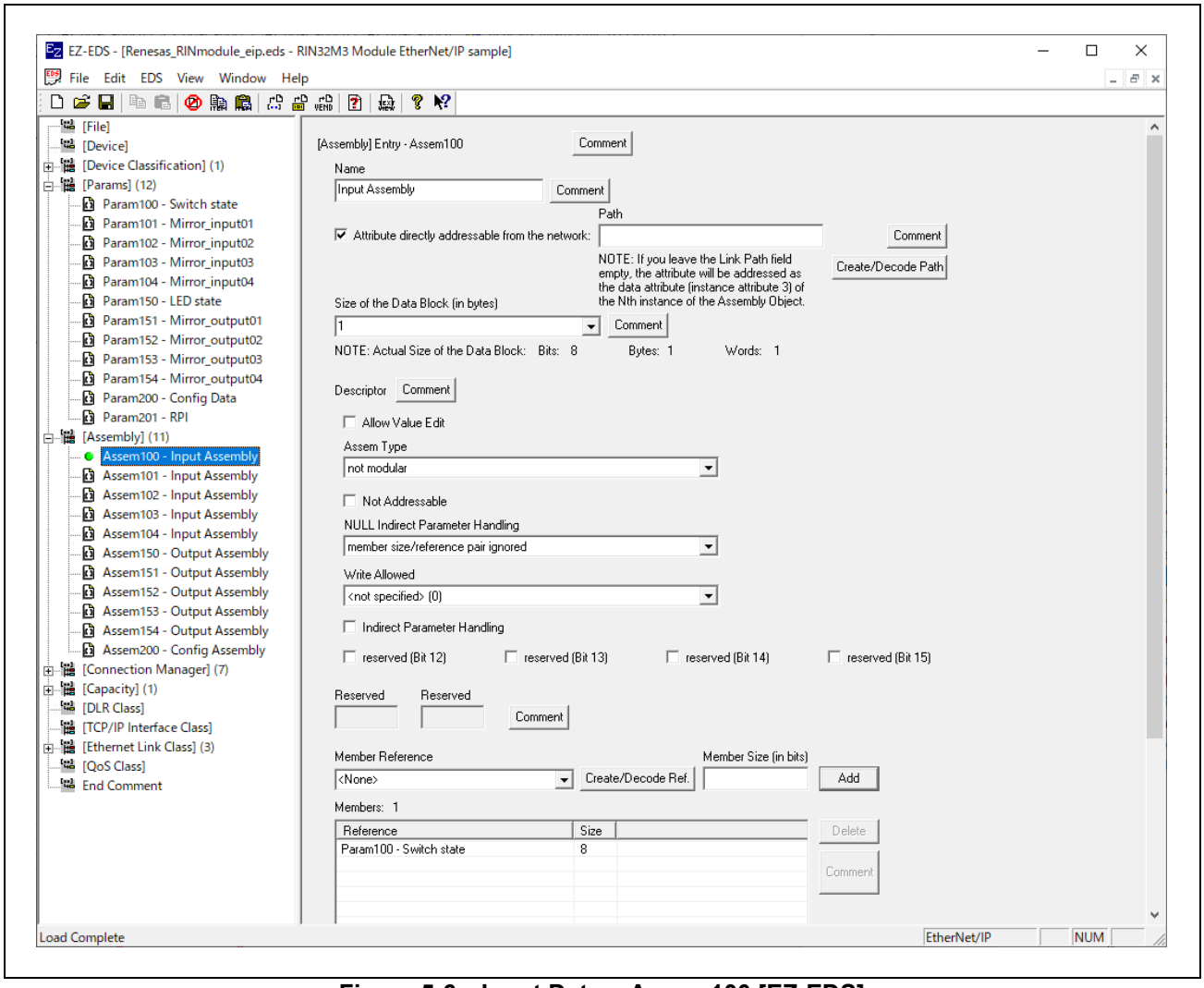

Figure 5-6 Input Data – Assem100 [EZ-EDS]

5.3 Output/Input ユーザーアプリケーションデータ操作

データ送受信に用いる API とデータ用変数を示します。

サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- Remote-IO (LED/Switch)：評価ボードの LED 点灯制御および Switch 状態を送受信
- Mirror：マスターから受信したデータをミラーバック送信

Table 5-4 API and Sample application data variable

sample		Sample app.	Data variable	Purpose	API	reference
05_eip_large	02_eip	get LED Data [#1]	ledRequest	Reception	goal_eipAssemblyObjectRead()	5.2.1 Output データ
		get Mirror Data [#3]	outData[]			
		set Switch Data [#2]	swStatus	Transmission	goal_eipAssemblyObjectWrite()	5.2.2 Input データ
		set Mirror data [#4]	inData[]			
	-	get Mirror Data_1	rpcData[]	Reception	goal_eipAssemblyObjectRead()	5.2.1 Output データ
		get Mirror Data_2	rpcData[]			
		get Mirror Data_3	rpcData[]			
		set Mirror Data_1	rpcData[]	Transmission	goal_eipAssemblyObjectWrite()	5.2.2 Input データ
		set Mirror Data_2	rpcData[]			
		set Mirror Data_3	rpcData[]			

以下は サンプルソフト 02_eip でのアプリケーション実装例になります。

```
1.  void appl_loop(
2.      void
3.  )
4.  {
5.      GOAL_STATUS_T res;                              /* result */
6.
7.      if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >=
    APPL_DM_TIMEOUT_TRIGGER_VAL))
8.      {
9.  /** USER APPLICATION --> **/
10.     uint8_t ledRequest;
11.     uint8_t swStatus;
12.
13.     /* GET Output Assembly Data (O->T) */
14.     res = goal_eipAssemblyObjectRead(pHdlEip, GOAL_APP_ASM_ID_OUTPUT_DM1, &ledRequest,
    GOAL_APP_ASM_SIZE_OUTPUT_DM1);
15.
16.     /* sample application(RemoteIO) set LED */
17.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE1, (ledRequest & 0x01) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
18.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE2, (ledRequest & 0x02) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
19.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE3, (ledRequest & 0x04) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
20.   goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE4, (ledRequest & 0x08) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
21.
22.     /* sample application(RemoteIO) get Switch */
23.     if (GOAL_RES_OK(res)) {
24.         swStatus = 0;
25.         for (uint8_t i = 0; i < 4; i++) {
26.             swStatus |= (plat_remoteIoSwGet(i) << i);
27.     }
28.     /* SET Input Assembly Data (T->O) */
29.     res = goal_eipAssemblyObjectWrite(pHdlEip, GOAL_APP_ASM_ID_INPUT_DM1, &swStatus,
    GOAL_APP_ASM_SIZE_INPUT_DM1);
30. }
31.
32.
33.     /* GET Output Assembly Data (O->T) */
34.     if (GOAL_RES_OK(res)) {
35.     res = goal_eipAssemblyObjectRead(pHdlEip, GOAL_APP_ASM_ID_OUTPUT_DM2, &outData[0],
    GOAL_APP_ASM_SIZE_OUTPUT_DM2);
36. }
37.
38. /* mirror output data to input data */
39. if (GOAL_RES_OK(res)) {
40.     GOAL_MEMCPY(&inData[0], &outData[0], GOAL_APP_ASM_SIZE_OUTPUT_DM2);
41.
42.     /* SET Input Assembly Data (T->O) */
43.     res = goal_eipAssemblyObjectWrite(pHdlEip, GOAL_APP_ASM_ID_INPUT_DM2, &inData[0],
    GOAL_APP_ASM_SIZE_INPUT_DM2);
44. }
45. /** <-- USER APPLICATION **/
46.
47.    /* update base timestamp */
48.    tsDmTout = goal_timerTsGet();
49.     }
  … omit …
50. }
```

[#1] get LED Data

[#2] set Switch Data

[#3] get Mirror Data

[#4] set Mirror Data

アプリケーションのデータ更新は uGOAL のループ周期で呼ばれる appl_loop()関数の中で制御します。データ更新周期は、データ転送方式により異なります。

サンプルソフトでは、転送方式により Table 5-5 の更新周期となっています。

Table 5-5 sample soft update cycle

Data Transfer	Update cycle Define	Sample soft default [ms]	Sample	
			02	05
Cyclic	APPL_DM_TIMEOUT_TRIG_VAL	1	✓	✓
RPC	APPL_RPC_TIMEOUT_TRIG_VAL	125	-	✓

RPC を利用したデータ転送では、EtherNet/IP の最大送受信データサイズ 69byte 以上のプロセスデータが送受信可能となりますが、アプリケーションの更新周期が増加します。Table 5-6 を参考に、扱うデータサイズにより更新周期を指定してください。

Table 5-6 Data size and update cycle comparison

Data Transfer	Data size [byte]	Average cycle [ms]
Cyclic	69	1
RPC	128	40
	256	70
	495	125

6. EtherCAT

本サンプルにて、ユーザが設定するプロファイル設定を下記に示します。

サブデバイス(スレーブ) のプロファイル情報は、ESI (EtherCAT Slave Information)ファイルでデバイスに関する機器情報をメインデバイス(マスター) 機器に提供します。そのため、スレーブが保持する機器情報とESI ファイルの情報は一致していなければいけません。

ここでは、スレーブが保持するプログラムファイルの機器情報と ESI ファイルのプロファイル設定について説明します。

サブデバイスが保持する機器情報:	appl\03_ecat\goal_appl.h
オブジェクトディクショナリ:	appl\03_ecat\goal_appl_ecat_objects.c
ESI ファイル:	appl\03_ecat\esi\Renesas_RINmodule_03ecat.xml

ESI ファイルに記述される機器情報の一部

- ベンダープロファイル
- データ構成
- 利用可能なデータ通信の種類
- オブジェクトディクショナリ

ESI ファイルは XML エディタを使用して作成、変更することが可能です。EtherCAT Technology Group (ETG) が提供する Conformance Test Tool (CTT) を利用して XML ファイルを修正、登録情報の正当性確認することが可能です。

Conformance Test Tool | EtherCAT Technology Group

[補足] XML Notepad [Microsoft Open Source] などの XML をツリー表示可能なエディッタを利用することでノードや定義の編集が容易になります。

XmlNotepad (microsoft.github.io)

EtherCAT では、プロファイル設定に関する情報は、プログラムファイルや ESI ファイルの他に、オブジェクトディクショナリと呼ばれる情報テーブルで定義する必要があります。このオブジェクトディクショナリは、通信部分とアプリケーション部分の橋渡しをするための情報テーブルです。このテーブルには、そのデバイスが取り扱う仕様が網羅されており、パラメータのセットによって機能の設定や設定情報のモニタを行います。

サンプルソフトのオブジェクトディクショナリの例を示します。Default value は 03_ecat サンプルの値を示しています。

Table 6-1 Object dictionary (1/3)

Index	Name	Subindex	Data type	Default value	Sample	
					03	06
0x1000	**Device Type**	0x00	UINT32	0x00001389	✓	✓
0x1001	**Error Register**	0x00	UINT8	0x00	✓	✓
0x1008	**Manufacturer Device Name**	0x00	STRING	RIN32M3 Module	✓	✓
0x1009	**Manufacturer Hardware Version**	0x00	STRING	1.0.00	✓	✓
0x100A	**Manufacturer Software Version**	0x00	STRING	1.0.30	✓	✓
0x1018	**Identity Object**	0x00	UINT8	0x04	✓	✓
	Vendor Id	0x01	UINT32	0x00000766	✓	✓
	Product Code	0x02	UINT32	0x00000800 *	✓	✓
	Revision Number	0x03	UINT32	0x000000002	✓	✓
	Serial Number	0x04	UINT32	0x000001234	✓	✓
0x1600	**Receive PDO Mapping Parameter 1**	0x00	UINT8	0x01	✓	✓
	Mapping Entry 1	0x01	UINT32	0x62000108	✓	✓
0x1601	**Receive PDO Mapping Parameter 2**	0x00	UINT8	0x01	✓	✓
	Mapping Entry 1	0x01	UINT32	0x62010180	✓	✓
0x1700	**Receive PDO Mapping Parameter 3**	0x00	UINT8	0x03	-	✓
	Mapping Entry 1	0x01	UINT32	0x621001F8	-	✓
	Mapping Entry 2	0x02	UINT32	0x621002F8	-	✓
	Mapping Entry 3	0x03	UINT32	0x621003F8	-	✓
0x1A00	**Transmit PDO Mapping Parameter 1**	0x00	UINT8	0x01	✓	✓
	Mapping Entry 1	0x01	UINT32	0x60000108	✓	✓
0x1A01	**Transmit PDO Mapping Parameter 2**	0x00	UINT8	0x00	✓	✓
	Mapping Entry 1	0x01	UINT32	0x60010180	✓	✓
0x1B00	**Transmit PDO Mapping Parameter 3**	0x00	UINT8	0x03	-	✓
	Mapping Entry 1	0x01	UINT32	0x601001F8	-	✓
	Mapping Entry 2	0x02	UINT32	0x601002F8	-	✓
	Mapping Entry 3	0x03	UINT32	0x601003F8	-	✓

