

# ROS2rapper インターフェース解説書

2024/JUL/16

株式会社アックス

1. 概要.....	3
1.1. はじめに.....	3
1.2. ROS2とROS2rapper.....	4
1.3. サポートしている機能.....	5
1.3.1. トピックのパブリッシュ.....	5
1.3.2. トピックのサブスクライブ.....	5
1.3.3. UDPデータグラムの送信.....	5
1.3.4. UDPデータグラムの受信.....	5
1.4. 制限事項.....	6
1.5. ROS2・RTPS用語の対応.....	6
2. アーキテクチャ.....	7
3. ROS2rapper IPとユーザ・ロジックの接続.....	11
3.1. 概要.....	11
3.2. ROS2rapperのクロック、機能の有効化など.....	12
3.3. Ethernetモジュールとの接続.....	13
3.3.1. Ethernet MACとの接続.....	14
3.3.2. PHYの接続.....	15
3.3.3. パケット送信間隔パラメータの設定.....	15
3.4. ROS2rapperのポートの設定.....	16
3.5. アプリケーション・データ・レジスタ(送信)の用意と接続.....	17
3.6. アプリケーション・データ・メモリの用意と接続.....	18
3.7. IP受信ペイロード・メモリの用意と接続.....	19
3.8. UDP送信メモリの用意と接続.....	20
3.9. UDP受信メモリの用意と接続.....	21
4. ROS2通信.....	22
4.1. ROS2rapper各機能の有効化.....	22
4.2. パブリッシャの送信メッセージ(アプリケーション・データ・レジスタ)の変更.....	23
4.3. サブスクライバの受信メッセージ(アプリケーション・データ・メモリ)の読み出し.....	25
5. UDPデータグラム通信.....	26
5.1. UDPデータグラムの送信.....	26
5.2. UDPデータグラムの受信.....	27
6. 参考文献.....	28
7. 付録.....	29
7.1. ros2_etherモジュールのインターフェース仕様.....	29
7.2. ディレクトリ構成.....	35
7.3. RTPS通信で使用するUDPポート番号 一覧.....	37

# 1. 概要

## 1.1. はじめに

“ROS2rapper”は、ROS2の通信のRTPSプロトコルなどのハードウェア実装であり、HLS C++とVerilogで記述されている。

本文書では、ROS2rapperを利用したシステムの設計を行うユーザーのための情報を提供する。

## 1.2. ROS2とROS2rapper

ROS2採用システムにおけるROS2rapperの位置づけを、Fig.1.1に示す。着色された部分が、ROS2rapperの各モジュールである。

ROS2rapperは、RTPS/UDP/IPプロトコルを実装している。それに加えて、ROS2rapperのIP及び実装サンプルには通信の物理層にEthernetを用いた実装も含んでいる。

なお、Ethernet以外の物理層も使用可能である。そのためには、物理層およびROS2rapperとのインターフェース部分を、別途実装すること。

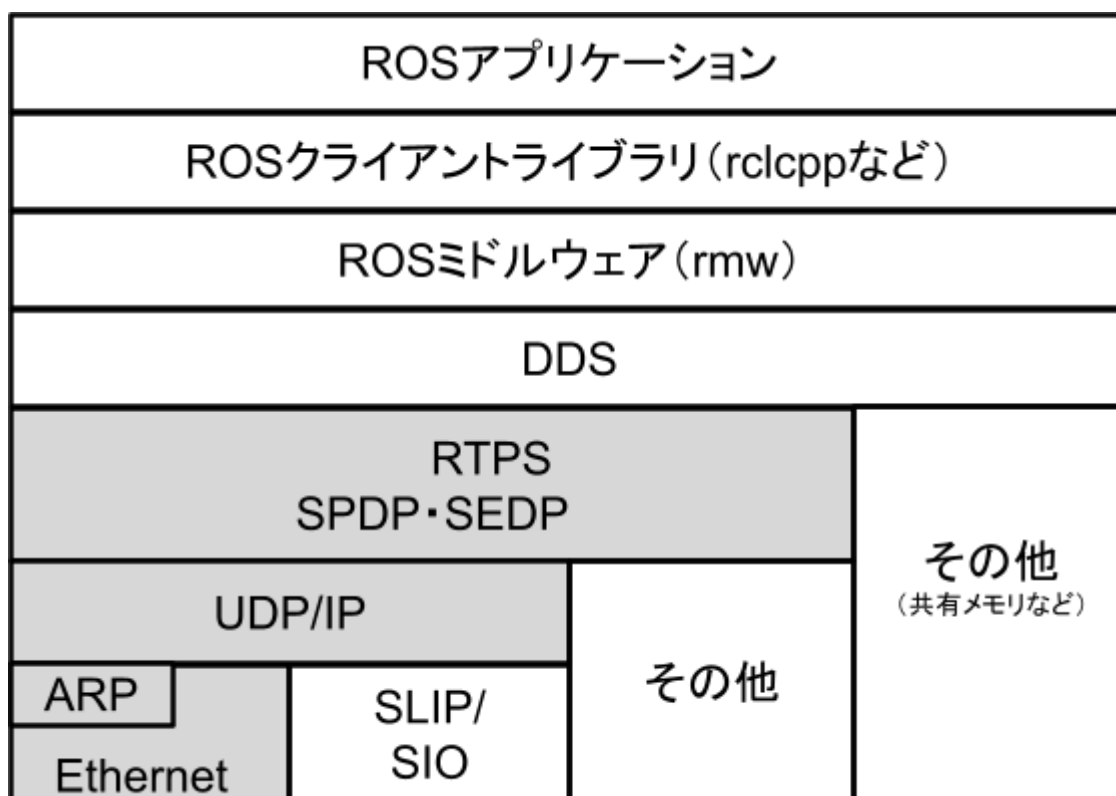


Fig.1.1 ROS2アーキテクチャにおけるROS2rapperの位置づけ

## 1.3. サポートしている機能

本IPが提供する機能をTable 1.1に示す。

本ROS2rapper IPは、下記の各機能を独立して個々に有効化/無効化できる。

- ROS2パブリッシュ機能
- ROS2サブスクライブ機能
- Ethernet

Table 1.1 サポート機能一覧

機能		内容	備考
プロトコル	RTPS SPDP SEDP	ROS2トピックのパブリッシュ	
		ROS2トピックのサブスクライブ	
	UDP/IP	UDPデータグラムの送信	
		UDPデータグラムの受信	
通信物理層	Ethernet	Ethernet層	Ethernet以外の物理層に差し替え可能
	ARP	物理アドレス解決	

### 1.3.1. トピックのパブリッシュ

ROS2ネットワークへトピックをパブリッシュする。トピックをサブスクライブしたすべてのノードに対し、一定周期でメッセージを送信する。メッセージの型は任意であるが、ユーザが適切にシリアライズする必要がある。

### 1.3.2. トピックのサブスクライブ

ROS2ネットワーク上のトピックをサブスクライブする。パブリッシャからメッセージが届くと、その内容をメモリ・インターフェースで外部メモリに書き出し、通知信号をアサート(「1」と)する。

### 1.3.3. UDPデータグラムの送信

任意のペイロードを持つUDPデータグラムを送信する。ユーザは、宛先IPアドレス・宛先ポート・送信元ポートを指定できる。

### 1.3.4. UDPデータグラムの受信

指定したポート番号宛のUDPデータグラムを受信することができる。ポート番号は1つのみ指定できる。RTPS通信に支障が出るため、RTPS通信で使用するポート番号(付録7.3参照)は指定

