Oriental motor

ROS パッケージ

Modbus RTU Node

取扱説明書

第 2 版

目次

1 はじめに

■Modbus RTU Node について

Modbus RTU Node は Modbus RTU に対応している製品を制御するノードです。ROS のメッセージを配信するだけで Modbus RTU 制御を実現できます。

■対象製品

ステッピングモーター：AZ シリーズ(位置決め機能内蔵タイプ、RS-485 通信付きパルス列入力タイプ)
AR シリーズ(位置決め機能内蔵タイプ)
RKⅡシリーズ(位置決め機能内蔵タイプ)
CVD シリーズ(RS-485 通信タイプ)
ブラシレスモーター　：BLH シリーズ(RS-485 通信タイプ)
BLE シリーズ(RS-485 通信タイプ)
BLV シリーズ

■動作確認環境

・OS：Ubuntu16.04(Linux)
・ROS ディストリビューション：Kinetic Kame
・ROS バージョン ： 1.12.14

■対象製品の使い方について

対象製品の詳細については各シリーズのユーザーズマニュアルでご確認ください。

■RS-485 通信コネクタへの接続について

通常、パソコンには RS-485 通信の端子はついておりません。そのため、パソコンから RS-485 通信を行うためには、RS-485 通信機器（RS-485 通信ボード、USB-RS-485 通信変換ケーブル等）が必要となります。当社では RS-485 通信機器は扱っておりませんので、お客様にてご用意いただけますようお願いします。RS-485 通信機器と当社ドライバ間の接続については、製品のユーザーズマニュアルにてご確認ください。

■RS-485 通信パラメータ

RS-485 の通信パラメータは通信 ID、Baudrate のみ変更可能です。その他の通信パラメータはすべて以下の内容で使用します。

RS-485 通信パラメータ	設定範囲
通信 ID(Modbus)	1～31　※BLH は 1～15
Baudrate(Modbus)	9600、19200、38400、57600、115200、230400bps ※230400bps は AZ、CVD、BLH のみ
通信順序(Modbus) AZ、CVD、BLH のみ	初期値(0:EvenAddress-HighWord & Big-Endian)
通信パリティ	初期値(1:偶数パリティ)
通信ストップビット	初期値(0:1 ビット)
通信タイムアウト(Modbus)[ms]	初期値(0:監視しない)
通信異常アラーム(Modbus)	初期値(3)
送信待ち時間(Modbus)[ms]	初期値(3) AZ、CVD、BLH 初期値(10) AR、RKⅡ、BLE、BLV
サイレントインターバル(Modbus)[ms] AZ、CVD、BLH のみ	初期値(0:自動で設定する)
スレーブエラー検出時応答(Modbus) AZ、CVD、BLH のみ	初期値(1:例外応答を返信する)
グループ ID 初期値(Modbus) AZ、CVD、BLH のみ	初期値(-1:無効)

2 注意事項

(1) 実際のシステム構築に際しては、システムを構成する各機器・装置の仕様をご確認のうえ、定格・性能に対し余裕を持った使い方をし、万一故障があっても危険を最小にする安全回路などの安全対策を講じてください。

(2) システムを安全にご使用いただくため、システムを構成する各機器・装置のマニュアルや取扱説明書などを入手し、「安全上のご注意」「安全上の要点」など安全に関する注意事項を含め、内容を確認の上使用してください。

(3) システムが適合すべき規格・法規または規制に関しては、お客様自身でご確認ください。

(4) 本資料の一部または全部を、オリエンタルモーター株式会社の許可なしに複写、複製、再配布することを禁じます。

(5) 本資料の記載内容は、2020 年 10 月時点のものです。本資料の記載内容は、改良のため予告なく変更されることがあります。

(6) 本資料は機器の通信接続確立までの手順について記載したものであって、機器個別の操作や設置および配線方法に関しては記載しておりません。通信接続手順以外の詳細に関しては、対象製品の取扱説明書を参照してください。