* 06_ecat_largesize: Product Code = 0x00000804

Table 6-2 Object dictionary (2/3)

Index	Name	Subindex	Data type	Default value	Sample	
					03	06
0x1C00	**Sync Manager Communication Type**	0x00	UINT8	0x04	✓	✓
	Communication Type Sync Manager 0	0x01	UINT8	0x01	✓	✓
	Communication Type Sync Manager 1	0x02	UINT8	0x02	✓	✓
	Communication Type Sync Manager 2	0x03	UINT8	0x03	✓	✓
	Communication Type Sync Manager 3	0x04	UINT8	0x04	✓	✓
0x1C12	**Sync Manager 2 PDO Assignment**	0x00	UINT8	0x02 *	✓	✓
	PDO Mapping 1	0x01	UINT16	0x1600	✓	✓
	PDO Mapping 2	0x02	UINT16	0x1601	✓	✓
	PDO Mapping 3	0x03	UINT16	0x1700	-	✓
0x1C13	**Sync Manager 3 PDO Assignment**	0x00	UINT8	0x02 *	✓	✓
	PDO Mapping 1	0x01	UINT16	0x1A00	✓	✓
	PDO Mapping 2	0x02	UINT16	0x1A01	✓	✓
	PDO Mapping 3	0x03	UINT16	0x1B00	-	✓
0x1C32	**Output SyncManager parameter**	0x00	UINT8	0x20	✓	✓
	Synchronization Type	0x01	UINT16	0x0000	✓	✓
	Cycle Time	0x02	UINT32	0x00000000	✓	✓
	Synchronization Types Supported	0x04	UINT16	0x0000	✓	✓
	Minimum Cycle Time	0x05	UINT32	0x00000000	✓	✓
	Calc and Copy Time	0x06	UINT32	0x00000000	✓	✓
	SM-Event Missed Counter	0x0B	UINT16	0x0000	✓	✓
	Cycle Time Too Small	0x0C	UINT16	0x0000	✓	✓
	Sync Error	0x20	UINT8	0x00	✓	✓
0x1C33	**Input SyncManager parameter**	0x00	UINT8	0x20	✓	✓
	Synchronization Type	0x01	UINT16	0x0000	✓	✓
	Cycle Time	0x02	UINT32	0x00000000	✓	✓
	Synchronization Types Supported	0x04	UINT16	0x0000	✓	✓
	Minimum Cycle Time	0x05	UINT32	0x00000000	✓	✓
	Calc and Copy Time	0x06	UINT32	0x00000000	✓	✓
	SM-Event Missed Counter	0x0B	UINT16	0x0000	✓	✓
	Cycle Time Too Small	0x0C	UINT16	0x0000	✓	✓
	Sync Error	0x20	UINT8	0x00	✓	✓

* 06_ecat_largesize: Mapping Entry = 0x03

Table 6-3 Object dictionary (3/3)

Index	Name	Subindex	Data type	Default value	Sample	
					03	06
0x6000	**Read Input data**	0x00	UINT8	0x01	✓	✓
	Switch Input 1-8	0x01	UINT8	0x00	✓	✓
0x6001	**data in (dm)**	0x00	UINT8	0x01	✓	✓
	data in (dm) 1-16	0x01	ARRAY[0:15]	0x00	✓	✓
0x6010	**data in (rpc)**	0x00	UINT8	0x03	-	✓
	data in (rpc) 1-31	0x01	ARRAY[0:30]	0x00	-	✓
	data in (rpc) 32-62	0x02	ARRAY[0:30]	0x00	-	✓
	data in (rpc) 63-93	0x03	ARRAY[0:30]	0x00	-	✓
0x6200	**Write Output data**	0x00	UINT8	0x01	✓	✓
	LED Output 1-8	0x01	UINT8	0x00	✓	✓
0x6201	**data out (dm)**	0x00	UINT8	0x01	✓	✓
	data out (dm) 1-16	0x01	UINT16	0x00	✓	✓
0x6210	**data in (rpc)**	0x00	UINT8	0x03	-	✓
	data out (rpc) 1-31	0x01	ARRAY[0:30]	0x00	-	✓
	data out (rpc) 32-62	0x02	ARRAY[0:30]	0x00	-	✓
	data out (rpc) 63-93	0x03	ARRAY[0:30]	0x00	-	✓

以降で示す API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

6.1 Device Identity

ベンダ ID やベンダ名、プロダクトコード等について説明します。

これらのサブデバイスが保持する機器情報はプログラムファイル内にマクロで定義されています。

Table 6-4 EtherCAT parameter macro

Parameter	Macro	Default value [03_ecat]
Vendor ID	APPL_ECAT_VENDOR_ID	0x00000766
Vendor Name	APPL_ECAT_VENDOR	Renesas Electronics Corp.
Product Code	APPL_ECAT_PRODUCT_CODE	0x00000800 *
Revision	APPL_ECAT_REVISION_NUMBER	0x00000002
Serial	APPL_ECAT_SERIAL_NUMBER	0x00001234

* 06_ecat_largesize: Product Code = 0x00000804

6.1.1 ベンダ ID

ベンダ ID は ETG によって割り当てられます。デフォルト値は 0x00000766 (ルネサスエレクトロニクス) です。

(1) サブデバイスが保持する機器情報

” APPL_ECAT_VENDOR_ID” の値を変更します。

```
#ifndef APPL_ECAT_VENDOR_ID
# define   APPL_ECAT_VENDOR_ID     (0x00000766)          /**< Vendor ID */
#endif
```

(2) オブジェクトディクショナリ

オブジェクトディクショナリファイルでは、Index = 0x1018 (Identity Object)、Subindex = 0x01 にベンダ ID を保持します。”uint32ValueDef” には マクロ定義 “APPL_ECAT_VENDOR_ID”が反映されるので個別の設定は不要です

```
/***********************************************/
/* 1018 - Identity Object                      */
/***********************************************/
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1018, GOAL_ECAT_OBJCODE_RECORD, 0x0023);
}
 … omit …

if (GOAL_RES_OK(res)) {

    uint32ValueMin = 0x0;
    uint32ValueDef = APPL_ECAT_VENDOR_ID;
    uint32ValueMax = 0xFFFFFFFF;

    res = goal_ecatdynOdSubIndexAdd(
        pHdlEcat,
        0x1018,
        0x01,
        GOAL_ECAT_DATATYPE_UNSIGNED32,
        EC_OBJATTR_RD | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC,
        (uint8_t *) &uint32ValueDef,
        (uint8_t *) &uint32ValueMin,
        (uint8_t *) &uint32ValueMax,
        4,
        NULL);
}
```

(3) ESI ファイル

“id” および “Vendor Id” の値を変更します。

```
1.    <Vendor>
2.        <Id>#x00000766</Id>
3.        <Name>Renesas Electronics Corp.</Name>
4.    </Vendor>
```

```
1.    <Object>
2.        <Index>#x1018</Index>
3.        <Name>Identity Object</Name>
      … omit …

4.            <SubItem>
5.                <Name>Vendor Id</Name>
6.                <Info>
7.                    <DefaultValue>#x00000766</DefaultValue>
8.                </Info>
9.            </SubItem>
```

< XML ツリービュー >

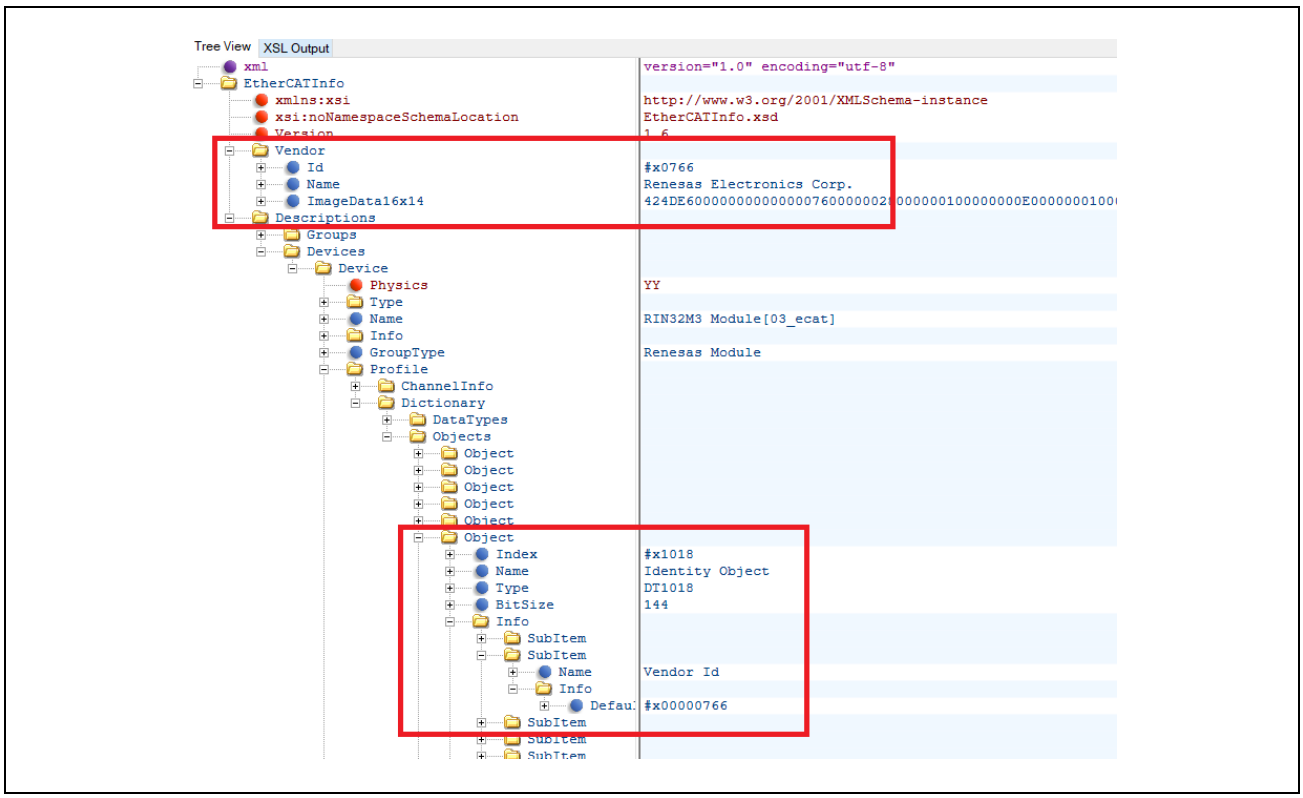

Figure 6-1 Vendor ID

6.1.2 ベンダ名

ベンダ名はベンダ ID に紐づいている必要があります。デフォルト値は “Renesas Electronics Corp.” です。

(1) サブデバイスが保持する機器情報

” APPL_ECAT_VENDOR” の値を変更します。

```
1.  #ifndef APPL_ECAT_VENDOR
2.  # define   APPL_ECAT_VENDOR     Renesas Electronics Corp.       /**< Vendor Name */
3.  #endif
```

(2) オブジェクトディクショナリ

オブジェクトディクショナリファイル内の設定は不要です。

(3) ESI ファイル

“Name” の値を変更します。

```
1.  <Vendor>
2.      <Id>#x00000766</Id>
3.      <Name>Renesas Electronics Corp.</Name>
4.  </Vendor>
```

< XML ツリービュー >

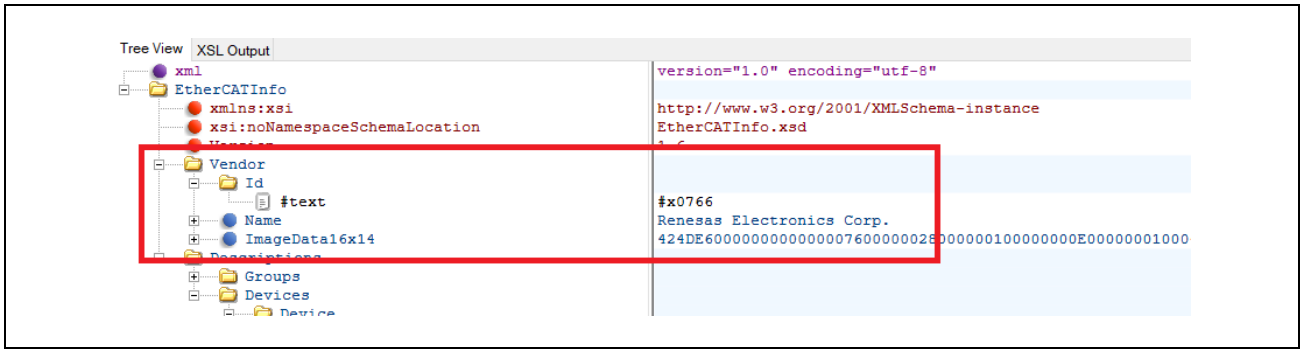

Figure 6-2 Vendor Name

6.1.3 プロダクトコード

任意の製品コードを設定します。03_ecat サンプルでのデフォルト値は 0x00000800 です。

(1) サブデバイスが保持する機器情報

” APPL_ECAT_VENDOR” の値を変更します。

```
#ifndef APPL_ECAT_PRODUCT_CODE
# define   APPL_ECAT_PRODUCT_CODE      (0x00000800)       /**< Product code */
#endif
```