してはならない。なお、受信するUDPデータグラムはペイロード長が248バイト以内でなければならない。その長さを超えるデータグラムは、破棄される。

## 1.4. 制限事項

本ROS2rapperのIPには、以下の制限事項が存在する。

- ノード数は1個
- パブリッシュ/サブスクライブできるトピックは合計4つ
- 通信相手のノードの離脱は検出できない
- メッセージ・サイズや通信可能ノード数などの制限(定数一覧を参照)

## 1.5. ROS2・RTPS用語の対応

本IPはRTPSの実装であるが、本書では主に、対応するROS2の用語で説明する。

なお、本文中やソースコード中に、一部RTPSの用語を使用している箇所がある。用語の読み替えのため、以下にその対照表を示す。

Table1.2 ROS2とRTPSの用語の対応

ROS2用語	RTPS用語
Node	Participant
Message	Application Data
Topic	Endpoint
Publisher	Writer
Subscriber	Reader

## 2. アーキテクチャ

ROS2rapperの内部の機能ブロック図を、Fig. 2.1に示す。

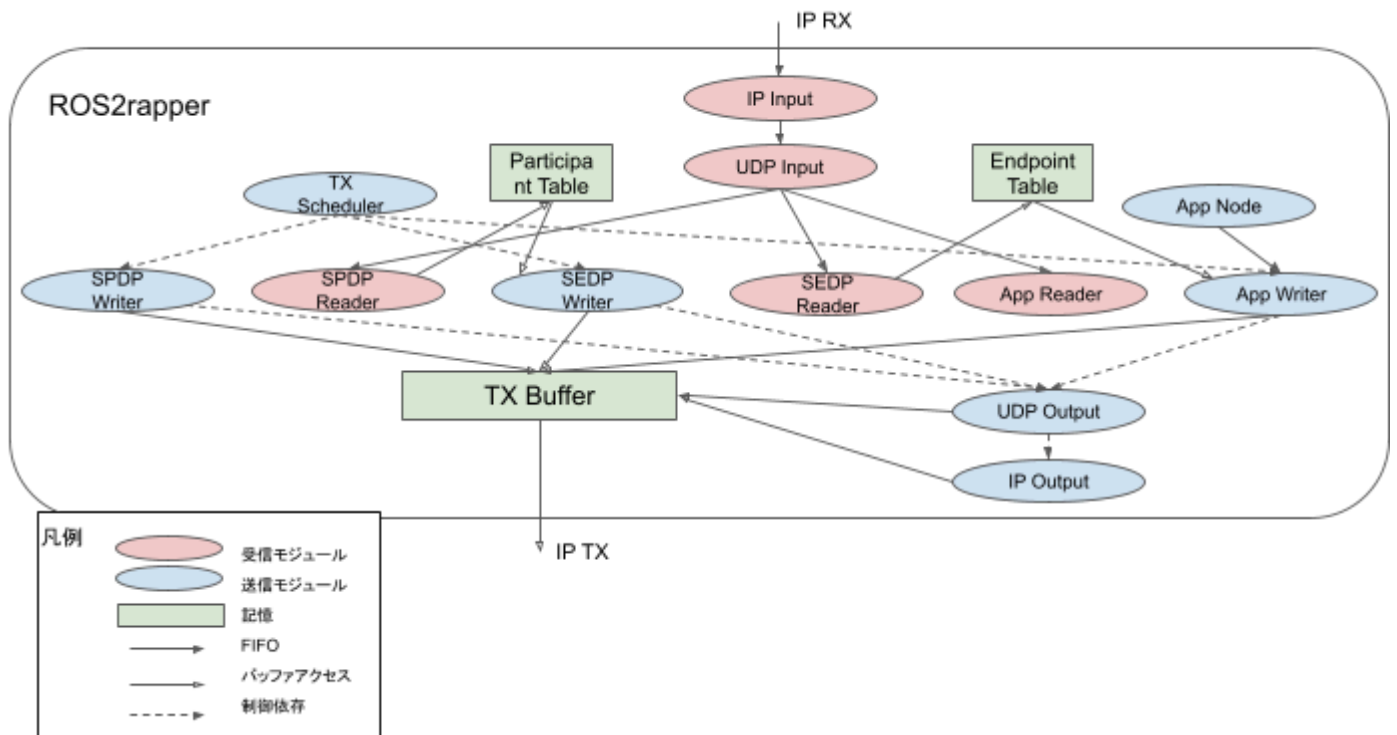


Fig. 2.1 ROS2rapper機能ブロック図

通信の物理層をEthernetとして、ROS2rapperを使用する時(ros2\_ether)のモジュール構成図を、Fig. 2.2に示す。

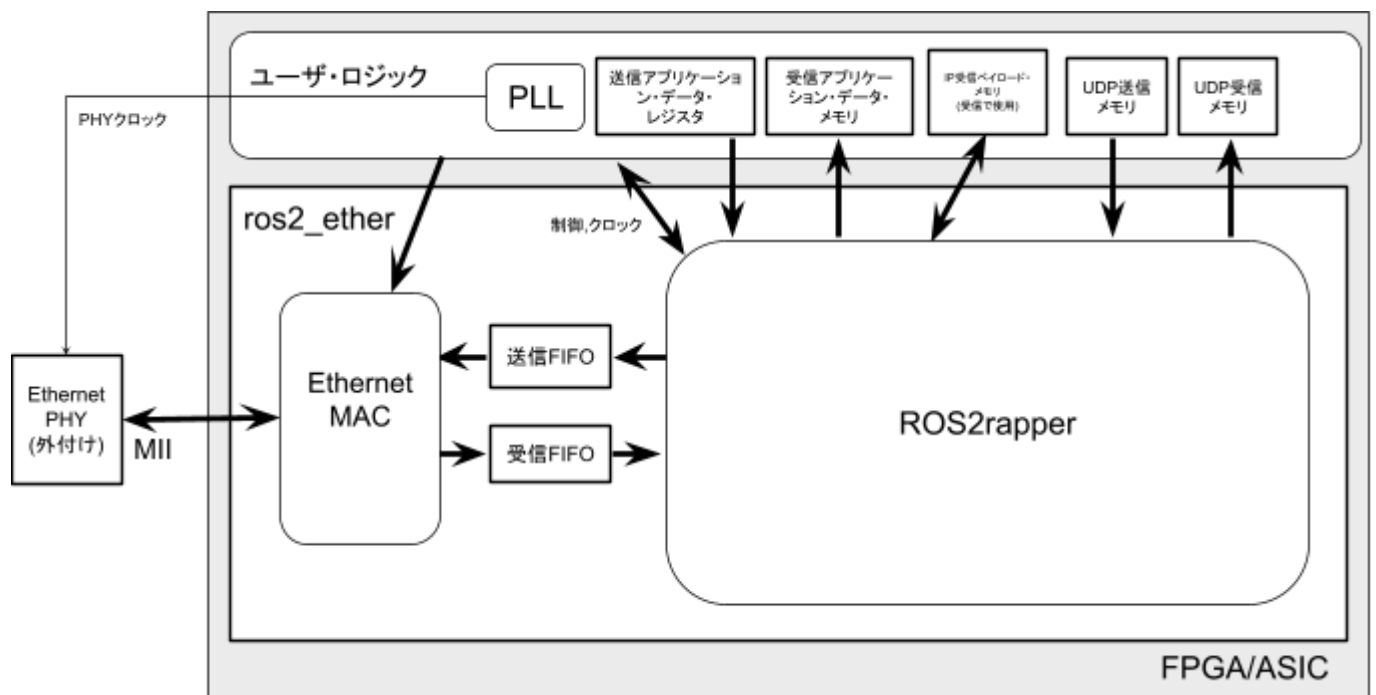


Fig.2.2 ROS2rapperのEthernet使用時のモジュール構成



以下に、主なモジュールの概要を記述する。

- ros2\_ether  
ros2\_etherは、下記のサブ・モジュールから成る。
  - ROS2rapper
    - RTPS/UDP/IP通信 各層の実装
  - Ethernet MAC
    - EthernetのMAC層  
[verilog-ethernet](https://github.com/alexforencich/verilog-ethernet)を使用  
<https://github.com/alexforencich/verilog-ethernet>
  - 送信FIFO
    - Ethernet MACへのEthernetフレームをバッファリングするためのFIFO
  - 受信FIFO
    - Ethernet MACからのEthernetフレームをバッファリングするためのFIFO
- アプリケーション・データ・レジスタ(送信)  
ROS2パブリッシャとして送信するデータを保持するレジスタまたはメモリ。ユーザ・ロジックで用意する。
- アプリケーション・データ・メモリ(受信)  
ROS2サブスクライバとして受信したデータを保持するメモリ。ROS2rapper IPから書き込まれる。ユーザ・ロジックで用意する。
- UDP送信メモリ  
UDPデータグラムとして送信するデータを保持するメモリ。ユーザ・ロジックで用意する。
- UDP受信メモリ  
受信したUDPデータグラムを保持するメモリ。ROS2rapper IPから書き込まれる。ユーザ・ロジックで用意する。
- IP受信ペイロード・メモリ  
受信したIPパケットのフラグメントを格納して、全フラグメントの到着を待ち合わせるためのIP受信用のメモリ。  
ユーザは、このメモリを用意し、接続しなければならない。
  - ユーザが用意すべきメモリの大きさは、バイト単位で、  
$$\text{MAX\_PENDINGS} \times \text{IP\_MAX\_PAYLOAD\_LEN} \times \text{MAX\_IP\_FRAGMENTS}$$
  
で計算できる。  
定数のそれぞれの意味は次の通り。
    - MAX\_PENDINGS、IP\_MAX\_PAYLOAD\_LEN、MAX\_IP\_FRAGMENTSの積で計算される。定数のそれぞれの意味は次の通り。
    - MAX\_PENDINGS: 同時にIPフラグメント組み立て待ちを行えるIPデータグラムの個数
    - IP\_MAX\_PAYLOAD\_LEN: 最大IP通信ペイロードサイズ
    - MAX\_IP\_FRAGMENTS: 最大IPフラグメント数
  - デフォルト値はTable A-2を参照のこと。
  - フラグメント化されないUDP受信のみ、または送信のみを行う場合は、このIP受信ペイロード・メモリは不要
- ユーザ・ロジック

ユーザの実現した機能を実装した論理であり、各モジュールの初期化や送受信メッセージの作成、送受信を行う。

- アプリケーション・データ・レジスタ(送信)
  - パブリッシュするメッセージの記憶域
- アプリケーション・データ・メモリ(受信)
  - サブスクライブしたトピックより受信したメッセージの記憶域

## 3. ROS2rapper IPとユーザ・ロジックの接続

### 3.1. 概要

ここでは、ユーザ・ロジックを作成する開発者向けに、本ROS2rapper IPとユーザの論理(ユーザ・ロジック)を接続するための開発について説明する。