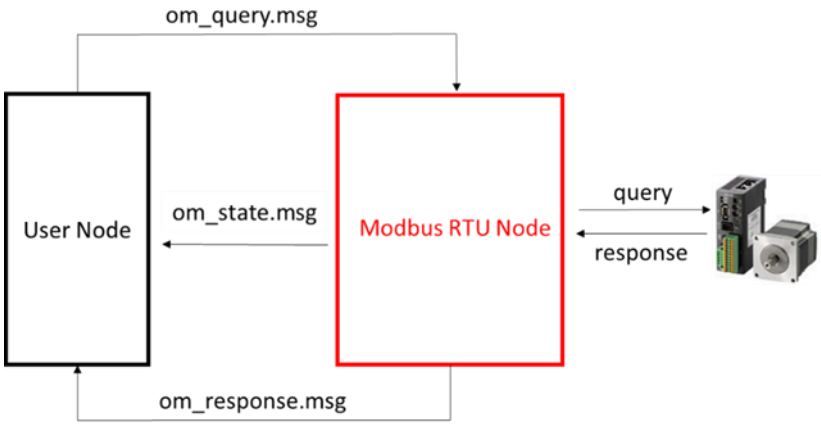
(7) 本資料は ROS や Linux の専門知識を持った方を対象としています。ROS のインストール・使い方、Linux についてのお問い合わせはサポート対象外となりますので、あらかじめご了承ください。

3 準備

3.1 概要

Modbus RTU Node は Modbus RTU 通信から標準化されている ROS メッセージへのラッパーを提供します。
以下の流れで処理が行われます。

1) ユーザーノードが Modbus RTU Node にクエリデータを配信
2) Modbus RTU Node が配信されたデータを購読
3) ドライバが通信可能であれば、ドライバにクエリを送信
4) ドライバからのレスポンスを解析、ユーザーノードに配信
5) エラーが発生した場合はステータスを更新。ステータスはユーザーノードに定期周期で配信



3.2 Modbus RTU Node の導入方法

任意の場所にユーザー作業フォルダを作成し、一度ビルドを行います。次に Modbus RTU Node パッケージを解凍します。解凍後、ユーザー作業フォルダの「src」フォルダに(/home/catkin_ws/src)に解凍したフォルダをコピー、ペーストします。その後、catkin_make のコマンドでビルドを行い、ビルドに成功すると導入は完了です。

① ユーザー作業フォルダの作成

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ~/catkin_ws
$ catkin_make
```

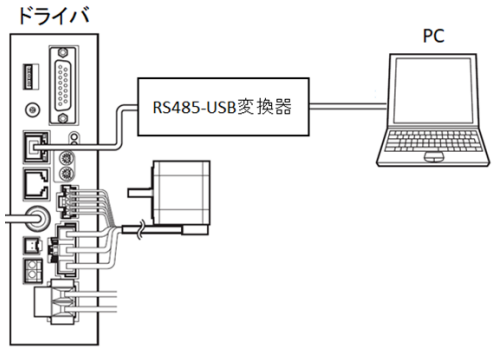
② ビルド