(2) オブジェクトディクショナリ

オブジェクトディクショナリファイルでは、Index = 0x1018 (Identity Object)、Subindex = 0x02 にプロダクトコードを保持します。”uint32ValueDef” には マクロ定義 “APPL_ECAT_PRODUCT_CODE” が反映されるので個別の設定は不要です

```
/*********************************************/
/* 1018 - Identity Object                    */
/*********************************************/
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1018, GOAL_ECAT_OBJCODE_RECORD, 0x0023);
}
… omit …

if (GOAL_RES_OK(res)) {

    uint32ValueMin = 0x0;
    uint32ValueDef = APPL_ECAT_PRODUCT_CODE;
    uint32ValueMax = 0xFFFFFFFF;

    res = goal_ecatdynOdSubIndexAdd(
        pHdlEcat,
        0x1018,
        0x02,
        GOAL_ECAT_DATATYPE_UNSIGNED32,
        EC_OBJATTR_RD | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC,
        (uint8_t *) &uint32ValueDef,
        (uint8_t *) &uint32ValueMin,
        (uint8_t *) &uint32ValueMax,
        4,
        NULL);
}

```

(3) ESI ファイル

“ProductCode” および “Product Code” の値を変更します。

```
1.  <Devices>
2.      <Device Physics="YY">
3.          <Type ProductCode="#x00000800" RevisionNo="#x000000001">Renesas Module</Type>
4.          <Name>RIN32M3 Module[Remote-03_ecat]</Name>
5.          <info>
```

```
1.  <Object>
2.      <Index>#x1018</Index>
3.      <Name>Identity Object</Name>
        … omit …
4.          <SubItem>
5.              <Name>Product Code</Name>
6.              <Info>
7.                  <DefaultValue>#x00000800</DefaultValue>
8.              </Info>
9.          </SubItem>
```

< XML ツリービュー >

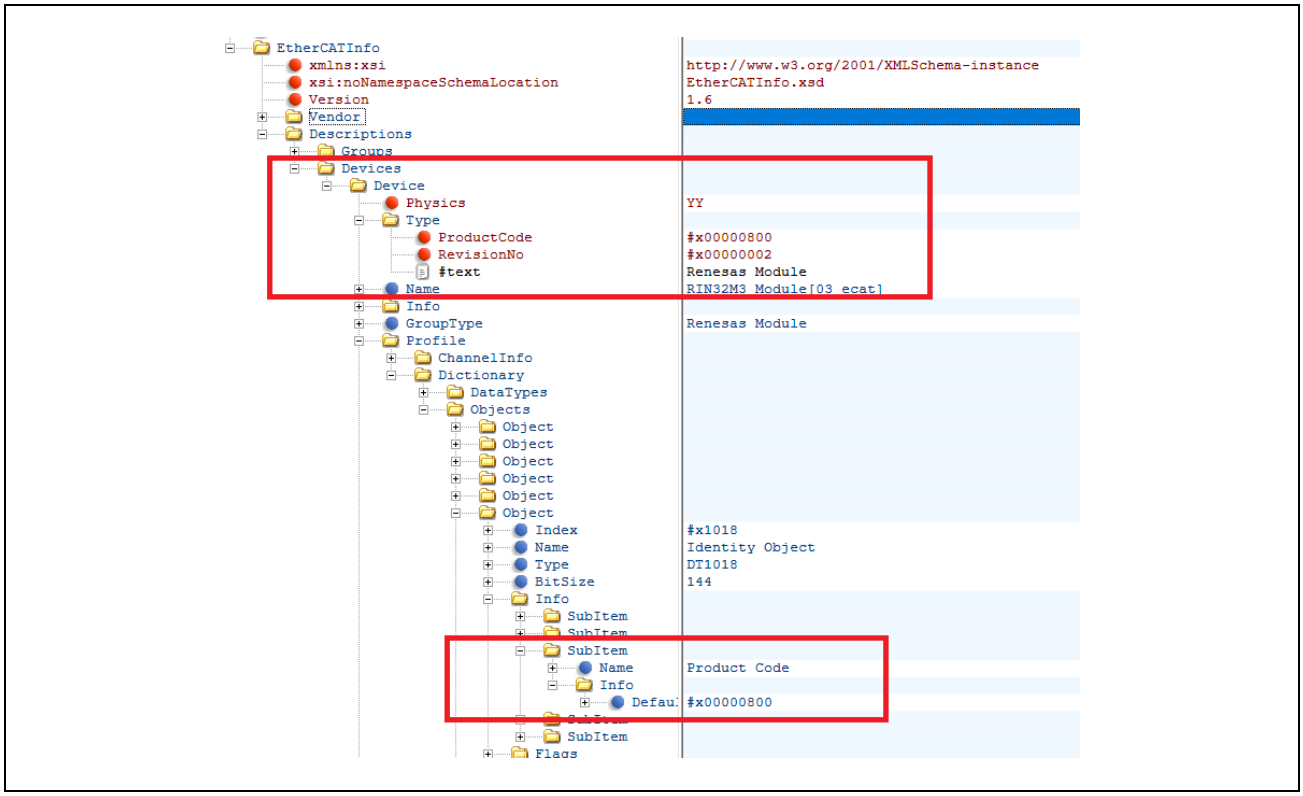

Figure 6-3 Product Code

6.1.4 リビジョン

任意の製品コードを設定します。デフォルト値は 0x00000002 です。

(1) サブデバイスが保持する機器情報

” APPL_ECAT_REVISION_NUMBER” の値を変更します。

```
#ifndef APPL_ECAT_REVISION_NUMBER
# define   APPL_ECAT_REVISION_NUMBER     (0x00000002)      /**< Revision Number */
#endif
```

(2) オブジェクトディクショナリ

オブジェクトディクショナリファイルでは、Index = 0x1018 (Identity Object)、Subindex = 0x03 にリビジョン番号を保持します。” uint32ValueDef” には マクロ定義 “APPL_ECAT_REVISION_NUMBER” が反映されるので個別の設定は不要です

```
/**********************************************/
/* 1018 - Identity Object                     */
/**********************************************/
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1018, GOAL_ECAT_OBJCODE_RECORD, 0x0023);
}
 … omit …

if (GOAL_RES_OK(res)) {

    uint32ValueMin = 0x0;
    uint32ValueDef = APPL_ECAT_REVISION_NUMBER;
    uint32ValueMax = 0xFFFFFFFF;

    res = goal_ecatdynOdSubIndexAdd(
        pHdlEcat,
        0x1018,
        0x03,
        GOAL_ECAT_DATATYPE_UNSIGNED32,
        EC_OBJATTR_RD | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC | EC_OBJATTR_NO_DFLT,
        (uint8_t *) &uint32ValueDef,
        (uint8_t *) &uint32ValueMin,
        (uint8_t *) &uint32ValueMax,
        4,
        NULL);
}
```

(3) ESI ファイル

“RevisionNo” および “Revision number” の値を変更します。

```
1.  <Devices>
2.      <Device Physics="YY">
3.          <Type ProductCode="#x00000800" RevisionNo="#x000000002">Renesas Module</Type>
4.          <Name>RIN32M3 Module[03_ecat]</Name>
5.          <info>
```

```
1.  <Object>
2.      <Index>#x1018</Index>
3.      <Name>Identity Object</Name>
    … omit …
4.          <SubItem>
5.              <Name>Revision number</Name>
6.              <Info>
7.                  <DefaultValue>#x00000002</DefaultValue>
8.              </Info>
9.          </SubItem>
```

< XML ツリービュー >

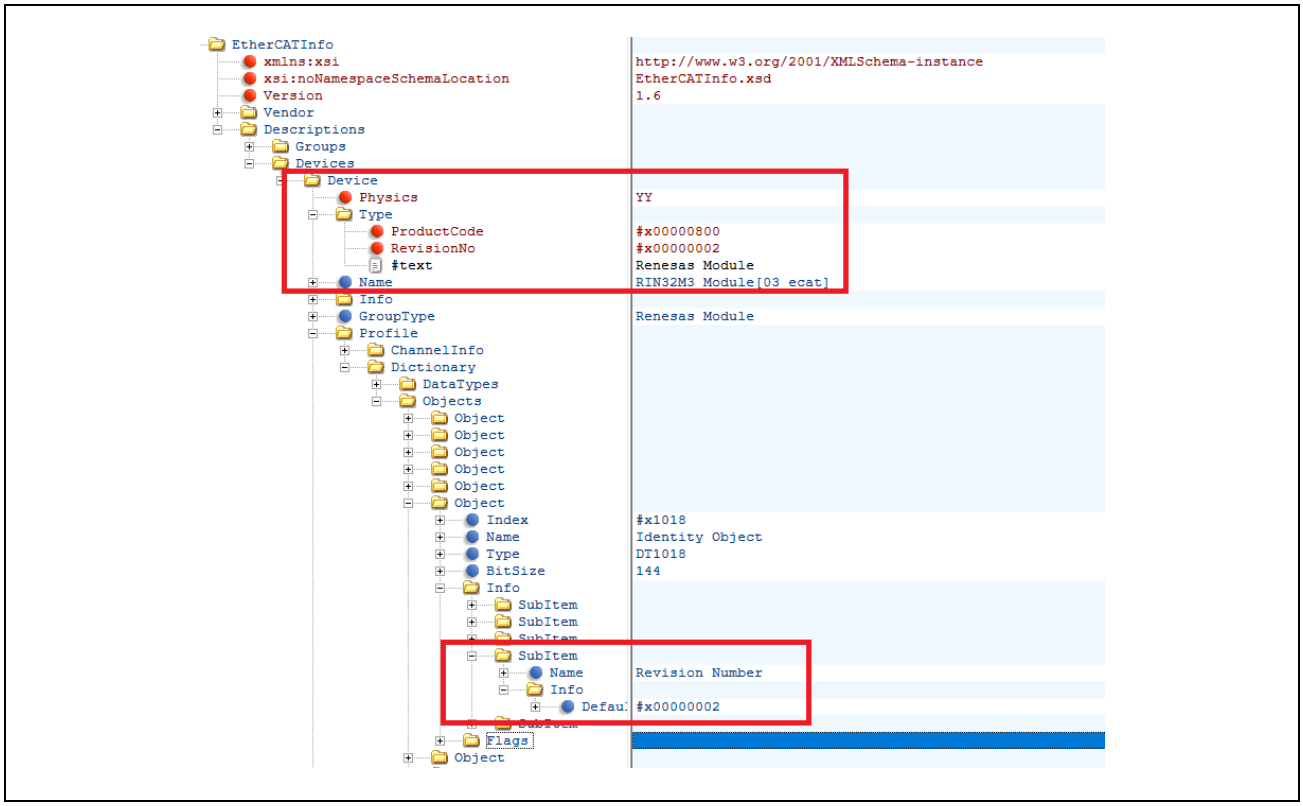

Figure 6-4 Revision Number

6.1.5 シリアル

任意のシリアル番号を設定します。デフォルト値は 0x00001234 です。

(1) サブデバイスが保持する機器情報

” APPL_ECAT_SERIAL_NUMBER” の値を変更します。

```
#ifndef APPL_ECAT_SERIAL_NUMBER
# define   APPL_ECAT_SERIAL_NUMBER      (0x00001234)        /**< Serial Number */
#endif
```

(2) オブジェクトディクショナリ

オブジェクトディクショナリファイルでは、Index = 0x1018 (Identity Object)、Subindex = 0x04 にシリアル番号を保持します。” uint32ValueDef” には マクロ定義 “APPL_ECAT_SERIAL_NUMBER” が反映されるので個別の設定は不要です

```
/**********************************************/
/* 1018 - Identity Object                     */
/**********************************************/
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1018, GOAL_ECAT_OBJCODE_RECORD, 0x0023);
}
… omit …

if (GOAL_RES_OK(res)) {

    uint32ValueMin = 0x0;
    uint32ValueDef = APPL_ECAT_SERIAL_NUMBER;
    uint32ValueMax = 0xFFFFFFFF;

    res = goal_ecatdynOdSubIndexAdd(
        pHdlEcat,
        0x1018,
        0x04,
        GOAL_ECAT_DATATYPE_UNSIGNED32,
        EC_OBJATTR_RD | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC | EC_OBJATTR_NO_DFLT,
        (uint8_t *) &uint32ValueDef,
        (uint8_t *) &uint32ValueMin,
        (uint8_t *) &uint32ValueMax,
        4,
        NULL);
}
```