ROS2rapper IPには外部からクロックを与えること。

ROS2rapper IPの入力ポートは信号線であるから、適宜ユーザ・ロジックなどにレジスタ(D Flip Flop)を設けるなどして値を保持して、ポートへの信号を安定供給しなければならない。

ポートに与える信号の変化は、ROS2rapperに与えるクロックに同期していること。

複数ビットからなるポートや信号線は、LSB 0(最下位ビットが、ビット番号0)である。

開発の概略手順は、以下である。

1. 次のVerilogヘッダファイルをインクルードする。
  - `ros2rapper/src/ros2rapper/include/ros2_config.vh`
  - `ros2rapper/src/ether/include/ros2_ether_config.vh`
2. `ros2_ether.v`にある`ros2_ether`をインスタンス化する。
  - その定義は`ros2rapper/src/ether/rtl/ros2_ether.v`にある
  - `ros2_ether`モジュールのポートとの接続、IP受信ペイロード・メモリとの接続を行う

次節以降ではFPGA(ASIC)へROS2rapperを接続する上での要点と、ユーザ・ロジック作成上の留意点やヒントとなる情報を記述する。

実装例として、`examples`ディレクトリ中の以下のコードを参照のこと。

- `topic-pub-sub/top.v` : トピックのパブリッシュ/サブスクライブの実装サンプル
- `udp-tx-rx/top.v` : UDPパケット送受信の実装サンプル

付録には以下の技術情報や、参考情報へのリンクを記述する。

- ROS2rapperのIPで、Fig2.2に示した `ros2_ether`モジュールのポート仕様
- 同ポート使用一覧表中の定数と設定の注意事項
- 参考文献、特に以下の技術情報へのリンク
  - ノード名・トピック名・トピック型名の制限: ROS2やRTPSの仕様書
  - アプリケーションデータのシリアル化仕様: OMGの仕様書

## 3.2. ROS2rapperのクロック、機能の有効化など

### 3.2.1 ROS2rapperのクロック

ROS2rapper IPへのクロック clkは、外部から供給する。  
クロック周波数は、50MHz以上が望ましい。

### 3.2.2 ROS2rapperのリセット

ROS2rapper IPのリセットはrst\_nを0にすることで行われる。  
rst\_nは非同期でよい。

### 3.2.2 ROS2rapper各機能の有効化

先述のとおり、本ROS2rapper IPは、下記の各機能を独立して個々に有効化/無効化できる。

- ROS2パブリッシュ機能, ros2sub\_enにより制御
- ROS2サブスクライブ機能, ros2pub\_enにより制御
- Ethernet MAC, ether\_enにより制御

パブリッシュとサブスクライブは独立して個々に、または同時に使用できる

ROS2rapperの各機能を有効化するには、順序がある。

後述する各機能のポートの設定を行った上で、

物理通信手段(本実装サンプルではEther MAC)を有効化した後に、

ROS2のパブリッシュ、あるいはサブスクライブの機能の有効化を行わなければならない。

Table 3.1. ROS2rapperシステム・ポートの設定例

ROS2rapperクロック/リセット			
ポート名	方向 (ROS2rapper から見て)	ビット幅	説明
clk	input	1	クロック(50MHz以上)
rst_n	input	1	リセット(0でアサート)
ros2pub_en	input	1	トピックのパブリッシュを有効にする(1でアサート)
ros2sub_en	input	1	サブスクライブを有効にする(1でアサート)
ether_en	input	1	Ethernet MACを有効にする(1でアサート)。ether_enを一度アサートした後、デアサート(0)してはならない

### 3.3. Ethernetモジュールとの接続

ethernetモジュールとの接続は下記の2つがある

- Ethernet MAC
- PHY

### 3.3.1. Ethernet MACとの接続

下記のポートはEthernet MACを制御する。

ユーザ・ロジックを各ポートに接続して、使用するネットワーク環境に応じた設定値を与える。

Ethernet MACを有効化するには、ポートのether\_enをアサート(1)する。

ether\_enは一度アサートしたら、デアサート(0)してはならない。

- Ethernet・IPプロトコル  
ether\_en  
mac\_addr  
ip\_addr  
gateway\_ip\_addr  
subnet\_mask
- ARP  
arp\_req\_retry\_count  
arp\_req\_retry\_interval  
arp\_req\_timeout