```
$ cd ~/catkin_ws
$ catkin_make
```

※ユーザー作業フォルダは任意の場所に作成できますが、ここでは以下のフォルダに作成しています。
ユーザー作業フォルダ：/home/catkin_ws/
※ビルド時に自動生成されるファイルが生成されず、ビルドに失敗するときがあります。繰り返しビルドを実行すると生成されビルドに成功します。

3.3　PC とドライバの接続

PC とドライバの接続方法を以下に示します。ドライバの RS-485 通信コネクタに市販の LAN ケーブルを接続し、RS485-USB 変換器で USB に変換して PC に接続してください。RS485-USB 変換器はお客様にてご用意ください。



3.4　ノード起動方法

最初にターミナルから「roscore」コマンドを入力しマスターを立ち上げます。その後、別のターミナルから Modbus RTU Node とユーザーノードを起動します。ユーザーノードが配信を行うと通信が開始されます。Modbus RTU Node は ROS の機能である launch コマンドで起動します。引数を省略した場合はデフォルト値が設定されます。

引数名	型	内容
com	文字列型	シリアルポートに対応するデバイスファイル名 デフォルト値(”/dev/ttyUSB0”)
topicID	数値型	Modbus RTU Node を複数起動時にノードとメッセージを識別するための ID(0～15)。デフォルト値(0)
baudrate	数値型	ボーレート[bps] 9600、19200、38400、57600、115200、230400 デフォルト値(9600)
updateRate	数値型	ユーザーノードに配信する周期[Hz](ステータスのみ) 0～1000 デフォルト値(1) ※0 のときはステータスの定期配信を実行しません。 ※環境により配信できる最大周期は変わります。
firstGen	文字列型	第 1 世代のスレーブ ID(1～31)を指定 “1,2,3,…31”　デフォルト値(空文字)
secondGen	文字列型	第 2 世代のスレーブ ID(1～31)を指定 “1,2,3,…31”　デフォルト値(空文字)

(例) BLV(スレーブ ID=1,2)と AZ(スレーブ ID=3,4)をそれぞれ 2 台使用する場合

```
$ roslaunch om_modbus_master om_modbusRTU.launch com:="/dev/ttyUSB0" topicID:=1 baudrate:=115200 updateRate:=1000 firstGen:="1,2," secondGen:="3,4,"
```

※第 1 世代は AR、RKⅡ、BLE、BLV　第 2 世代は AZ、CVD、BLH となっています。

※指定できるスレーブ ID は第 1 世代、第 2 世代を合わせて 8 個までです。

※com は環境によって名前が変わるのでターミナルで ls /dev/tty*と入力してシリアルデバイス名の確認を行い、表示されたデバイス名を入力してください。

※Modbus RTU Node のノード名は om_modbusRTU□になります。□には topicID の数値が入ります。

※USB ポートを使用する場合は以下のコマンドで USB ポートの読み書きの権限を与える必要があります。

```
$ sudo chmod 666 /dev/ttyUSB0
```

※同一のシリアルポートを指定して複数ノードを起動した場合はユーザーノードからメッセージを配信時にノードが強制終了します。

4 メッセージ

Modbus RTU Node で使用するメッセージについて説明します。各メッセージの初期値はすべて 0 です。

4.1 クエリデータの配信

ユーザーノードからは以下のメッセージを配信します。Modbus RTU Node が配信されたメッセージを受け取り、クエリに変換しドライバに送信します。配信する際はトピック名とメッセージ名を宣言する必要があります。トピック名は”om_query□”という名前で□には Modbus RTU Node 起動時に指定した topicID が入ります。メッセージ名は”om_query”で固定です。

メッセージ名	型	内容
slave_id	int8	スレーブ ID(0～31)
func_code	int8	ファンクションコード(0:read、1:write、2:read&write)
write_addr	int32	書き込みの起点となる上位のレジスタアドレス
read_addr	int32	読み込みの起点となる上位のレジスタアドレス
write_num	int8	書き込みのデータ数(1～32 個)
read_num	int8	読み出しのデータ数(1～32 個)
data[32]	int32	レジスタへの書き込みデータ(write、read&write の時のみ指定)

4.2 レスポンス購読

read、read&write のときはレスポンスを解析しユーザーノードに配信、write のときはドライバのレスポンスをそのまま配信します。レスポンスデータを購読する際はトピック名とメッセージ名を宣言する必要があります。トピック名は”om_response□”という名前で□には Modbus RTU Node 起動時に指定した topicID が入ります。メッセージ名は”om_response”で固定です。

メッセージ名	型	内容
slave_id	int8	購読したレスポンスのスレーブ ID
func_code	int8	購読したレスポンスのファンクションコード
data[32]	int32	ドライバからのレスポンスを解析した結果

4.3 ステータス購読

ステータス情報は定期周期で Modbus RTU Node から配信されます。周期は launch コマンドの updateRate 引数で指定します。ステータスを購読する際はトピック名とメッセージ名を宣言する必要があります。トピック名は”om_state□”という名前で□には Modbus RTU Node 起動時に指定した topicID が入ります。メッセージ名は”om_state”で固定です。

メッセージ名	型	内容
state_driver	int8	0:通信可能　1:通信中
state_mes	int8	0:メッセージ無し　1:メッセージ到達　2:メッセージエラー
state_error	int8	0:エラー無し　1:無応答　2:例外応答

4.4 ノードの通信間隔

ユーザーノードから連続で配信する際はステータスの state_driver が 1 から 0 になるまで時間を空けてください。ステータスを使用しない場合は間隔を 50[ms]以上(第 1 世代の read&write を使用するときは 100[ms]以上)長くしてください。state_driver が 1(通信中)のときにメッセージを配信した場合、そのメッセージは破棄され、エラーなどは発生しません。

5 パッケージ構成

パッケージの構成は以下の通り。