(3) ESI ファイル

“Serial number” の値を変更します。

```
1.    <Object>
2.        <Index>#x1018</Index>
3.        <Name>Identity Object</Name>
      … omit …

4.              <SubItem>
5.                  <Name>Serial number</Name>
6.                  <Info>
7.                      <DefaultValue>#x00001234</DefaultValue>
8.                  </Info>
9.              </SubItem>
```

< XML ツリービュー >

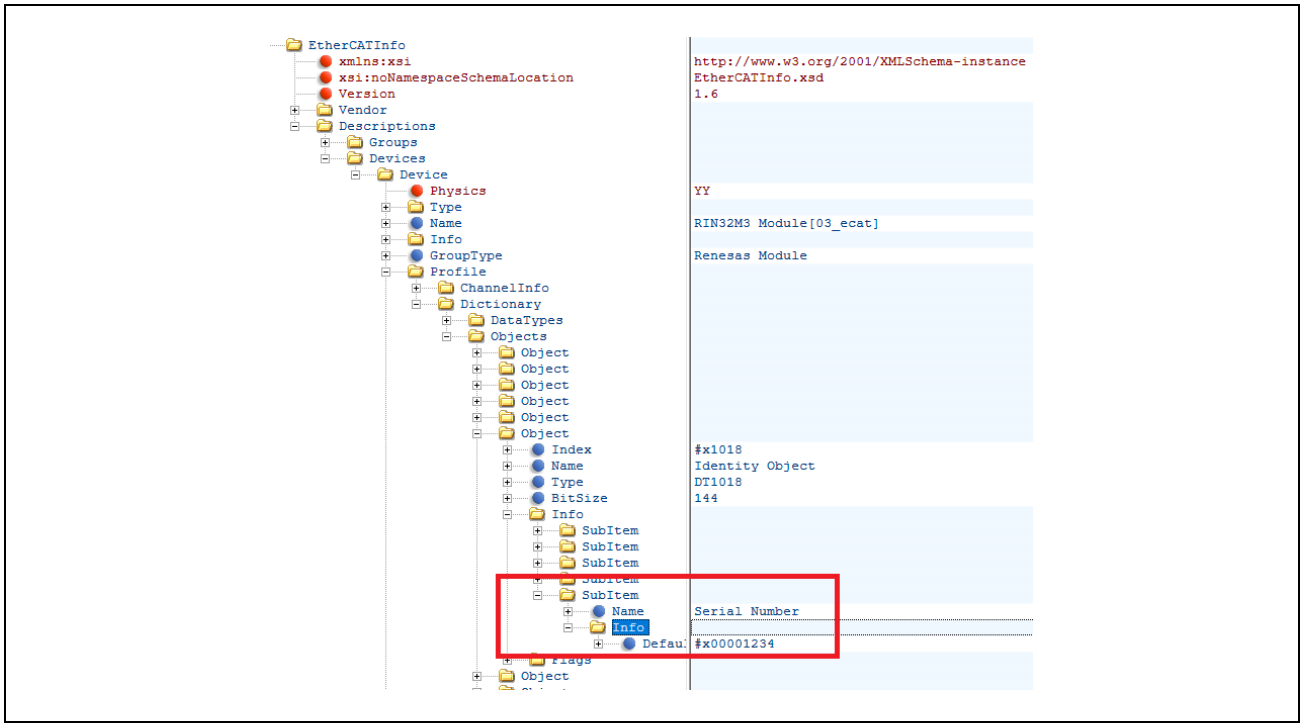

Figure 6-5 Serial Number

6.2 Data Model

アプリケーションに応じて設定する Input/Output データやサイズについて説明します。

6.2.1 Output データ

メインデバイス機器からサブデバイス(本モジュール) が受信するサンプルソフトの構成を示します。

Table 6-5 RxPDO content

Direction	Index	subindex	size	Name	Sample	
					03	06
RX	0x6200	0x00	1	Number of Entries	✓	✓
		0x01	1	LED Output 1-8	✓	✓
	0x6201	0x00	OD_RPC_SUBIDX_DEF (3)	Number of Entries	✓	✓
		0x01	OD_DM_SUBIDX_SIZE (16)	data out (dm) 1-16	✓	✓
	0x6210	0x00	OD_RPC_SUBIDX_DEF (3)	Number of Entries	-	✓
		0x01	OD_RPC_SUBIDX_SIZE (31)	data out (rpc) 1-31	-	✓
		0x02	OD_RPC_SUBIDX_SIZE (31)	data out (rpc) 32-62	-	✓
		0x03	OD_RPC_SUBIDX_SIZE (31)	data out (rpc) 63-93	-	✓

Table 6-6 RxPDO Assignment

Direction	SyncManager	Index	Name	Size	Mapping	Sample	
						03	06
RX	SyncManager2 (0x1C12)	0x1600	RxPDO1	1	0x6200 [1]	✓	✓
		0x1601	RxPDO2	16	0x6201 [1]	✓	✓
		0x1700	RxPDO3	93	0x6210 [1], 0x6210 [2], 0x6210 [3]	-	✓

Output データを変更するためには、プログラムファイル、オブジェクトディクショナリ、ESI ファイルを変更する必要があります。

(1) **サブデバイスが保持する機器情報**

RxPDO (Output データ) は g_dmobj_led に格納されます。

```
1.  extern uint8_t   g_dmobj_led;
```

(2) **オブジェクトディクショナリ**

1. メインデバイスから受信する RxPDO のオブジェクトを定義します。

ここでは以下の定義をします。

1-1. Object 生成 / 名前を定義

1-2. Subindex 0 にエントリー数を定義

1-3. Subindex 1~ に データパラメータを生成 / 名前を定義

1-1. Object 生成 / 名前を定義

```
1.  /*************************************************/
2.  /* 6200 - Write Digital Output data              */
3.  /*************************************************/
4.  if (GOAL_RES_OK(res)) {
5.      res = goal_ecatdynOdObjAdd(pHdlEcat, 0x6200, GOAL_ECAT_OBJCODE_ARRAY,
    GOAL_ECAT_DATATYPE_UNSIGNED8);
6.  }
7.
8.  /* set name */
9.  if (GOAL_RES_OK(res)) {
10.     res = goal_ecatdynOdObjNameAdd(pHdlEcat, 0x6200, "Write Output data");
11. }
12.
```

Create Object

Set Object Name

1-2. Subindex 0 にエントリー数を定義

```
13. /* 0x6200[0] Number of Entries */
14. if (GOAL_RES_OK(res)) {
15.
16.     uint8ValueMin = 0x00;
17.     uint8ValueMax = 0x01;
18.     uint8ValueDef = 0x01;
19.
20.     res = goal_ecatdynOdSubIndexAdd(
21.         pHdlEcat,
22.         0x6200,
23.         0x00,
24.         GOAL_ECAT_DATATYPE_UNSIGNED8,
25.         EC_OBJATTR_RD | EC_OBJATTR_OPT | EC_OBJATTR_NUMERIC,
26.         (uint8_t *) &uint8ValueDef,
27.         (uint8_t *) &uint8ValueMin,
28.         (uint8_t *) &uint8ValueMax,
29.         1,
30.         NULL);
31. }
32.
33. if (GOAL_RES_OK(res)) {
34.     res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x6200, 0x00, "Number of Entries");
35. }
36.
```

Create - Subindex 0

Set Name - Subindex 0

1-3. Subindex 1~ に データパラメータを生成 / 名前を定義

```
37.  /* 0x6200[1] digital output 1-8 */
38.  if (GOAL_RES_OK(res)) {
39.
40.      uint8ValueMin = 0x00;
41.      uint8ValueMax = 0xFF;
42.      uint8ValueDef = 0x00;
43.
44.      res = goal_ecatdynOdSubIndexAdd(
45.          pHdlEcat,
46.          0x6200,
47.          0x01,
48.          GOAL_ECAT_DATATYPE_UNSIGNED8,
49.          EC_OBJATTR_RD | EC_OBJATTR_WR | EC_OBJATTR_RXPDOMAPPING | EC_OBJATTR_OPT |
     EC_OBJATTR_NUMERIC,
50.          (uint8_t *) &uint8ValueDef,
51.          (uint8_t *) &uint8ValueMin,
52.          (uint8_t *) &uint8ValueMax,
53.          1,
54.          &g_dmobj_led);
55.  }
56.
57.  if (GOAL_RES_OK(res)) {
58.      res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x6200, 0x01, "digital output 1-8");
59.  }
60.
```

Create - Subindex 1

Mapp to RxPDO

Set Name - Subindex 1

2. 上記 1 で生成した Output データを RxPDO にアサインします。

ここでは以下の定義をします。

2-1. RxPDO Object 生成 / 名前を定義

2-2. Subindex 0 にエントリー数を定義

2-3. Subindex 1~ にデータパラメータを PDO にマッピング、生成 / 名前を定義

2-1. RxPDO Object 生成 / 名前を定義

```
1.  /*****************************************************************/
2.  /* 0x1600 - Receive PDO Mapping Parameter 1        */
3.  /*****************************************************************/
4.  if (GOAL_RES_OK(res)) {
5.      res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1600, GOAL_ECAT_OBJCODE_RECORD,
    EC_DEFTYPE_PDO_MAP);
6.  }
7.
8.  /* set name */
9.  if (GOAL_RES_OK(res)) {
10.     res = goal_ecatdynOdObjNameAdd(pHdlEcat, 0x1600, "Receive PDO Mapping Parameter 1");
11. }
12.
```

Create Object

Set Name - Object

2-2. Subindex 0 に Object のエントリー数を定義

```
13. /* 0x1600[0] Number of Entries */
14. if (GOAL_RES_OK(res)) {
15.
16.     uint8ValueMin = 0x00;
17.     uint8ValueMax = 0x01;
18.     uint8ValueDef = 0x01;
19.
20.     res = goal_ecatdynOdSubIndexAdd(
21.         pHdlEcat,
22.         0x1600,
23.         0x00,
24.         GOAL_ECAT_DATATYPE_UNSIGNED8,
25.         EC_OBJATTR_RD_PREOP | EC_OBJATTR_WR_PREOP | EC_OBJATTR_RD_SAFEOP |
    EC_OBJATTR_RD_OP | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC,
26.         (uint8_t *) &uint8ValueDef,
27.         (uint8_t *) &uint8ValueMin,
28.         (uint8_t *) &uint8ValueMax,
29.         1,
30.         NULL);
31. }
32.
33. if (GOAL_RES_OK(res)) {
34.     res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x1600, 0x00, "Number of Entries");
35. }
36.
```

Create - Subindex 0

Set Name - Subindex 0

2-3. Subindex 1~ に RxPDO をマッピング、生成 / 名前を定義

```
37.  /* 0x1600[1] Mapping Entry 1 */
38.  if (GOAL_RES_OK(res)) {
39.
40.      uint32ValueMin = 0x00000000;
41.      uint32ValueMax = 0xFFFFFFFF;
42.      uint32ValueDef = 0x62000108;
43.
44.      res = goal_ecatdynOdSubIndexAdd(
45.          pHdlEcat,
46.          0x1600,
47.          0x01,
48.          GOAL_ECAT_DATATYPE_UNSIGNED32,
49.          EC_OBJATTR_RD_PREOP | EC_OBJATTR_WR_PREOP | EC_OBJATTR_RD_SAFEOP |
     EC_OBJATTR_RD_OP | EC_OBJATTR_NUMERIC,
50.          (uint8_t *) &uint32ValueDef,
51.          (uint8_t *) &uint32ValueMin,
52.          (uint8_t *) &uint32ValueMax,
53.          4,
54.          NULL);
55.  }
56.
57.  if (GOAL_RES_OK(res)) {
58.      res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x1600, 0x01, "Mapping Entry 1");
59.  }
60.
```

MAPPING: Object 0x6200, Subindex [1], 1Byte

Create - Subindex 1

Set Name - Subindex 1

(3) **ESI ファイル**

1. メインデバイスから受信する Output データのオブジェクトを定義します。
ここではオブジェクト 0x6200 の設定を例に示します。

```
1.   <Object>
2.       <Index>#x6200</Index>
3.       <Name>Write Output data</Name>
4.       <Type>DT6200</Type>
5.       <BitSize>24</BitSize>
6.       <Info>
7.           <SubItem>
8.               <Name>Number of Entries</Name>
9.               <Info>
10.                  <MinValue>#x00</MinValue>
11.                  <MaxValue>#x01</MaxValue>
12.                  <DefaultValue>#x01</DefaultValue>
13.              </Info>
14.          </SubItem>
15.          <SubItem>
16.              <Name>LED Outputs 1-8</Name>
17.              <Info>
18.                  <MinValue>#x00</MinValue>
19.                  <MaxValue>#xFF</MaxValue>
20.                  <DefaultValue>#x00</DefaultValue>
21.              </Info>
22.          </SubItem>
23.      </Info>
24.  </Object>
25.
```