実装サンプルではROS2rapperは通信にEthernet MACを使用するため、Ethernet MACに適切な値を与えている。

設定例をTable 3.2.に記述する。

Table 3.2. Ethernet MAC関連のポートの設定例

ポート名	設定値の例	説明
ether_en	1	Ethernet MAC有効(すべての設定値が与えられてからアサート(1)すること。アサート後にデアサート(0)してはならない)
mac_addr	48'h00_00_00_00_00_02	6バイト
ip_addr	{8'd100, 8'd1, 8'd168, 8'd192}	192.168.1.100の場合
gateway_ip_addr	{8'd1, 8'd1, 8'd168, 8'd192}	192.168.1.1の場合
subnet_mask	{8'd0, 8'd255, 8'd255, 8'd255}	255.255.255.0の場合
arp_req_retry_count	4	ARPの試行の最大回数
arp_req_retry_interval	(125000000*2)	単位:クロック数
arp_req_timeout	(125000000*30)	単位:クロック数

### 3.3.2. PHYの接続

クロック、リセット、Ethernet PHYの各端子とは以下のポートで接続する。  
PHYのリファレンス・クロックは、ユーザ・ロジックで25MHzを生成して供給すること。

Table 3.3. クロック、リセット、Ethernet PHYのポート

Ethernet MII			
ポート名	方向 (ROS2rapper から見て)	ビット幅	説明
phy_rx_clk	input	1	Ethernet MII 受信クロック
phy_rxd	input	4	Ethernet MII 受信データ
phy_rx_dv	input	1	Ethernet MII受信データ有効
phy_rx_er	input	1	Ethernet MII 受信エラー
phy_tx_clk	input	1	Ethernet MII 送信クロック
phy_txd	output	4	Ethernet MII 送信データ
phy_tx_en	output	1	Ethernet MII 送信イネーブル
phy_rst_n	output	1	Ethernet MII リセット

### 3.3.3. パケット送信間隔パラメータの設定

ros2rapper\_etherモジュールのパラメータで、パケット送信間隔を設定できる。パラメータの一覧はTable A-2を参照のこと。

RTPSの制御パケットの送信間隔は、通信相手検出までの時間やネットワーク帯域に応じて変更すること。

アプリケーション・データのパケットの送信間隔は、ROS2rapperのトピックをサブスクライブしたノードにメッセージが届く間隔である。アプリケーションに応じて変更すること。

### 3.4. ROS2rapperのポートの設定

ROS2rapperの制御は、ユーザ・ロジックを、下に示すポートに接続し、適切な値を与えることで行う。

Table 3.4. ROS2rapperのポートの設定例

ポート名	設定値の例	説明
ros2_rx_udp_port	1234	UDP受信したいポート番号
ros2_node_udp_port	52000	
ros2_port_num_seed	7400	計算式: (ROS2のDomain ID)*250+7400
ros2_guid_prefix	0x010f37adde09000001000000	
ros2_node_name_len	19	単位: バイト数 ('\0'を含める)
ros2_pub_topic_name_len	11	単位: バイト数 ('\0'を含める)
ros2_pub_topic_type_name_len	29	単位: バイト数 ('\0'を含める)
ros2_pub_app_data_len	25	単位: バイト数
ros2_node_name	"elpmaxe_reppar2sor"	
ros2_pub_topic_name	"rettahc/tr"	
ros2_pub_topic_type_name	"_gnirtS::_sdd::gsm::sgsm_dts"	
ros2_pub_app_data	"!dlrow reppar2SOR ,olleh"	



### 3.5. アプリケーション・データ・レジスタ(送信)の用意と接続

アプリケーション・データ・レジスタは、ROS2rapperがパブリッシュしたトピックとして送信するデータを保持するレジスタである。

ROS2rapper IPIは、パブリッシャ動作時に、外部へ送出するメッセージをアプリケーション・データ・レジスタから得る。

アプリケーション・データ・レジスタは、ユーザ・ロジックとして、メモリまたはD-FFで実現して用意し接続する。

アプリケーション・データ・レジスタは、ポートros2\_pub\_app\_dataと接続する。

なお、アプリケーション・データ・レジスタ(ros2\_pub\_app\_dataの値)の変更には、後述の手順が必要である。

送信メッセージros2\_pub\_app\_dataの、データの並びは次のとおり

- 1バイト中のビットの並び: LSB 0
- バイト単位で送出される
  - データを整数とした場合のバイト・オーダー: Little Endian固定

ros2\_pub\_app\_dataにセットするアプリケーション・データは下記の順序とする。

- ros2\_pub\_app\_data[7:0] が アプリケーション・データの1バイト目
- ros2\_pub\_app\_data[15:8] が アプリケーション・データの2バイト目
- ...

#### ※注意

データが文字列である場合、user\_defined\_reg[63:56]を文字列の先頭として、

ros2\_pub\_app\_data[63:0]=user\_defined\_reg[63:0] のように接続すると、送出されるデータは普通人が期待するものとは逆順となってしまう。よって、

```
ros2_pub_app_data[63:0]= { user_defined_reg[7:0],  
                           user_defined_reg[15:8],  
                           user_defined_reg[23:16],  
                           :  
                           user_defined_reg[63:56] }
```

のように、するとよい。

### 3.6. アプリケーション・データ・メモリの用意と接続

アプリケーション・データ・メモリは、ROS2rapperがサブスクライブしたトピックから受信したメッセージを保持するRAMである。

ROS2rapper IPIは、サブスクライバ動作時に、外部から受信したメッセージをアプリケーション・データ・メモリに格納する。

先述のとおり、アプリケーション・データ・メモリは、ユーザ・ロジックとして、メモリまたはD-FFで実現して用意し接続する。

アプリケーション・データ・メモリは、ポート`ros2_sub_app_data_{addr, ce, we, wdata}`と接続する。

なお、アプリケーション・データ・メモリの読み出しには、後述の手順が必要である。

アプリケーション・データ・メモリは、以下の仕様とすること。

- ポート数: 1RW
- データ幅: 8ビット
- アドレス幅:  $\lceil \log_2(\text{ROS2\_MAX\_APP\_DATA\_LEN}) \rceil$  ビット
- ワード数: `ROS2_MAX_APP_DATA_LEN`
- クロックに同期して読み書きを行う
- 以下のポートを持つ
  - `ce`: チップイネーブル(1でアサート)
  - `we`: ライト・イネーブル(1でアサート)
  - `addr`: アドレス
  - `wdata`: 書き込みデータ
  - `rdata`: 読み出しデータ(出力)
- `ce`がアサート(1)されているとき、アドレス`addr`のデータを`rdata`に出力する。
- `ce`がアサート(1)されていて、かつ`we`がアサート(1)されているとき、アドレス`addr`に`wdata`の値を書き込む。

### 3.7. IP受信ペイロード・メモリの用意と接続

IP受信ペイロード・メモリは、ROS2機能有効時に受信したIPフラグメント組み立てに使用する。  
IP受信ペイロード・メモリのサイズは、2960バイト程度が望ましい。

- IP受信ペイロード・メモリは、ポートip\_payloadsmem\_{addr, ce, we, wdata, rdata}と接続する。
- IP受信ペイロード・メモリは、以下の仕様とすること。
  - ポート数: 1RW
  - データ幅: 8ビット
  - アドレス幅:  $\lceil \log_2(\text{PAYLOADSMEM\_DEPTH}) \rceil$  ビット
  - ワード数: PAYLOADSMEM\_DEPTH
  - クロックに同期して読み書きを行う
  - 以下のポートを持つ
    - ce: チップイネーブル(1でアサート)
    - we: ライト・イネーブル(1でアサート)
    - addr: アドレス
    - wdata: 書き込みデータ
    - rdata: 読み出しデータ(出力)
  - ceがアサート(1と)されているとき、アドレスaddrのデータをrdataに出力する。
  - ceがアサート(1と)されていて、かつweがアサート(1と)されているとき、アドレスaddrにwdataの値を書き込む。