```
om_modbus_master
 |
 ├── include
 |     └── om_modbus_master
 |           ├── ICheckData.h
 |           ├── IConvertQueryAndResponse.h
 |           ├── ISetMessage.h
 |           ├── ISetResponse.h
 |           ├── om_base.h
 |           ├── om_broadcast.h
 |           ├── om_first_gen.h
 |           ├── om_node.h
 |           ├── om_ros_message.h
 |           └── om_second_gen.h
 |
 ├── launch
 |     └── om_modbusRTU.launch
 |
 ├── msg
 |     ├── om_query.msg
 |     ├── om_response.msg
 |     └── om_state.msg
 |
 ├── src
 |     ├── om_base.cpp
 |     ├── om_broadcast.cpp
 |     ├── om_first_gen.cpp
 |     ├── om_node.cpp
 |     ├── om_ros_message.cpp
 |     └── om_second_gen.cpp
 |
 ├── sample
 |     ├── cpp
 |     |     ├── AZ
 |     |     |    ├── sample1_1.cpp
 |     |     |    ├── sample1_2.cpp
 |     |     |    ├── sample2_1.cpp
 |     |     |    ├── sample2_2.cpp
 |     |     |    └── sample3.cpp
 |     |     |
 |     |     └── BLV
 |     |          ├── sample1_1.cpp
 |     |          ├── sample1_2.cpp
 |     |          ├── sample2.cpp
 |     |          └── sample3.cpp
 |     |
 |     └── python
 |          ├── AZ
 |          |    ├── sample1_1.py
 |          |    ├── sample1_2.py
 |          |    ├── sample2_1.py
 |          |    ├── sample2_2.py
 |          |    └── sample3.py
```

```
|           |
|           └── BLV
|                ├── sample1_1.py
|                ├── sample1_2.py
|                ├── sample2.py
|                └── sample3.py
|
├── CMakeLists.txt
└── package.xml
```

各ファイルの内容を以下に示します。

名前	概要
om_modbusRTU.launch	roslaunch で起動するノードやパラメータを指定
om_query.msg	クエリデータの定義
om_response.msg	レスポンスデータの定義
om_state.msg	ステータスデータの定義
om_ros_message.cpp	ROS 通信クラスのソースファイル
om_ros_message.h	ROS 通信クラスのヘッダーファイル
om_base.cpp	シリアル通信クラスのソースファイル
om_base.h	シリアル通信クラスのヘッダーファイル
om_first_gen.cpp	Modbus RTU 基底クラス(第 1 世代)のソースファイル
om_firts_gen.h	Modbus RTU 基底クラス(第 1 世代)のヘッダーファイル
om_second_gen.cpp	Modbus RTU 派生クラス(第 2 世代)のソースファイル
om_second_gen.h	Modbus RTU 派生クラス(第 2 世代)のヘッダーファイル
om_broadcast.cpp	Modbus RTU 派生クラス(ブロードキャスト)のソースファイル
om_broadcast.h	Modbus RTU 派生クラス(ブロードキャスト)のヘッダーファイル
om_node.cpp	メイン関数のソースファイル
om_node.h	メイン関数のヘッダーファイル
ICheckData.h	インターフェイス
IConvertQueryAndResponse.h	インターフェイス
ISetMessage.h	インターフェイス
ISetResponse.h	インターフェイス
CMakeLists.txt	ビルド設定
package.xml	パッケージの名前、バージョン、作者などの情報
sample1_1(.cpp、.py)	書き込みのサンプル(AZ、BLV)
sample1_2(.cpp、.py)	読み込みのサンプル(AZ、BLV)
sample2(.cpp、.py)	モーターの動作のサンプル(BLV) 3 ワイヤによる動作。出荷時には運転データ No.0, No.1 は VR1、外部設定器により回転速度を指定するので運転データ No.2 に書き込み
sample2_1(.cpp、.py)	モーターの動作(ストアードデータ運転)のサンプル(AZ)
sample2_2(.cpp、.py)	モーターの動作(ダイレクトデータ運転)のサンプル(AZ)
sample3(.cpp、.py)	モーターの動作(2 軸)&検出位置モニタのサンプル(AZ) モーターの動作(2 軸)&検出速度モニタのサンプル(BLV)

6 ビルド設定

ROSはcatkinという独自のビルドシステムを使用します。これはCMakeを拡張したものでありCMakeでは"CMakeLists.txt"にビルドするために必要な情報を記載します。以下に"CMakeLists.txt"の設定を示します。