Object / Name

Subindex 0

Subindex 1

< XML ツリービュー >

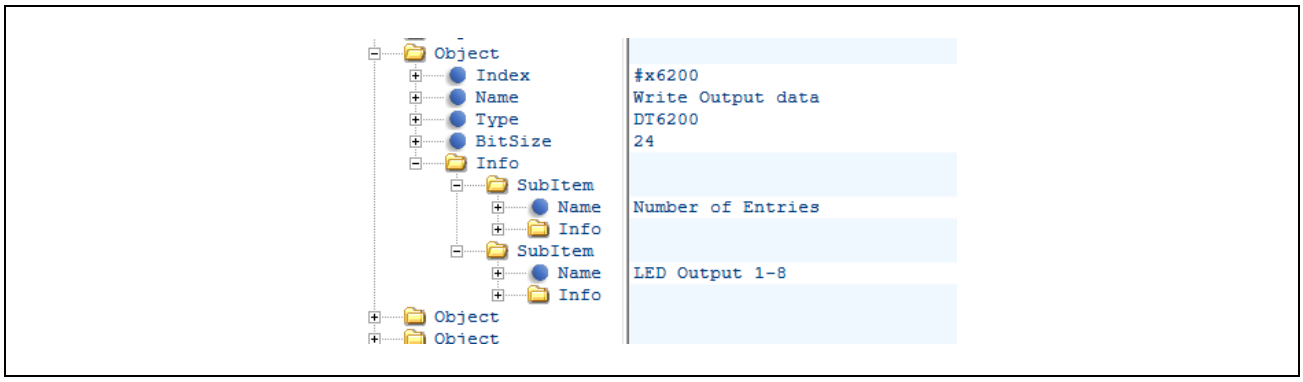

Figure 6-6 Set Object [0x6200]

2. 上記 1 で定義した Output データを RxPDO にマッピングします。

```
1.   <Object>
2.     <Index>#x1600</Index>
3.     <Name>Receive PDO Mapping Parameter 1</Name>
4.     <Type>DT1600</Type>
5.     <BitSize>48</BitSize>
6.     <Info>
7.         <SubItem>
8.             <Name>Number of Entries</Name>
9.             <Info>
10.                <MinValue>#x00</MinValue>
11.                <MaxValue>#x01</MaxValue>
12.                <DefaultValue>#x01</DefaultValue>
13.            </Info>
14.        </SubItem>
15.        <SubItem>
16.            <Name>Mapping Entry 1</Name>
17.            <Info>
18.                <MinValue>#x00000000</MinValue>
19.                <MaxValue>#xFFFFFFFF</MaxValue>
20.                <DefaultValue>#x62000108</DefaultValue>
21.            </Info>
22.        </SubItem>
23.    </Info>
24.    <Flags>
25.      <Access>ro</Access>
26.    </Flags>
27.  </Object>
28.
```

Object / Name

Subindex 0

Subindex 1

MAPPING: Object 0x6200, Subindex [1], 1Byte

< XML ツリービュー >

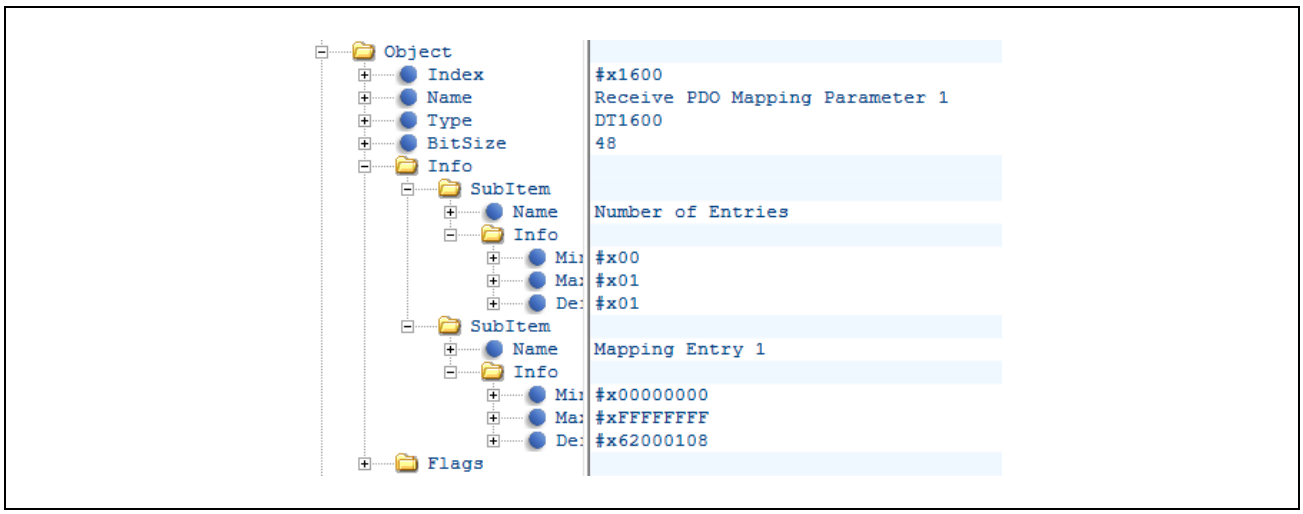

Figure 6-7 Set Object [0x1600]

3. 上記 1 で定義した RxPDO を Sync Manager にアサインします。

```
<Object>
  <Index>#x1C12</Index>
  <Name>RxPDO assign</Name>
  <Type>DT1C12</Type>
  <BitSize>64</BitSize>
  <Info>
    <SubItem>
      <Name>SubIndex 000</Name>
      <Info>
        <MinValue>#x00</MinValue>
        <MaxValue>#x03</MaxValue>
        <DefaultValue>#x03</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 001</Name>
      <Info>
        <MinValue>#x1600</MinValue>
        <MaxValue>#x17FF</MaxValue>
        <DefaultValue>#x1600</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 002</Name>
      <Info>
        <MinValue>#x1600</MinValue>
        <MaxValue>#x17FF</MaxValue>
        <DefaultValue>#x1601</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 003</Name>
      <Info>
        <MinValue>#x1600</MinValue>
        <MaxValue>#x17FF</MaxValue>
        <DefaultValue>#x1700</DefaultValue>
      </Info>
    </SubItem>
  </Info>
  <Flags>
    <Access WriteRestrictions="PreOP">rw</Access>
  </Flags>
</Object>
```

Object / Name - RxPDO

Subindex 0

Subindex 1

ASSIGNMENT: Object 0x1600

Subindex 2

ASSIGNMENT: Object 0x1601

Subindex 3 *

ASSIGNMENT: Object 0x1700

* subindex 3 は、06_ecat_largesize のみサンプル例として定義しています。

< XML ツリービュー >

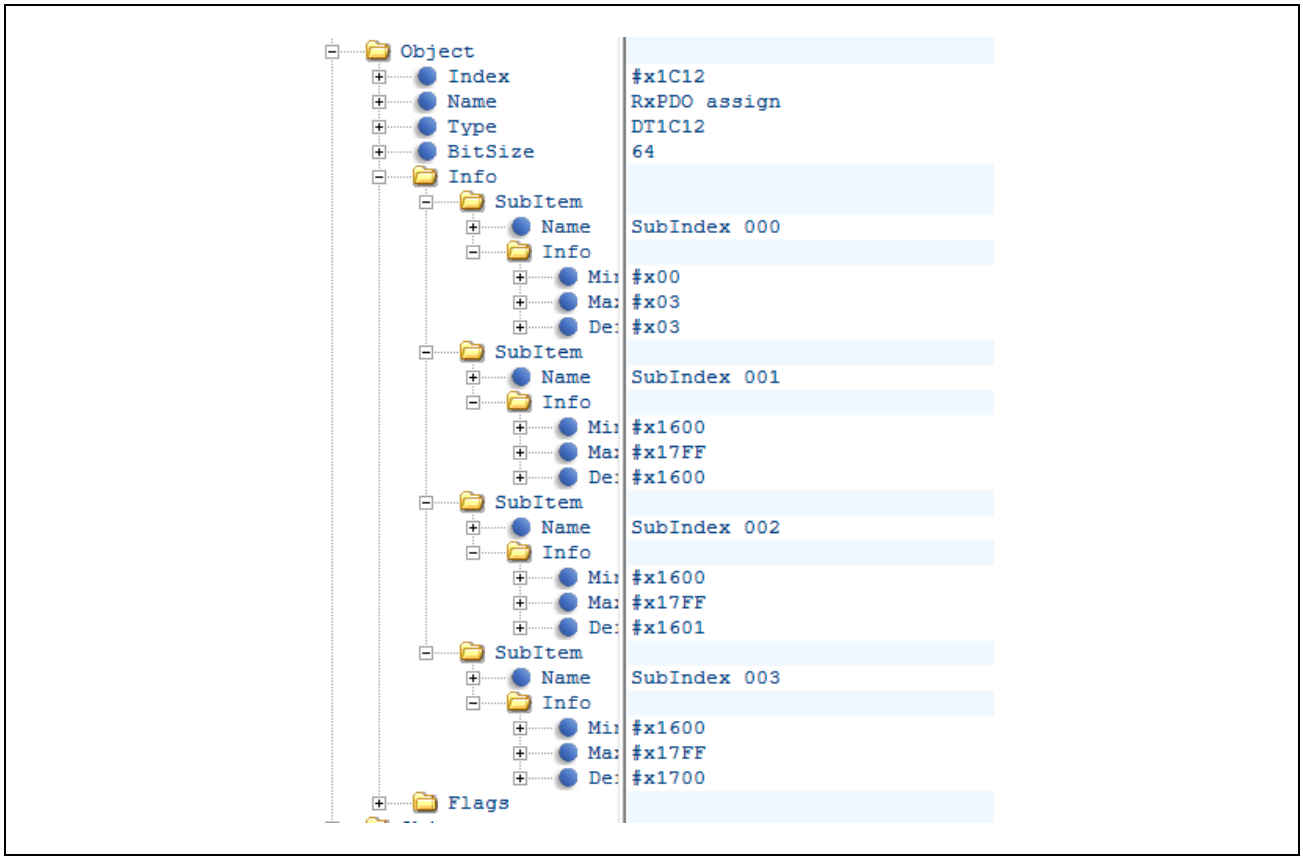

Figure 6-8 Set Object [0x1C12]

4. 上記で定義した RxPDO のエントリーを定義します。RxPDO は Sync Manager 2 PDO Assignment へエントリーします。

```
<RxPdo Fixed="0" Sm="2">
    <Index>#x1600</Index>
    <Name>RxPDO 1</Name>
    <Entry>
        <Index>#x6200</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>8</BitLen>
        <Name>LED</Name>
        <DataType>USINT</DataType>
    </Entry>
</RxPdo>
<RxPdo Fixed="0" Sm="2">
    <Index>#x1601</Index>
    <Name>RxPDO 2</Name>
    <Entry>
        <Index>#x6201</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>128</BitLen>
        <Name>dout_dm_1</Name>
        <DataType>ARRAY [0..15] OF BYTE</DataType>
    </Entry>
</RxPdo>
<RxPdo Fixed="0" Sm="2">
    <Index>#x1700</Index>
    <Name>RxPDO 3</Name>
    <Entry>
        <Index>#x6210</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>248</BitLen>
        <Name>dout_rpc_1</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
    <Entry>
        <Index>#x6210</Index>
        <SubIndex>#x02</SubIndex>
        <BitLen>248</BitLen>
        <Name>dout_rpc_2</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
    <Entry>
        <Index>#x6210</Index>
        <SubIndex>#x03</SubIndex>
        <BitLen>248</BitLen>
        <Name>dout_rpc_3</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
</RxPdo>
```

RxPDO1:
ENTRY: Sync Manager 2 Assignment
Index 0x1600,
Object 0x6200 [1]

RxPDO2:
ENTRY: Sync Manager 2 Assignment
Index 0x1601,
Object 0x6201 [1]

RxPDO3: *
ENTRY: Sync Manager 2 Assignment
Index 0x1700,
Object 0x6210 [1]
Object 0x6210 [2]
Object 0x6210 [3]