### 3.8. UDP送信メモリの用意と接続

UDP送信メモリは、ROS2rapperが送信するUDPデータグラムを保持するRAMである。  
ROS2rapper IPIは、udp\_txbuf\_reiがアサート(1と)されると、UDP送信メモリ内のデータグラムを送信する。

UDP送信メモリは、ユーザ・ロジックとして、メモリまたはD-FFで実現して用意し接続する。  
UDP送信メモリは、ポートudp\_txbuf\_{addr, ce, we, rdata}と接続する。  
なお、UDP送信メモリへの書き込みには、後述の手順が必要である。

アプリケーション・データ・メモリは、以下の仕様とすること。

- ポート数: 1RW
- データ幅: 32ビット
- アドレス幅:  $\lceil \log_2(\text{ROS2\_MAX\_APP\_DATA\_LEN}) \rceil$  ビット
- ワード数:  $2^{\text{UDP\_TXBUF\_AWIDTH}}$
- クロックに同期して読み書きを行う
- 以下のポートを持つ
  - ce: チップイネーブル(1でアサート)
  - we: ライト・イネーブル(1でアサート)
  - addr: アドレス
  - wdata: 書き込みデータ
  - rdata: 読み出しデータ(出力)
- ceがアサート(1)されているとき、アドレスaddrのデータをrdataに出力する。
- ceがアサート(1)されていて、かつweがアサート(1)されているとき、アドレスaddrにwdataの値を書き込む。

### 3.9. UDP受信メモリの用意と接続

UDP受信メモリは、設定されたUDP受信ポートへ届いたデータグラムを保持するRAMである。ROS2rapper IPIは、外部から受信したUDPデータグラムをUDP受信メモリに格納する。

UDP受信メモリは、ユーザ・ロジックとして、メモリまたはD-FFで実現して用意し接続する。

UDP受信メモリは、ポート`udp_rxbuf_{addr, ce, we, wdata}`と接続する。

なお、UDP受信メモリの読み出しには、後述の手順が必要である。

アプリケーション・データ・メモリは、以下の仕様とすること。

- ポート数: 1RW
- データ幅: 32ビット
- アドレス幅:  $\lceil \log_2(\text{ROS2\_MAX\_APP\_DATA\_LEN}) \rceil$  ビット
- ワード数:  $2^{\text{UDP\_RXBUF\_AWIDTH}}$
- クロックに同期して読み書きを行う
- 以下のポートを持つ
  - `ce`: チップイネーブル(1でアサート)
  - `we`: ライト・イネーブル(1でアサート)
  - `addr`: アドレス
  - `wdata`: 書き込みデータ
  - `rdata`: 読み出しデータ(出力)
- `ce`がアサート(1)されているとき、アドレス`addr`のデータを`rdata`に出力する。
- `ce`がアサート(1)されていて、かつ`we`がアサート(1)されているとき、アドレス`addr`に`wdata`の値を書き込む。

## 4. ROS2通信

### 4.1. ROS2rapper各機能の有効化

先述のとおり、本ROS2rapper IPは、下記の各機能を独立して個々に有効化/無効化できる。

- ROS2パブリッシュ機能
- ROS2サブスクライブ機能
- Ethernet

ROS2rapperの機能を有効化する前に、Ethernet MACなど物理層の有効化や初期化が必要である。

つまり、本実装サンプルでは先述のポートの設定を行った上で、Ether MAC(物理通信手段)を有効化した後、ROS2rapperのROS2のパブリッシュ、あるいはサブスクライブ機能の有効化を行う。

機能の有効化は次のとおり。

- トピックのパブリッシュを有効にする(すなわちパブリッシャとして使用する)には `ros2pub_en` を、アサート(1と)する。
- サブスクライブを有効にする(すなわちサブスクライバとして使用する)には `ros2sub_en` を、それぞれアサート(1と)する。
- パブリッシュとサブスクライブは独立に個々に、または同時に使用できる
- Ethernet MACを有効化するには、ポートの `ether_en` をアサート(1と)する。
  - `ether_en` を一度アサートしたら、デアサート(0と)してはならない。

なお、サンプル実装では、`ros2_ether`と接続するポートはレジスタとしてユーザ・ロジックの中に実装して、接続している。

## 4.2. パブリッシャの送信メッセージ(アプリケーション・データ・レジスタ)の変更

パブリッシャ有効時には、ユーザ・ロジック内に用意された、アプリケーション・データ・レジスタからのデータを、ROS2送信メッセージとして自動的に周期的に送出する。  
アプリケーション・データ・レジスタの出力は、ポートros2\_pub\_app\_dataの入力としなければならない。

送信内容(ros2\_pub\_app\_dataの入力)を変更するためには、後述の手続きが必要である。

ユーザ・ロジックは、ros2\_pub\_app\_dataの入力を変更する前に、ROS2rapperより「メッセージ変更権」を取得する必要がある。これは、変更途中のメッセージがROS2rapperにより読み出され、送信されることを防ぐためである。

下記のポートが、メッセージ変更権の制御のために用意されている。

- ros2\_pub\_app\_data\_req(ROS2rapperの入力): メッセージ変更権を要求する
- ros2\_pub\_app\_data\_grant(ROS2rapperからの出力): メッセージ変更権が与えられた
- ros2\_pub\_app\_data\_rel(ROS2rapperの入力): メッセージ変更権を解放

送信メッセージ(ros2\_pub\_app\_data入力)変更時のポート操作手順を以下に記述する。

1. ros2\_pub\_app\_data\_reqをアサート(1と)して、メッセージ変更権を要求する
2. メッセージ変更権を取得できた場合、次のサイクルでros2\_pub\_app\_data\_grantがアサート(1と)される。
3. 当該信号がアサートされていなければ1.へ戻る
4. メッセージの変更を行う。
5. ros2\_pub\_app\_data\_relを1クロックの間アサート(1と)し、メッセージ変更権を解放する。

送信アプリケーション・データ・レジスタに書き込むアプリケーション・データのフォーマットは、Extensible and Dynamic Topic Types for DDS 仕様(参考文献[2])の、Classic CDR representation with Little Endian encodingに従うこと。エンコーディングの詳細は、参考文献[2] 7.4.3.5節のエンコーディング・ルールを参照すること。

1バイト中のビットの並びはLSB 0である。バイト・オーダーはリトル・エンディアンでなければならない。

例) 32ビット整数x2(0x12345678, 0xaabbaabb)を含むアプリケーション・データ

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
78	56	34	12	bb	aa	bb	aa

32ビット整数 "0x12345678"

32ビット整数 "0xaabbaabb"

例) 文字列"hello, world!"を含むアプリケーション・データ

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte 16
0d	00	00	00	68	65	6c	6c	6f	2c	20	77	6f	72	6c	64	21

文字列長 13

文字列 "hello, world!"