```
## CMakeバージョン
cmake_minimum_required(VERSION 2.8.3)

## パッケージ名
project(om_modbus_master)

## C++11を使用
add_compile_options(-std=c++11)

## 依存パッケージを指定
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  message_generation
)

## Generate messages in the 'msg' folder
add_message_files(
   FILES
   om_query.msg
   om_response.msg
   om_state.msg
)

## Generate added messages and services with any dependencies listed here
generate_messages(
  DEPENDENCIES
  std_msgs
)

## CMakeファイルを生成するために必要なcatkin固有の情報をビルドシステムに伝える
catkin_package(
 INCLUDE_DIRS include
CATKIN_DEPENDS roscpp rospy std_msgs message_runtime
)

## includeディレクトリ
include_directories(include ${catkin_INCLUDE_DIRS})

## ビルドするべき実行ファイルのターゲット
add_executable(om_modbusRTU       src/om_ros_message.cpp       src/om_base.cpp       src/om_first_gen.cpp
src/om_second_gen.cpp  src/om_node.cpp src/om_broadcast.cpp)
add_dependencies(om_modbusRTU_node ${${PROJECT_NAME}_EXPORTED_TARGETS}${catkin_EXPORTED_TARGETS})

## 実行可能なターゲットがどのライブラリにリンクするかを指定
target_link_libraries(om_modbusRTU ${catkin_LIBRARIES})
```

7 サンプルコード

「sample」フォルダ内にはユーザーノードのサンプル(C++, Python)が含まれています。Python のサンプルコードは以下をターミナルに入力すると実行できます。(サンプルコードを実行する前に Modbus RTU Node を起動しておいてください)
サンプルの内容についてはサンプルのソースコード内のコメントを参照してください。

```
$ cd ~/catkin_ws/src/om_modbus_master/sample/python/BLV
$ python sample1_1.py
```

※ユーザー作業フォルダ : /home/catkin_ws/

Modbus RTU Node は以下の launch コマンドで起動しています。(BLV のサンプルの場合)

```
$ roslaunch om_modbus_master om_modbusRTU.launch com:="/dev/ttyUSB0" topicID:=1 baudrate:=115200 updateRate:=1000 firstGen:="1,2,"
```

オリエンタルモーターおよびオリエンタルモーターに対する実施許諾者は、本ソフトウェアの使用に付随または関連して生ずる直接的または間接的な損失、損害等（ハードウェア又は他のソフトウェアの破損、事業利益の喪失、事業の中断、事業情報の喪失などにかかる損害を含みますが、これらに限定されません）について、いかなる場合においても一切の責任を負いません。

オリエンタルモーター株式会社
作成：2020 年 10 月 28 日