* RxPDO3 は、06_ecat_largesize のみサンプル例として定義しています。

< XML ツリービュー >

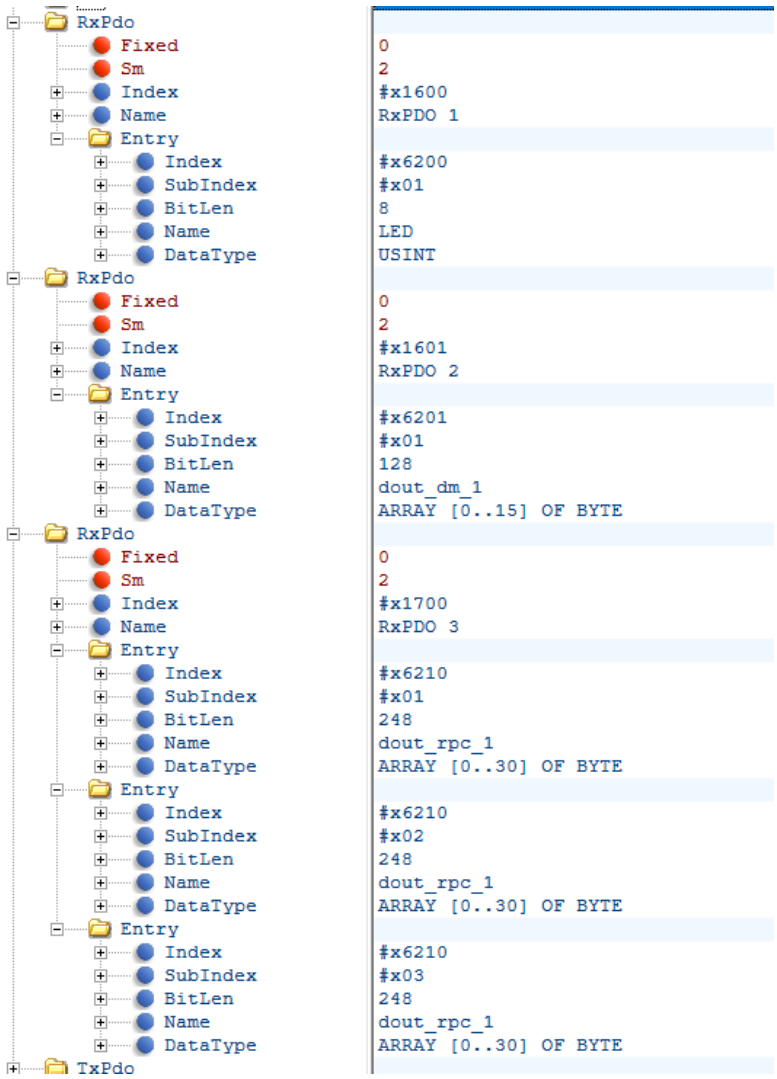

Figure 6-9 RxPDO Entry

6.2.2 Input データ

サブデバイス(本モジュール) がメインデバイスへ送信するデータを定義します。

Table 6-7 TxPDO contents

Direction	Index	subindex	size	Name	Sample	
					03	06
TX	0x6000	0x00	1	Number of Entries	✓	✓
		0x01	1	Switch Input 1-8	✓	✓
	0x6001	0x00	OD_RPC_SUBIDX_DEF (3)	Number of Entries	✓	✓
		0x01	OD_DM_SUBIDX_SIZE (16)	data in (dm) 1-16	✓	✓
	0x6200	0x00	OD_RPC_SUBIDX_DEF (3)	Number of Entries	-	✓
		0x01	OD_RPC_SUBIDX_SIZE (31)	data in (rpc) 1-31	-	✓
		0x02	OD_RPC_SUBIDX_SIZE (31)	data in (rpc) 32-62	-	✓
		0x03	OD_RPC_SUBIDX_SIZE (31)	data in (rpc) 63-93	-	✓

Table 6-8 TxPDO Assignment

Direction	SyncManager	Index	Name	Size	Mapping	Sample	
						03	06
TX	SyncManager3 (0x1C13)	0x1A00	TxPDO1	1	0x6000 [1]	✓	✓
		0x1A01	TxPDO2	16	0x6001 [1]	✓	✓
		0x1B00	TxPDO3	93	0x6010 [1], 0x6010 [2], 0x6010 [3]	-	✓

Input データを変更するためには、プログラムファイル、オブジェクトディクショナリファイル、ESI ファイルを変更する必要があります。

(1) **サブデバイスが保持する機器情報**

Input データは g_dmobj_sw に格納されます。

```
1.  uint8_t   g_dmobj_sw;
```

(2) **オブジェクトディクショナリ**

1. メインデバイスから受信する RxPDO contents のオブジェクトを定義します。

ここでは以下の定義をします。

1-1. Object 生成 / 名前を定義

1-2. Subindex 0 にエントリー数を定義

1-3. Subindex 1~ に データパラメータを生成 / 名前を定義

1-1. Object 生成 / 名前を定義

```
1.  /*********************************************/
2.  /* 6000 - Read Digital Input data            */
3.  /*********************************************/
4.  if (GOAL_RES_OK(res)) {
5.      res = goal_ecatdynOdObjAdd(pHdlEcat, 0x6000, GOAL_ECAT_OBJCODE_ARRAY,
    GOAL_ECAT_DATATYPE_UNSIGNED8);
6.  }
7.
8.  /* set name */
9.  if (GOAL_RES_OK(res)) {
10.     res = goal_ecatdynOdObjNameAdd(pHdlEcat, 0x6000, "Read Input data");
11. }
12.
```

Create Object

Set Name - Object

1-2. Subindex 0 にエントリー数を定義

```
13. /* 0x6000[0] Number of Entries */
14. if (GOAL_RES_OK(res)) {
15.
16.     uint8ValueMin = 0x00;
17.     uint8ValueMax = 0x01;
18.     uint8ValueDef = 0x01;
19.
20.     res = goal_ecatdynOdSubIndexAdd(
21.         pHdlEcat,
22.         0x6000,
23.         0x00,
24.         GOAL_ECAT_DATATYPE_UNSIGNED8,
25.         EC_OBJATTR_RD | EC_OBJATTR_OPT | EC_OBJATTR_NUMERIC,
26.         (uint8_t *) &uint8ValueDef,
27.         (uint8_t *) &uint8ValueMin,
28.         (uint8_t *) &uint8ValueMax,
29.         1,
30.         NULL);
31. }
32.
33. if (GOAL_RES_OK(res)) {
34.     res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x6000, 0x00, "Number of Entries");
35. }
36.
```

Create - Subindex 0

Set Name - Subindex 0

1-3. Subindex 1~ に データパラメータを生成 / 名前を定義

```
37.  /* 0x6000[1] digital input 1-8 */
38.  if (GOAL_RES_OK(res)) {
39.
40.      uint8ValueMin = 0x00;
41.      uint8ValueMax = 0x02;
42.      uint8ValueDef = 0x00;
43.
44.      res = goal_ecatdynOdSubIndexAdd(
45.          pHdlEcat,
46.          0x6000,
47.          0x01,
48.          GOAL_ECAT_DATATYPE_UNSIGNED8,
49.          EC_OBJATTR_RD | EC_OBJATTR_TXPDOMAPPING | EC_OBJATTR_OPT |
     EC_OBJATTR_NUMERIC,
50.          (uint8_t *) &uint8ValueDef,
51.          (uint8_t *) &uint8ValueMin,
52.          (uint8_t *) &uint8ValueMax,
53.          1,
54.          &g_dmobj_sw);
55.  }
56.
57.  if (GOAL_RES_OK(res)) {
58.      res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x6000, 0x01, "Switch Input 1-8");
59.  }
60.
```

Create - Subindex 1

Mapp to TxPDO

Set Name - Subindex 1

2. 上記 1 で生成した Output データを TxPDO にアサインします。

ここでは以下の定義をします。

2-1. TxPDO Object 生成 / 名前を定義

2-2. Subindex 0 にエントリー数を定義

2-3. Subindex 1~ にデータパラメータを PDO にマッピング、生成 / 名前を定義

2-1. TxPDO Object 生成 / 名前を定義

```
/*************************************************************/
/* 1A00 - Transmit PDO Mapping Parameter 1           */
/*************************************************************/
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjAdd(pHdlEcat, 0x1A00, GOAL_ECAT_OBJCODE_RECORD,
EC_DEFTYPE_PDO_MAP);
}

/* set name */
if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdObjNameAdd(pHdlEcat, 0x1A00, "Transmit PDO Mapping Parameter 1");
}
```

Create Object

Set Name - Object

2-2. Subindex 0 に Object のエントリー数を定義

```
/* 0x1A00[0] Number of Entries */
if (GOAL_RES_OK(res)) {

    uint8ValueMin = 0x00;
    uint8ValueMax = 0x01;
    uint8ValueDef = 0x01;

    res = goal_ecatdynOdSubIndexAdd(
        pHdlEcat,
        0x1A00,
        0x00,
        GOAL_ECAT_DATATYPE_UNSIGNED8,
        EC_OBJATTR_RD_PREOP | EC_OBJATTR_WR_PREOP | EC_OBJATTR_RD_SAFEOP |
EC_OBJATTR_RD_OP | EC_OBJATTR_MAN | EC_OBJATTR_NUMERIC,
        (uint8_t *) &uint8ValueDef,
        (uint8_t *) &uint8ValueMin,
        (uint8_t *) &uint8ValueMax,
        1,
        NULL);
}

if (GOAL_RES_OK(res)) {
    res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x1A00, 0x00, "Number of Entries");
}
```

Create - Subindex 0

Set Name - Subindex 0

2-3. Subindex 1~ に PDO をマッピング、生成 / 名前を定義

```
37.  /* 0x1A00[1] Mapping Entry 1 */
38.  if (GOAL_RES_OK(res)) {
39.
40.      uint32ValueMin = 0x00000000;
41.      uint32ValueMax = 0xFFFFFFFF;
42.      uint32ValueDef = 0x60000108;
43.
44.      res = goal_ecatdynOdSubIndexAdd(
45.          pHdlEcat,
46.          0x1A00,
47.          0x01,
48.          GOAL_ECAT_DATATYPE_UNSIGNED32,
49.          EC_OBJATTR_RD_PREOP | EC_OBJATTR_WR_PREOP | EC_OBJATTR_RD_SAFEOP |
     EC_OBJATTR_RD_OP | EC_OBJATTR_NUMERIC,
50.          (uint8_t *) &uint32ValueDef,
51.          (uint8_t *) &uint32ValueMin,
52.          (uint8_t *) &uint32ValueMax,
53.          4,
54.          NULL);
55.  }
56.
57.  if (GOAL_RES_OK(res)) {
58.      res = goal_ecatdynOdSubIndexNameAdd(pHdlEcat, 0x1A00, 0x01, "Mapping Entry 1");
59.  }
60.
```

MAPPING: Object 0x6000, Subindex [1], 1Byte

Create - Subindex 1

Set Name - Subindex 1

(3) **ESI ファイル**

1. メインデバイスから受信する Input データのオブジェクトを定義します。
ここではオブジェクト 0x6000 の設定を例に示します。

```
1.  <Object>
2.      <Index>#x6000</Index>
3.      <Name>Read INtput data</Name>
4.      <Type>DT6000</Type>
5.      <BitSize>32</BitSize>
6.      <Info>
7.          <SubItem>
8.              <Name>Number of Entries</Name>
9.              <Info>
10.                 <MinValue>#x00</MinValue>
11.                 <MaxValue>#x01</MaxValue>
12.                 <DefaultValue>#x01</DefaultValue>
13.             </Info>
14.         </SubItem>
15.         <SubItem>
16.             <Name>Switch Input 1-8</Name>
17.             <Info>
18.                 <MinValue>#x00</MinValue>
19.                 <MaxValue>#xFF</MaxValue>
20.                 <DefaultValue>#x00</DefaultValue>
21.             </Info>
22.         </SubItem>
23.     </Info>
24. </Object>
```

Object / Name

Subindex 0

Subindex 1

< XML ツリービュー >

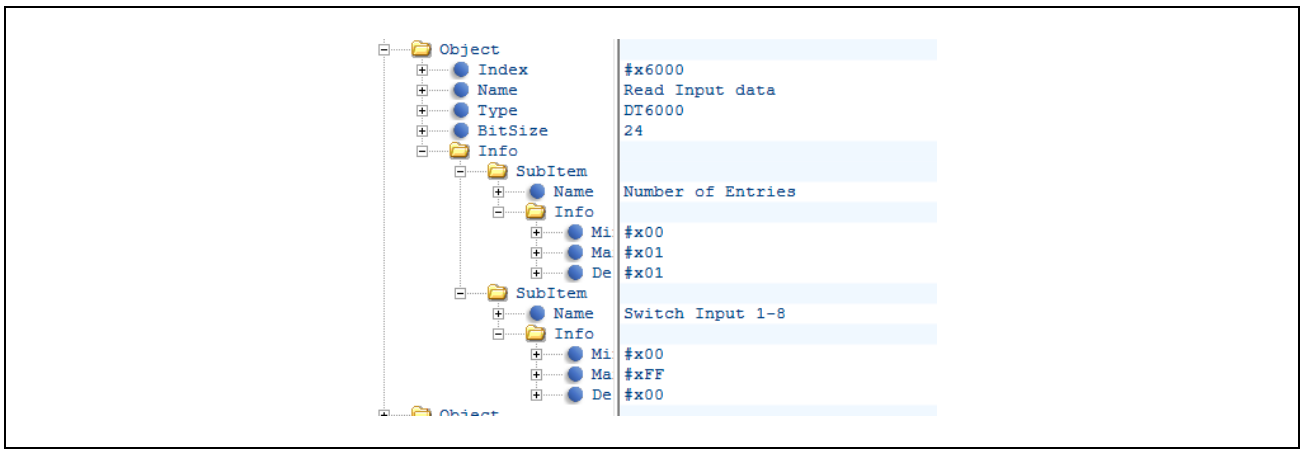

Figure 6-10 Set Object [0x6000]