### 4.3. サブスクライバの受信メッセージ(アプリケーション・データ・メモリ)の読み出し

サブスクライバ機能は、サブスクライブしたデータを受信し、メモリ・インターフェースを通して、ユーザ・ロジック内に用意されたアプリケーション・データ・メモリに受信データを書き込む。

受信したメッセージを読み出す際に「メッセージ読み出し権」の取得が必要となる。これは、ユーザ・ロジックのメッセージ読み出し中に、ROS2rapper IPが、次の受信メッセージでメモリを上書きしないための手順である。

下記のポートが、メッセージ変更権の制御のために用意されている。

- `ros2_sub_app_data_req`(ROS2rapperの入力): メッセージ読み出し権を要求する
- `ros2_sub_app_data_grant`(ROS2rapperからの出力): メッセージ読み出し権が与えられた
- `ros2_sub_app_data_re`(ROS2rapperの入力): メッセージ読み出し権を解放

受信メッセージ読み出し時のポート操作手順を以下に述べる。

1. `ros2_sub_app_data_req`をアサート(1と)する。
2. メッセージ読み出し権を取得できた場合、次のサイクルで`ros2_sub_app_data_grant`がアサート(1と)される。
3. 当該信号がアサートされていなければ1.へ戻る
4. メッセージ読み出しを行う。
5. `ros2_sub_app_data_re`を1クロックの間アサート(1と)し、メッセージ読み出し使用権を解放する。

受信アプリケーション・データ・メモリに受信したアプリケーション・データのフォーマットおよびバイト・オーダーは、`ros2_sub_app_data_rep_id`ポートより出力される "Representation Identifier" の値で判定する。Representation Identifierの値については、参考文献[3]のTable 10.3を参照すること。アプリケーション・データのフォーマットについては、参考文献[2] 7.4節を参照すること。

## 5. UDPデータグラム通信

### 5.1. UDPデータグラムの送信

ROS2rapperでは、シンプルなUDPデータグラムの送信を行うことができる。

下記のポートが、UDPデータグラム送信のために用意されている。

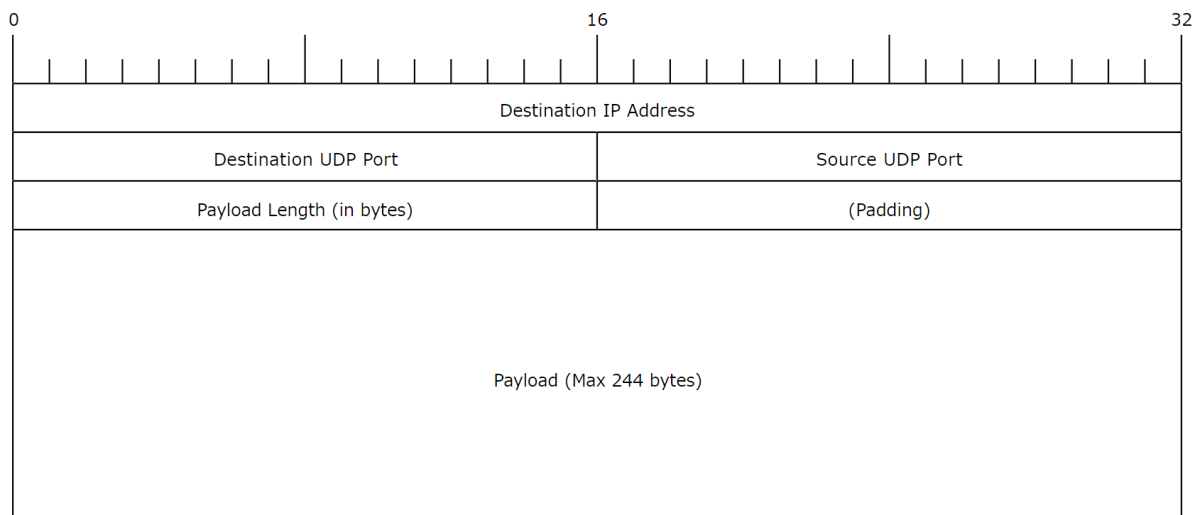
- `udp_txbuf_grant`(ROS2rapperからの出力): UDP送信メモリが空になったとき、アサート(1と)される
  - このとき、UDP送信メモリに送信データを置くことが可能である
- `udp_txbuf_rel`(ROS2rapperの入力):UDP送信メモリ中のUDPデータグラムの送信開始
  - 1クロックの間アサート(1と)する。

`udp_txbuf_grant`がデアサート(0)のときはUDP送信メモリのデータを更新してはならない。

以下、手順を述べる。

1. ポート:`udp_txbuf_grant`が1になるまで待つ。
2. UDP送信メモリに送信するUDPデータグラムを書き込む。
3. ポート:`udp_txbuf_rel`を1クロックの間アサートすると、送信が開始される。

UDP送信メモリ中のデータの形式をFig. 4.1.に示す。



なお、Payload Lengthフィールドの値は、244以下でなければならない。

Fig. 4.1. UDP送信メモリ中のデータの形式

## 5.2. UDPデータグラムの受信

ROS2rapperでは、シンプルなUDPデータグラムの受信を行うことができる。

下記のポートが、UDPデータグラム受信のために用意されている。

- `udp_rxbuf_grant`(ROS2rapperからの出力): UDP受信メモリにデータを置いた後、アサート(1と)する
  - このとき、UDP受信メモリを読出すことが可能である
- `udp_rxbuf_rel`(ROS2rapperの入力): 新しいUDPデータグラムの受信可能にする。
  - 1クロックの間アサート(1と)する。

以下、手順を述べる。

1. UDPデータグラムが受信され、UDP受信メモリに置かれると、`udp_rxbuf_grant`がアサート(1と)される
2. ユーザ・ロジックは、UDP受信メモリからデータを読出す
3. バッファからの読出しが完了したら、新しいUDPデータグラムを受信可能にするため、ユーザ・ロジックは、`udp_rxbuf_rel`を1クロックの間アサート(1と)する。

UDP受信メモリに置かれるデータの形式をFig. 4.2. に示す。

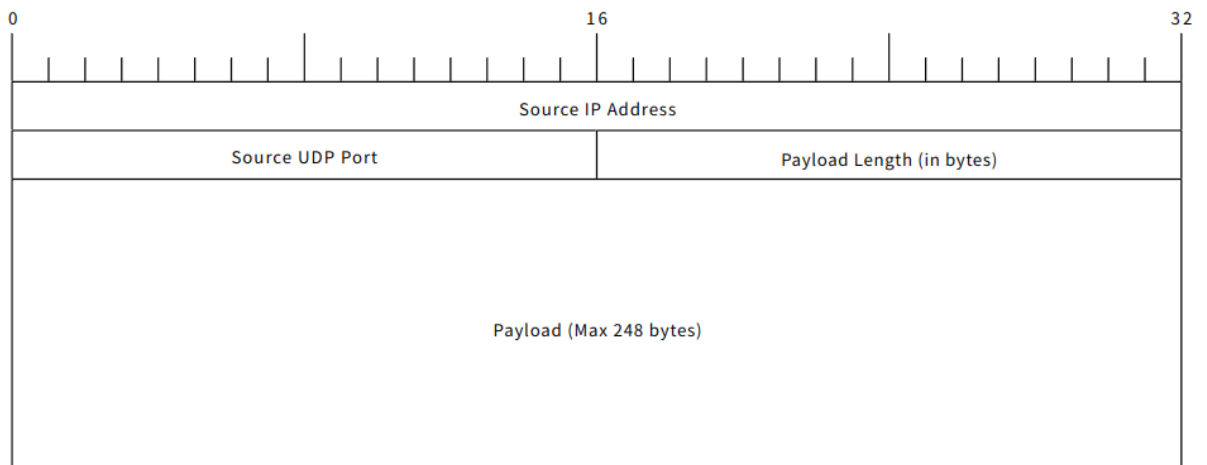


Fig. 4.2. UDP受信メモリ中のデータの形式

## 6. 参考文献

1. Topic and Service name mapping to DDS  
([https://design.ros2.org/articles/topic\\_and\\_service\\_names.html#mapping-of-ros-2-topic-and-service-names-to-dds-concepts](https://design.ros2.org/articles/topic_and_service_names.html#mapping-of-ros-2-topic-and-service-names-to-dds-concepts))
2. Extensible and Dynamic Topic Types for DDS  
(<https://www.omg.org/spec/DDS-XTypes/1.3/PDF>)
3. The Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (DDSI-RTPSTM) Specification Version 2.3  
(<https://www.omg.org/spec/DDSI-RTPS/2.3/PDF>)

