Renesas Motor Workbench

通信用 DLL 機能説明書

要旨

本資料は、Renesas Motor Workbench(RMW)の機能のうち、マイコンとの通信機能を抜粋した通信用 DLL の使用方法を説明しています。正しくお使いいただくために、本資料をよくお読みください。

目次

1. DLL 概要

1.1 Renesas Motor Workbench について

モータ制御開発支援ツールである Renesas Motor Workbench(以下「RMW」と表記) は、マイコン上で実行するモータ制御用ソフトウェアのグローバル変数の値に関して、パソコン上から値の読み込み及び書き込みやリアルタイムでの波形表示、モータの固有パラメータ測定や制御パラメータの同定結果を表示及び出力するツールです。

RMW の主な機能を以下に示します。

・実行プログラムの変数リスト情報の読み込み
・RMW で操作及び設定した環境情報の RMT ファイルへの保存
・RMT ファイルの読み込み及び各機能への反映
・USB 接続したマイコンとの接続及び切断
・マイコンからの値の読み込み
・マイコンへの値の書き込み
・マイコンから連続で値の読み込み
・モータ固有パラメータ測定
・制御パラメータ調整

1.2 DLL の概要

通信用 DLL「RMWCommunicationLibrary.dll」(以下、「DLL」と表記)は、RMW の機能のうち、マイコンとの接続及び切断、マイコンからの値の読み込み、マイコンへの値の書き込み、マイコンから連続で値の読み込みを行う機能(Scope 機能)、実行プログラムの変数リスト情報の読み込み機能(Map ファイル変換機能)を抜粋し、ユーザが作成したソフトウェア(C#／Excel VBA)から本 DLL の関数を実行することで、RMW の機能を利用可能にした DLL です。

1.3 システム構成

本 DLL のシステム構成を以下に示します。

ユーザが作成したソフトウェア(C#／Excel VBA)から本 DLL の関数を実行することで、ツール用通信ボードを経由してマイコンと通信し、値の読み込み及び書き込みやモータの制御等を行います。

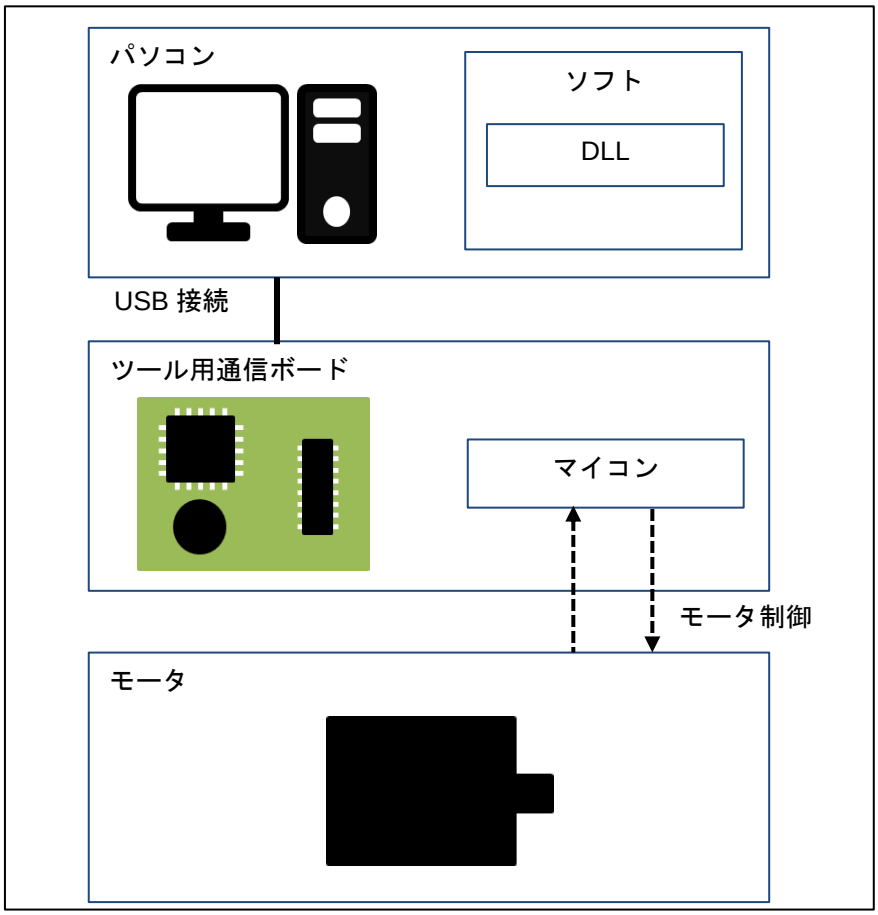


図 1-1 システム構成

1.4 動作環境

本 DLL の動作環境を以下に記載します。

表 1-1 動作環境

#	項目名	値
1	OS	Windows 10 のみ
2	.Net Framework	.Net Framework 4.6.1 以上
3	通信手段	シリアルポート接続
4	USB ドライバ	Windows10 用標準 USB ドライバ
5	開発環境	Visual Studio 2015 以降、Excel
6	対応言語	C#、Excel VBA
7	文字コード	UTF-16LE

※.Net Framework の更新が必要な場合、以下の Web ページで最新のインストーラを取得し、.NET Framework を更新してください。
https://dotnet.microsoft.com/download/dotnet-framework/net48

2. 各機能の概要

2.1 接続機能

本機能は、マイコンが接続している COM ポートとボーレートを指定してマイコンとの接続を行います。

2.2 切断機能

本機能は、接続機能で接続したマイコンとの切断を行います。

2.3 読み込み機能

本機能は、マイコンの読み込む変数のアドレスと変数の型を指定し実行することで、指定したアドレスを先頭アドレスとして、変数の型に応じたサイズ分、マイコンから値を読み込みます。