2. 上記 1 で定義した Input データを TxPDO にマッピングします。

```
1.   <Object>
2.       <Index>#x1A00</Index>
3.       <Name>Receive PDO Mapping Parameter 1</Name>
4.       <Type>DT1A00</Type>
5.       <BitSize>48</BitSize>
6.       <Info>
7.           <SubItem>
8.               <Name>Number of Entries</Name>
9.               <Info>
10.                  <MinValue>#x00</MinValue>
11.                  <MaxValue>#x01</MaxValue>
12.                  <DefaultValue>#x01</DefaultValue>
13.              </Info>
14.          </SubItem>
15.          <SubItem>
16.              <Name>Mapping Entry 1</Name>
17.              <Info>
18.                  <MinValue>#x00000000</MinValue>
19.                  <MaxValue>#xFFFFFFFF</MaxValue>
20.                  <DefaultValue>#x60000108</DefaultValue>
21.              </Info>
22.          </SubItem>
23.      </Info>
24.      <Flags>
25.        <Access>ro</Access>
26.      </Flags>
27.  </Object>
28.
```

Object / Name

Subindex 0

Subindex 1

MAPPING: Object 0x6000, Subindex [1], 1Byte

< XML ツリービュー >

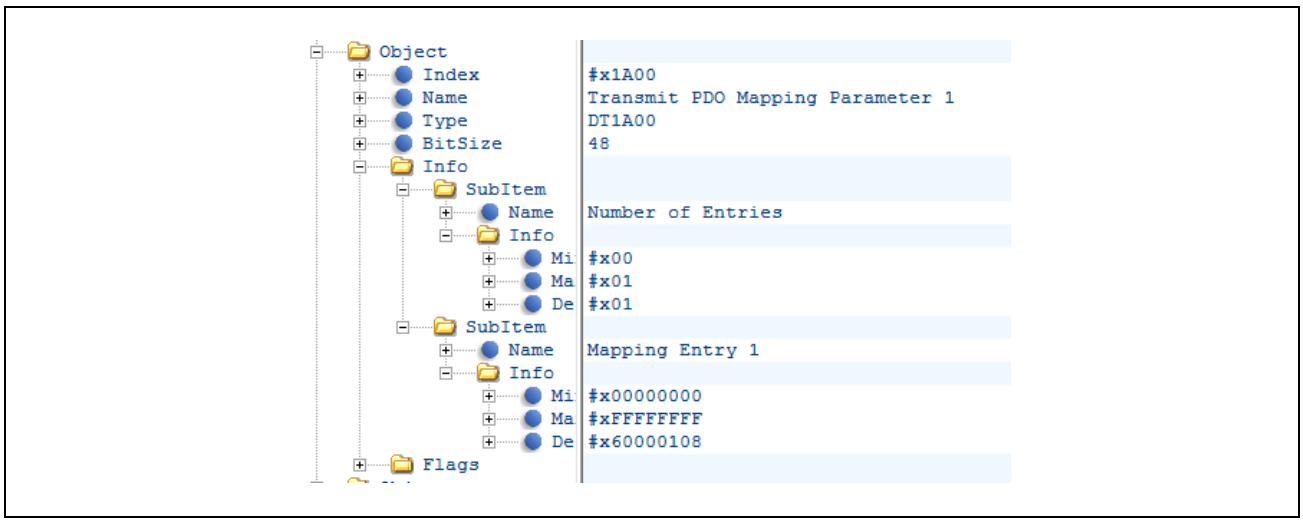

Figure 6-11 TxPDO Entry

3. 上記 1 で定義した TxPDO を Sync Manager にアサインします。

```
<Object>
  <Index>#x1C13</Index>
  <Name>TxPDO assign</Name>
  <Type>DT1C13</Type>
  <BitSize>64</BitSize>
  <Info>
    <SubItem>
      <Name>SubIndex 000</Name>
      <Info>
        <MinValue>#x00</MinValue>
        <MaxValue>#x03</MaxValue>
        <DefaultValue>#x03</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 001</Name>
      <Info>
        <MinValue>#x1A00</MinValue>
        <MaxValue>#x1BFF</MaxValue>
        <DefaultValue>#x1A00</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 002</Name>
      <Info>
        <MinValue>#x1A00</MinValue>
        <MaxValue>#x1BFF</MaxValue>
        <DefaultValue>#x1A01</DefaultValue>
      </Info>
    </SubItem>
    <SubItem>
      <Name>SubIndex 003</Name>
      <Info>
        <MinValue>#x1A00</MinValue>
        <MaxValue>#x1BFF</MaxValue>
        <DefaultValue>#x1B00</DefaultValue>
      </Info>
    </SubItem>
  </Info>
  <Flags>
    <Access WriteRestrictions="PreOP">rw</Access>
  </Flags>
</Object>
```

Object / Name - TxPDO

Subindex 0

Subindex 1

ASSIGNMENT: Object 0x1A00

Subindex 2

ASSIGNMENT: Object 0x1A01

Subindex 3 *

ASSIGNMENT: Object 0x1B00

* subindex 3 は、06_ecat_largesize のみサンプル例として定義しています。

< XML ツリービュー >

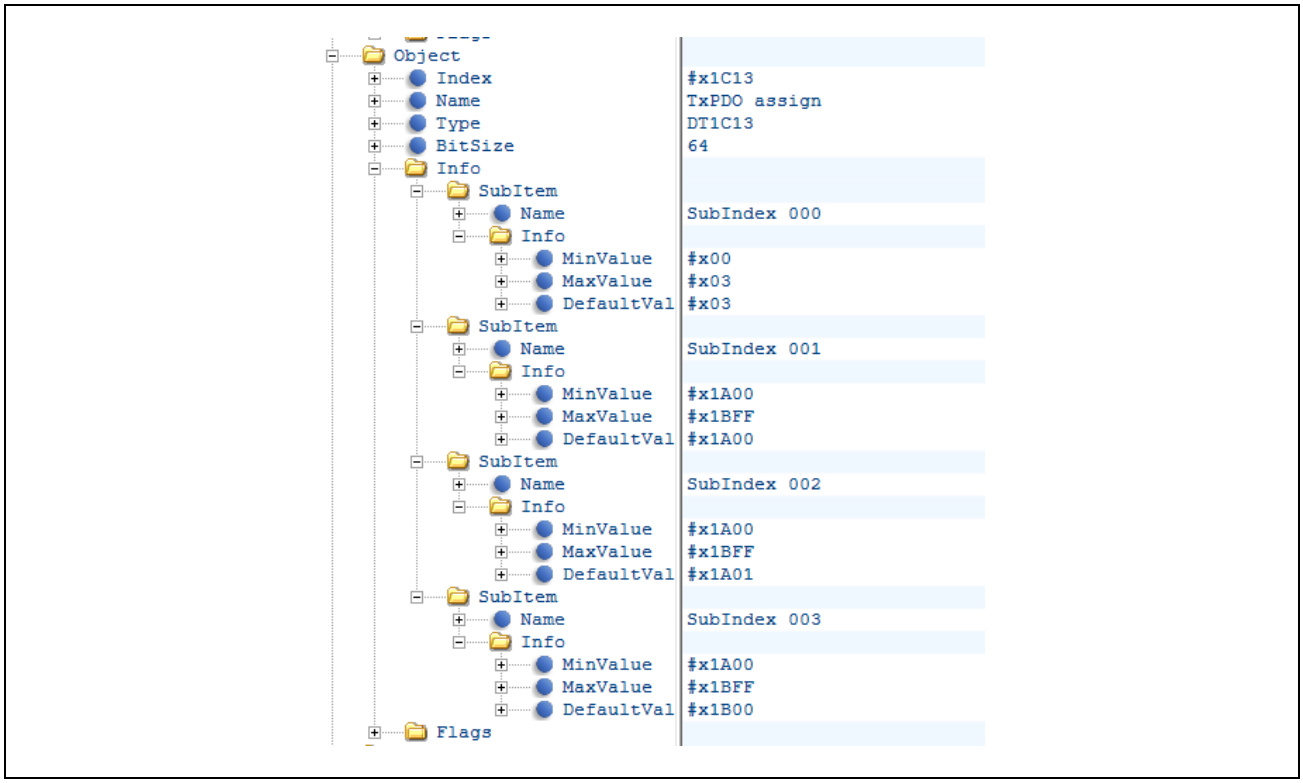

Figure 6-12 Set Object [0x1C13]

4. 上記 2 で定義した TxPDO のエントリーを定義します。TxPDO は Sync Mangar 3 PDO Assignment へエントリーします。

```
<TxPdo Fixed="0" Sm="3">
    <Index>#x1A00</Index>
    <Name>TxPDO 1</Name>
    <Entry>
        <Index>#x6000</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>8</BitLen>
        <Name>SW</Name>
        <DataType>USINT</DataType>
    </Entry>
</TxPdo>
<TxPdo Fixed="0" Sm="3">
    <Index>#x1A01</Index>
    <Name>TxPDO 2</Name>
    <Entry>
        <Index>#x6001</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>128</BitLen>
        <Name>din_dm_1</Name>
        <DataType>ARRAY [0..15] OF BYTE</DataType>
    </Entry>
</TxPdo>
<TxPdo Fixed="0" Sm="3">
    <Index>#x1B00</Index>
    <Name>TxPDO 3</Name>
    <Entry>
        <Index>#x6010</Index>
        <SubIndex>#x01</SubIndex>
        <BitLen>248</BitLen>
        <Name>din_rpc_1</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
    <Entry>
        <Index>#x6010</Index>
        <SubIndex>#x02</SubIndex>
        <BitLen>248</BitLen>
        <Name>din_rpc_2</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
    <Entry>
        <Index>#x6010</Index>
        <SubIndex>#x03</SubIndex>
        <BitLen>248</BitLen>
        <Name>din_rpc_3</Name>
        <DataType>ARRAY [0..30] OF BYTE</DataType>
    </Entry>
</TxPdo>
```

TxPDO1:
ENTRY: Sync Manager 3 Assignment
Index 0x1A00,
Object 0x6000 [1]

TxPDO2:
ENTRY: Sync Manager 3 Assignment
Index 0x1A01,
Object 0x6001 [1]

TxPDO3: *
ENTRY: Sync Manager 3 Assignment *
Index 0x1B00,
Object 0x6010 [1]
Object 0x6010 [2]
Object 0x6010 [3]

* Object 0x6010 は、06_ecat_largesize のみサンプル例として定義しています。

< XML ツリービュー >

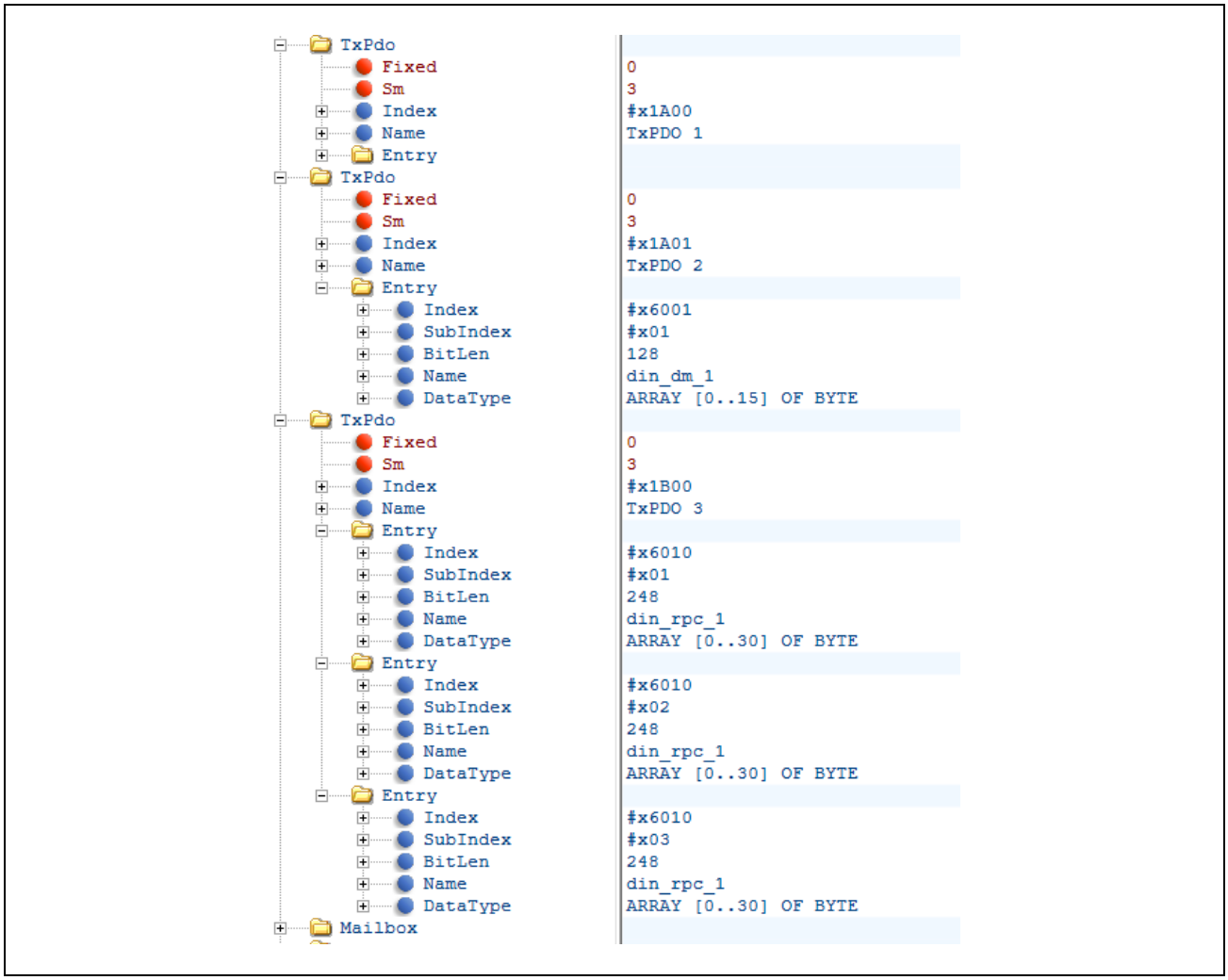

Figure 6-13 TxPDO Entry

6.3 Output/Input ユーザーアプリケーションデータ操作

データ送受信に用いる API とデータ用変数を示します。

サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- Remote-IO (LED/Switch)：評価ボードの LED 点灯制御および Switch 状態を送受信
- Mirror：マスターから受信したデータをミラーバック送信