## 7. 付録

### 7.1. ros2\_etherモジュールのインターフェース仕様

ros2\_etherモジュールのポート一覧をTable A-1に示す。

- 各信号は、LSB 0である。
- 接尾辞が\_nの信号は負論理、それ以外は正論理である。
  - 正論理の信号は、アサートで信号を1に、デアサートで信号を0にする。
  - 負論理の信号は、アサートで信号を0に、デアサートで信号を1にする。

Table A-1 ros2\_etherモジュールのポート仕様一覧

ポート名	方向	ビット幅	説明
クロック・リセット			
clk	input	1	クロック
rst_n	input	1	リセット(0でアサート)
機能有効化			
ether_en	input	1	Ethernet MACイネーブル 1: Ethernet有効 0: Ethernet無効 1に設定後、再び0に設定した場合の動作は保証されない。
ros2pub_en	input	1	トピック・パブリッシュ・イネーブル 1: トピック・パブリッシュ有効 0: トピック・パブリッシュ無効 1に設定後、再び0に設定した場合の動作は保証されない。
ros2sub_en	input	1	トピック・サブスクライブ・イネーブル 1: トピック・サブスクライブ有効 0: トピック・サブスクライブ無効 1に設定後、再び0に設定した場合の動作は保証されない。
Ethernet MII			
phy_rx_clk	input	1	Ethernet MII 受信クロック
phy_rxd	input	4	Ethernet MII 受信データ
phy_rx_dv	input	1	Ethernet MII受信データ有効
phy_rx_er	input	1	Ethernet MII 受信エラー
phy_tx_clk	input	1	Ethernet MII 送信クロック
phy_txd	output	4	Ethernet MII 送信データ
phy_tx_en	output	1	Ethernet MII 送信イネーブル
phy_rst_n	output	1	Ethernet MII リセット

Ethernetアドレス/IPアドレス			
mac_addr	input	48	MACアドレス
ip_addr	input	32	IPアドレス a.b.c.dを設定する場合、ip_addr[31:0] = {d,c,b,a}とする。
gateway_ip_addr	input	32	ゲートウェイIPアドレス
subnet_mask	input	32	サブネットマスク
ros2_fragment_expiration	input	32	IPフラグメントがすべて揃わない際のタイムアウト時間。 ROS2rapper内部サイクル数で指定する。
ARP			
arp_req_retry_count	input	6	ARPリトライ回数
arp_req_retry_interval	input	36	ARPリトライ間隔(サイクル数)
arp_req_timeout	input	36	ARPタイムアウト時間(サイクル数)
UDP			
ros2_rx_udp_port	input	16	UDP受信ポート番号 ここで指定したポート番号と一致するUDPデータグラムが受信バッファに受信される。
ROS2			
ros2_node_name	input	ROS2_MAX_NODE_NAME_LEN*8	ノード名 '\0'終端とし、終端以降はゼロで埋めること。
ros2_node_name_len	input	8	ノード名の長さ '\0'を含めたバイト数で指定すること。
ros2_node_udp_port	input	16	ノードのUDPポート番号
ros2_port_num_seed	input	16	UDPポート番号のシード値 ROS2で使用するポート番号導出のためのマジックナンバーであり次の計算式で得られる。 (ROS2のDomain ID)*250+7400
ros2_guid_prefix	input	96	GUID(グローバル一意識別子) プレフィクス
パブリッシュするトピック			
ros2_pub_topic_name	input	ROS2_MAX_TOPIC_NAME_LEN*8	トピック名 '\0'終端とし、終端以降はゼロで埋めること。
ros2_pub_topic_name_len	input	8	トピック名の長さ '\0'を含めたバイト数で指定すること。
ros2_pub_topic_type_name	input	ROS2_MAX_TOPIC_TYPE_NAME_LEN*8	トピック型名 '\0'終端とし、終端以降はゼロで埋めること。

ros2_pub_topic_type_name_len	input	8	トピック型名の長さ '\0'を含めたバイト数で指定すること。
ros2_pub_app_data	input	ROS2_MAX_APP_DATA_LEN*8	メッセージ データ自体の終端以降はゼロで埋めること。
ros2_pub_app_data_len	input	8	メッセージの長さ バイト数で指定すること。
ros2_pub_app_data_req	input	1	メッセージ変更権リクエスト 1を書き込むと、メッセージ変更権をリクエストする。
ros2_pub_app_data_rel	input	1	メッセージ変更権開放 1を書き込むと、メッセージ変更権を解放する。
ros2_pub_app_data_grant	output	1	メッセージ変更権 0: メッセージ書き込み不許可 1: メッセージ書き込み許可
サブスクライブするトピック			
ros2_sub_topic_name	input	ROS2_MAX_TOPIC_NAME_LEN*8	トピック名 '\0'終端とし、終端以降はゼロで埋めること。
ros2_sub_topic_name_len	input	8	トピック名の長さ '\0'を含めたバイト数で指定。
ros2_sub_topic_type_name	input	ROS2_MAX_TOPIC_TYPE_NAME_LEN*8	トピック型名 '\0'終端とし、終端以降はゼロで埋めること。
ros2_sub_topic_type_name_len	input	8	トピック型名の長さ '\0'を含めたバイト数で指定。
ros2_sub_app_data_len	output	8	受信したメッセージの長さ
ros2_sub_app_data_rep_id	output	16	受信したメッセージのRepresentation Identifier Representation Identifierについては、参考文献[2] Table 10.3を参照のこと。
ros2_sub_app_data_req	input	1	メッセージ読み出し権リクエスト 1を書き込むと、メッセージ読み出し権をリクエストする。
ros2_sub_app_data_rel	input	1	メッセージ読み出し権解放 1を書き込むと、メッセージ読み出し権を解放する。
ros2_sub_app_data_grant	output	1	メッセージ読み出し権 0: メッセージ読み出し不許可 1: メッセージ読み出し許可
ros2_sub_app_data_recv	output	1	メッセージ・データ受信通知
ros2_sub_app_data_addr	output	$\lceil \log_2(\text{ROS2\_MAX\_APP\_DATA\_LEN}) \rceil$	アプリケーション・データ・メモリのアドレス

		_APP_DATA_LEN)	ス ユーザ・ロジックに確保したメモリのアド レス
ros2_sub_app_data_ce	output		1 アプリケーション・データ・メモリのチップ イネーブル
ros2_sub_app_data_we	output		1 アプリケーション・データ・メモリのライ ト・イネーブル
ros2_sub_app_data_wdata	output		8 アプリケーション・データ・メモリの書き 込みデータ
<b>UDP受信メモリ</b>			
udp_rxbuf_addr	output	UDP_RXBUF_AWI DTH	UDP受信メモリ・メモリのアドレス
udp_rxbuf_ce	output		1 UDP受信メモリのチップイネーブル
udp_rxbuf_we	output		1 UDP受信メモリのライト・イネーブル
udp_rxbuf_wdata	output		32 UDP受信メモリの読み出しデータ
udp_rxbuf_rel	input		1 UDP受信メモリの読み出し権解放
udp_rxbuf_grant	output		1 UDP受信メモリの読み出し権
<b>UDP送信メモリ</b>			
udp_txbuf_addr	output	UDP_TXBUF_AWID TH	UDP送信メモリ・メモリのアドレス
udp_txbuf_ce	output		1 UDP送信メモリのチップイネーブル
udp_txbuf_rdata	input		32 UDP送信メモリの読み出しデータ
udp_txbuf_rel	input		1 UDP送信メモリの読み出し使用権開放
udp_txbuf_grant	output		1 UDP送信メモリの読み出し使用権
<b>IP受信ペイロード・メモリ</b>			
ip_payloadsmem_addr	output	PAYLOADSMEM_A WIDTH	IP受信ペイロード・メモリのアドレス
ip_payloadsmem_ce	output		1 IP受信ペイロード・メモリのチップイネー ブル
ip_payloadsmem_we	output		1 IP受信ペイロード・メモリのライト・イ ネーブル
ip_payloadsmem_wdata	output		8 IP受信ペイロード・メモリの書き込み データ
ip_payloadsmem_rdata	input		8 IP受信ペイロード・メモリの読み出し データ

Table A-2 パラメーター一覧



パラメータ名	説明	デフォルト値
PRESCALAR_DIV	パケット送信間隔カウンタの分周比	64
TX_INTERVAL_COUNT	パケット間送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) / 100$ (10msec)
TX_PERIOD_SPDP_WR_COUNT	SPDP Writerのパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_PUB_WR_COUNT	SEDP Publication WriterのDATAパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_SUB_WR_COUNT	SEDP Subscription WriterのDATAパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_PUB_HB_COUNT	SEDP Publication WriterのHEARTBEATパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_SUB_HB_COUNT	SEDP Subscription WriterのHEARTBEATパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_PUB_AN_COUNT	SEDP Publication WriterのACKNACKパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_SEDP_SUB_AN_COUNT	SEDP Subscription WriterのACKNACKパケット送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)
TX_PERIOD_APP_WR_COUNT	アプリケーション・データ(Application WriterのDATAパケット)送信間隔 (ROS2rapperのクロック数)	$(\text{`ROSCLK\_HZ} / \text{PRESCALAR\_DIV}) * 3$ (3sec)

Table A-3 定数一覧

定数名	説明	デフォルト値	定義されている場所 (*1)	変更時の注意点
ROS2_MAX_NODE_NAME_LEN (MAX_NODE_NAME_LEN)	ROS2ノード名の最大長	32	include/ros2_config.vh (hls/ros2.hpp)	これらの定数は、VerilogとHLS C++のヘッダに存在し、定数名が異なる。(・)に記述したものはHLS C++のヘッダとその所在である。これらを同時に変更す
ROS2_MAX_TOPIC_NAME_LEN (MAX_TOPIC_NAME_LEN)	ROS2トピック名の最大長	32	include/ros2_config.vh (hls/ros2.hpp)	

定数名	説明	デフォルト値	定義されている場所 (*1)	変更時の注意点
				必要がある。
ROS2_MAX_TOPIC_TYPE_NAME_LEN (MAX_TOPIC_TYPE_NAME_LEN)	ROS2トピック 型名の最大 長	64	include/ros2_config .vh (hls/ros2.hpp)	
ROS2_MAX_APP_DATA_LEN (MAX_APP_DATA_LEN)	ROS2メッ セージ・デー タの最大長	64	include/ros2_config .vh (hls/ros2.hpp)	
TARGET_PARTICIPANT_ID	ROS2ノード のParticipant ID	1	hls/common.hpp	
PAYLOADSMEM_DEPTH	IP受信ペイ ロード・メモリ のワード数	2960	include/ros2_config .vh	
PAYLOADSMEM_AWIDTH	IP受信ペイ ロード・メモリ のアドレス幅	$\$clog2(\text{PAYLOADSMEM\_DEPTH})$	include/ros2_config .vh	
MAX_PENDINGS	同時にIPフラ グメント組み 立て待ちを行 えるIPデータ グラムの個数	1	hls/ip.hpp	
IP_MAX_PAYLOAD_LEN	最大IP受信 ペイロード・メ モリサイズ	1480	hls/ip.hpp	1480=MTU(1500) - IPヘッダサイズ(20)
MAX_IP_FRAGMENTS	最大IP受信 ペイロードフ ラグメント数	2	hls/ip.hpp	
UDP_RXBUF_AWIDTH	UDP受信メモ リのアドレス 幅	6	ros2rapper/src/ethe rinclude/ros2_ether _config.vh	
UDP_TXBUF_AWIDTH	UDP送信メモ リのアドレス 幅	6	ros2rapper/src/ethe rinclude/ros2_ether _config.vh	
MAC_TX_FIFO_DEPTH	MAC内送信 FIFOの深さ	512	ros2rapper/src/ethe rinclude/ros2_ether _config.vh	2のべきでなければな らない

定数名	説明	デフォルト値	定義されている場所 (*1)	変更時の注意点
MAC_RX_FIFO_DEPTH	MAC内受信FIFOの深さ	2048	ros2rapper/src/ether rinclude/ros2_ether _config.vh	2のべきでなければなら ない
EXT_TX_FIFO_DEPTH	送信FIFOの 深さ	2	ros2rapper/src/ether rinclude/ros2_ether _config.vh	2のべきでなければなら ない
EXT_RX_FIFO_DEPTH	受信FIFOの 深さ	2048	ros2rapper/src/ether rinclude/ros2_ether _config.vh	2のべきでなければなら ない
ARP_CACHE_ADDR_WIDTH	ARPキャッ シュのアドレ ス幅	4	ros2rapper/src/ether rinclude/ros2_ether _config.vh	
ROS2CLK_HZ	ROS2rapper のクロック周 波数	100000000	include/ros2_config .vh	ROS2rapperのクロッ ク周波数に合わせて 変更する

注)

\*1 パスは、トップ・ディレクトリからの記述の無いものは、ros2rapper/src/ros2rapper/ 以下を記述

## 7.2. ディレクトリ構成

```

ros2rapper
|
|- src
|   |- ros2rapper : ROS2rapper
|   |   |- hls      : HLS C++記述
|   |   |- include  : Verilog ヘッダファイル
|   |   |- rtl      : Verilog RTL
|   |   |- test     : C-Sim用テストベンチ
|   |- ether : Ethernet層
|   |   |- include : Verilog ヘッダファイル
|   |   |- rtl     : Verilog RTL
|   |   |- verilog-ethernet : Ethernet MACのVerilog RTL
|- examples
|   |- topic-pub-sub : トピックのパブリッシュ/サブスクライブの実装サンプル
|   |- udp-tx-rx     : UDPパケット送受信の実装サンプル
|

```

- formal: SymbiYosysを使用したフォーマル検証用のスクリプト
- |
- README.md

### 7.3. RTPS通信で使用するUDPポート番号 一覧

RTPS通信では、下記のUDPポート番号を受信に使用する。したがって、UDPデータグラムの受信ポート番号 (ros2\_rx\_udp\_portで指定) はこれらのポート番号と重複してはならない。

- SPDP\_WELL\_KNOWN\_MULTICAST\_PORT
  - $7400 + \text{DOMAIN\_ID} * 250$
- METATRAFFIC\_UNICAST\_LOCATOR
  - $7400 + \text{DOMAIN\_ID} * 250 + 10 + 2 * \text{PARTICIPANT\_ID}$
- DEFAULT\_UNICAST\_LOCATOR
  - $7400 + \text{DOMAIN\_ID} * 250 + 11 + 2 * \text{PARTICIPANT\_ID}$

式中のDOMAIN\_IDは、下記の値である。

(ros2\_port\_num\_seedポートの値) - 7400 / 250