Table 6-9 API and Sample application data variable

sample		Sample app.	Data variable	Purpose	API	reference
06_ecat_large	03_ecat	get LED Data [#1]	g_dmobj_led	Reception	goal_ecatdynOdSubIndexAdd()	6.2.1 Output データ
		get Mirror Data [#3]	g_dmobj_outdata			
		set Switch Data [#2]	g_dmobj_sw	Transmission		6.2.2 Input データ
		set Mirror data [#4]	g_dmobj_indata			
	-	get Mirror Data_1	rpcData[]	Reception	goal_ecatdynOdSubIndexRpcAdd ()	6.2.1 Output データ
		get Mirror Data_2	rpcData[]			
		get Mirror Data_3	rpcData[]			
		set Mirror Data_1	rpcData[]	Transmission		6.2.2 Input データ
		set Mirror Data_2	rpcData[]			
		set Mirror Data_3	rpcData[]			

以下は サンプルソフト 03_ecat でのアプリケーション実装例になります。

```
1.  void appl_loop(
2.      void
3.  )
4.  {
    …. omit ….
5.
6.      if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >=
    APPL_DM_TIMEOUT_TRIGGER_VAL)) {
7.  /* output data reflected to LED */
8.      goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE1, (g_dmobj_led & 0x01) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
9.      goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE2, (g_dmobj_led & 0x02) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
10.     goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE3, (g_dmobj_led & 0x04) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
11.     goal_maLedSet(pMaLed, GOAL_MA_LED_REMOTE4, (g_dmobj_led & 0x08) ?
    GOAL_MA_LED_STATE_ON : GOAL_MA_LED_STATE_OFF);
12.
13. /* switch state be set to input data */
14. g_dmobj_sw = 0;
15. for (uint8_t i = 0; i < 4; i++) {
16.     g_dmobj_sw |= (plat_remoteIoSwGet(i) << i);
17. }
18.
19. /* Received data(RxPDO) copy to Transport data(TxPDO) */
20. GOAL_MEMCPY(g_dmobj_indata, g_dmobj_outdata, OD_DM_SUBIDX_SIZE);
21.
22. /* update base timestamp */
23. tsTout = goal_timerTsGet();
24. }

    … omit …

25. }
```

[#1] LED Data

[#2] Switch Data

[#3] [#4] Mirror Data

アプリケーションのデータ更新は uGOAL のループ周期で呼ばれる appl_loop()関数の中で制御します。データ更新周期は、データ転送方式により異なります。

サンプルソフトでは、転送方式により Table 6-10 の更新周期となっています。

Table 6-10 sample soft update cycle

Data Transfer	Update cycle Define	Sample soft default [ms]	Sample	
			03	06
Cyclic	APPL_DM_TIMEOUT_TRIG_VAL	1	✓	✓
RPC	APPL_RPC_TIMEOUT_TRIG_VAL	310	-	✓

RPC を利用したデータ転送では、EtherCAT の最大送受信データサイズ 64byte 以上のプロセスデータが送受信可能となりますが、アプリケーションの更新周期が増加します。Table 6-11 を参考に、扱うデータサイズにより更新周期を指定してください。

Table 6-11 Data size and update cycle comparison

Data Transfer	Data size [byte]	Average cycle [ms]
Cyclic	64	1
RPC	128	40
	256	70
	512	125
	1024	230
	1408	310

Appendix

A. IP アドレス設定

本章では、R-IN32M3 Module の IP アドレス設定について説明します。

R-IN32M3 Module の IP アドレスは、起動時に内部の不揮発性メモリに保存された GOAL_ID_NET (12)設定に従い設定されますが、ホスト CPU から IP アドレスを設定する *goal_maNetIpSet()* を呼び出して設定することも可能です。
本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションでは、デフォルト設定では、内部に保存された設定を元に IP アドレスが設定されるようになっており(Configured IP)、プログラム内に“GOAL_CONFIG_STATIC_IP”マクロを 1 に定義することでホストマイコンから任意の IP アドレス(Static IP)を設定することができます。

Table A-1 IP Configuration (GOAL_ID_NET)

Variable Name	Variable ID	Type	Max. Size	Description
IP	0	GOAL_CM_IPV4	4	IP address of first interface
NETMASK	1	GOAL_CM_IPV4	4	NETMASK of first interface
GW	2	GOAL_CM_IPV4	4	GATEWAY of first interface
VALID	3	GOAL_CM_UINT8	1	Validity of IP address: 0, Stored IP address is not valid, interface settings originate from network stack of system 1, Stored IP address is valid, will be applied to interface at start of device
DHCP_ENABLED	4	GOAL_CM_UINT8	1	DHCP enable: 0, DHCP disabled 1, DHCP enabled

不揮発性メモリに保存された IP アドレス設定を有効とするためには、VALID = 1 となっている必要がありますのでご注意ください。*goal_maNetIpSet()* を実行すると IP、NETMASK および GW 設定は不揮発性メモリにも保存されますが、VALID 設定については最後の引数 flgTemp で保存するかどうか指定することができます。(GOAL_FALSE：VALID 設定を更新、GOAL_TRUE：VALID 設定は更新せず)

```
1.  GOAL_STATUS_T goal_maNetIpSet(
2.      GOAL_MA_NET_T *pNetHdl,          /**< pointer to store NET handler */
3.      uint32_t addrIp,                 /**< IP address */
4.      uint32_t addrMask,               /**< subnet mask */
5.      uint32_t addrGw,                 /**< gateway */
6.      GOAL_BOOL_T flgTemp              /**< temporary IP config flag */
7.  );
```

また、DHCP を有効とする場合は、GOAL_ID_NET (12) の DHCP_ENABLED を 1 に設定するか、EtherNet/IP の場合は、*goal_eipCfgDhcpOn()* を呼び出します。02_eip サンプルでは、プログラム内に“GOAL_CONFIG_ENABLE_DHCP”マクロを 1 で定義することで、DHCP が有効になります。

Table A-2 に IP アドレスの設定方法の一覧を示します。

Table A-2 IP address setting list

Methods	Descriptions
Configured IP	・R-IN32M3 Module 内の不揮発性メモリに保持された値を使用します。 ・Management Tool を使って値の変更が可能です。詳細は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照してください。 ・本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションのデフォルト設定は、この方法になります。
Static IP	・主に評価用に用いられます。 ・変更した値は R-IN32M3 Module 内の不揮発性メモリに保持されます。 ・本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションで値の変更が可能です。プログラム内に“GOAL_CONFIG_STATIC_IP”マクロを 1 で定義することで、任意の IP アドレス設定が可能になります。
DHCP	・Management Tool を使って DHCP の有効無効の変更が可能です。 ・本サンプルの”02_eip”と”05_eip_large”サンプルでも DHCP の変更が可能で、デフォルト設定は無効です。プログラム内に“GOAL_CONFIG_ENABLE_DHCP”マクロを 1 で定義することで、DHCP が有効になります。 ・DHCP が有効、且つ、DHCP サーバがネットワーク上に無い場合は、R-IN32M3 Module 内の不揮発性メモリに保持された値を使用します。

B. Module Name と Customer ID 設定

本章では、R-IN32M3 Module の内部情報の 1 つである Module Name と Customer ID の設定について説明します。

R-IN32M3 Module の Module Name と Customer ID は、起動時に内部の不揮発性メモリに保存された GOAL_ID_DD (34)設定に従い設定されます。もし、ホスト CPU で定義したこれらの値と差異があれば、*appl_ccmCfgVarSet()* によって設定され、*appl_ccmCfgSave()* によって不揮発性メモリに保存されます。

Table B-3 IP Configuration (GOAL_ID_DD)

Variable Name	Variable ID	Type	Max. Size	Description
MODULENAME	0	GOAL_CM_STRING	20	Customer specific name of the module
CUSTOMERID	1	GOAL_CM_UNIT32	4	Customer Id
RESERVED	2	GOAL_CM_UINT8	1	-
FEATURE_DISABLE	3	GOAL_CM_UINT32	4	Each bit disables a function: bit 0, disable “HELLO DETECTION” bit 1, disable WINK bit 2, disable GETLIST bit 3, disable GET VALUE bit 4, disable SET VALUE

Module Name と Customer ID は、各サンプルの goal_appl.h ファイルに定義が用意されています。

Module Name については、マクロで定義されている”APPL_DD_MODULE_NAME”の値を変更します。デフォルト値は”R-IN32M3_Module”です。

例)

ファイル: appl\01_pnio\goal_appl.h

```
#ifndef APPL_DD_MODULE
# define APPL_DD_MODULE              R-IN32M3_Module       /**< module name */
#endif

…. omit ….

#define APPL_DD_MODULE_NAME       STRING(APPL_DD_MODULE)
```

Customer ID については、マクロで定義されている”APPL_DD_CUSTOMER_ID”の値を変更します。デフォルト値は 0x0000001 です。

例)

ファイル: appl\ 01_pnio\goal_appl.h

```
#ifndef APPL_DD_CUSTOMER_ID
# define APPL_DD_CUSTOMER_ID      (0x00000001)          /**< customer id */
#endif
```

C. データサイズ上限

本サンプルで搭載している uGOAL ミドルウェアの仕様により、取り扱うことが出来るデータのサイズには上限があり、各プロトコル、及び、”Long packet”データであるかどうかでその値が異なります。

ここで RPC Transfer は、R-IN32M3 Module とホストマイコン間の SPI 通信フレーム(128byte)における RPC データフレームを使って周期通信を行う方式です(Table 3-4 参照)。本来、非同期データ通信に用いることを目的とした RPC フレームを使用するため、通常の Cyclic データフレームを使用した方式に比べ大きいサイズのデータを送ることができますがアプリケーションの更新周期は制限があります。

Table C-1 Maximum data size for each protocol

Protocol	RPC transfer	Sample application	Maximum data size [byte]	Refer
PROFINET	-	01_pnio	69	Table 4-6
	✔	04_pnio_largesize	1434	
EtherNet/IP	-	02_eip	69	Table 5-6
	✔	05_eip_largesize	495	
EtherCAT	-	03_ecat	64	Table 6-11
	✔	06_ecat_largesize	1408	

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021.10.15	—	新規作成
1.01	2022.8.5	8	フォルダ構成を更新
		23	メモリ容量を更新
1.02	2023.5.31	—	軽微な修正
		8	2.1.1: 最新サンプルソフトウェアの構成に変更
		44	4: サンプルソフトウェア更新にともなう見直し
		62	4.3: Cyclic 通信,RPC 通信の更新周期とデータサイズの説明追加
		66	5: サンプルソフトウェア更新にともなう見直し
		77	5.3: Cyclic 通信,RPC 通信の更新周期とデータサイズの説明追加
		80	6: サンプルソフトウェア更新にともなう見直し
		116	6.3: Cyclic 通信,RPC 通信の更新周期とデータサイズの説明追加
1.03	2023.12.15	75	Table 5-3 を修正
		28, 122	RPC Transfer について説明追加

商標

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うと、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、ノイズ幅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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、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。
5. 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。
6. 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。
7. 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。
8. 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て、当社は、一切その責任を負いません。
9. 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。
10. 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いたします。
11. 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。
12. 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な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.4.0-1　2017.11)

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。