2.4 書き込み機能

本機能は、マイコンの書き込む変数のアドレスと書き込む値を指定し実行することで、マイコンの指定したアドレスに指定した値を書き込みます。

2.5 Scope 機能

本機能は、チャネル設定を含む Scope の設定及びトリガ情報を指定し実行することで、設定したチャネルの変数の値を連続的にマイコンから読み込みます。

2.6 Map ファイル変換機能

本機能は、指定された Map ファイルを変換し、メモリ上に展開、もしくは CSV ファイルに出力します。

メモリ上に展開する処理及び CSV ファイルを出力する処理が実行された時に共通で、Map ファイルの読み込み及び CSV 形式(カンマ区切り)への変換処理を実行します。

CSV 形式への変換処理が完了後、実行された関数に応じてメモリ上に展開するか、CSV 形式のファイルに出力します。

3. DLL の導入・削除

3.1 導入方法

3.1.1 Visual Studio への導入方法

DLL を Visual Studio に導入する方法を説明します。

本 DLL は、「.Net Framework」対応の DLL です。Visual Studio で新しいプロジェクトを作成する際には、プロジェクトの種類を「.Net Core」ではなく、「.Net Framework」を選択してください。

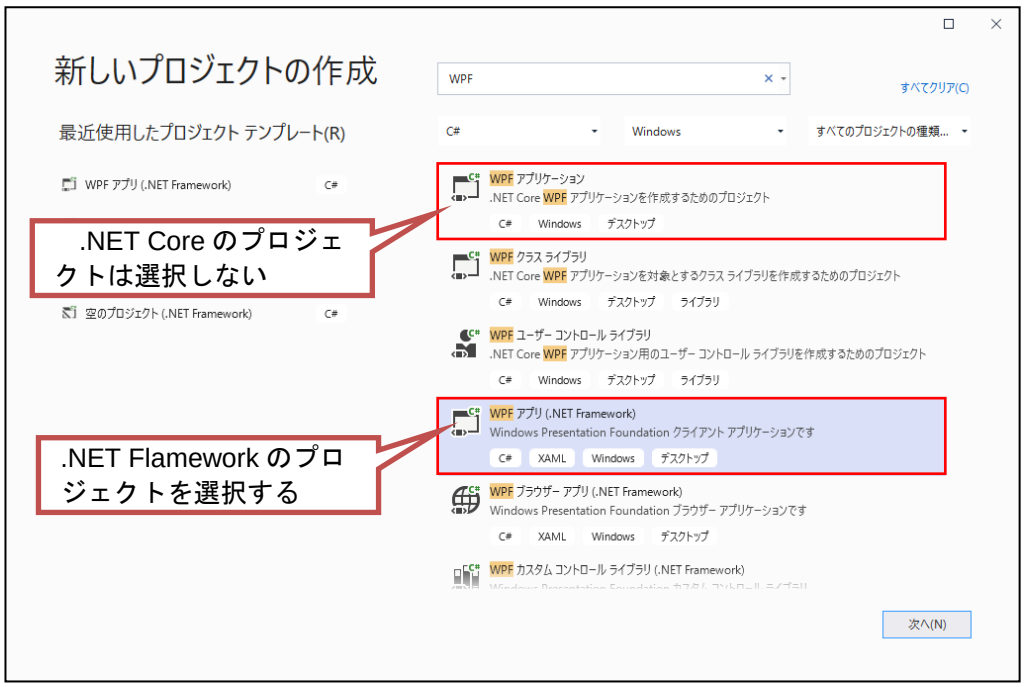


図 3-1 プロジェクトの種類

DLL を更新する場合は、「3.1.1(1)プロジェクトに DLL を追加」でプロジェクトに追加した DLL を、上書きしてください。

(1) プロジェクトに DLL を追加

画面上部から「表示」-「ソリューションエクスプローラー」を選択します。

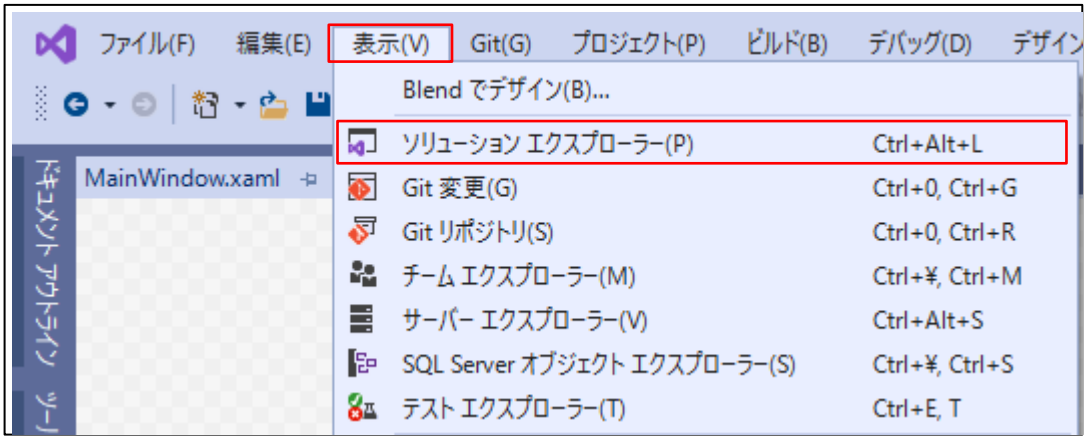


図 3-2 ソリューションエクスプローラーの表示

ソリューションエクスプローラーから、DLL を追加する場所(プロジェクトもしくは新しく DLL 格納用のフォルダを追加する)で右クリックし、「追加」-「既存の項目」を選択します。

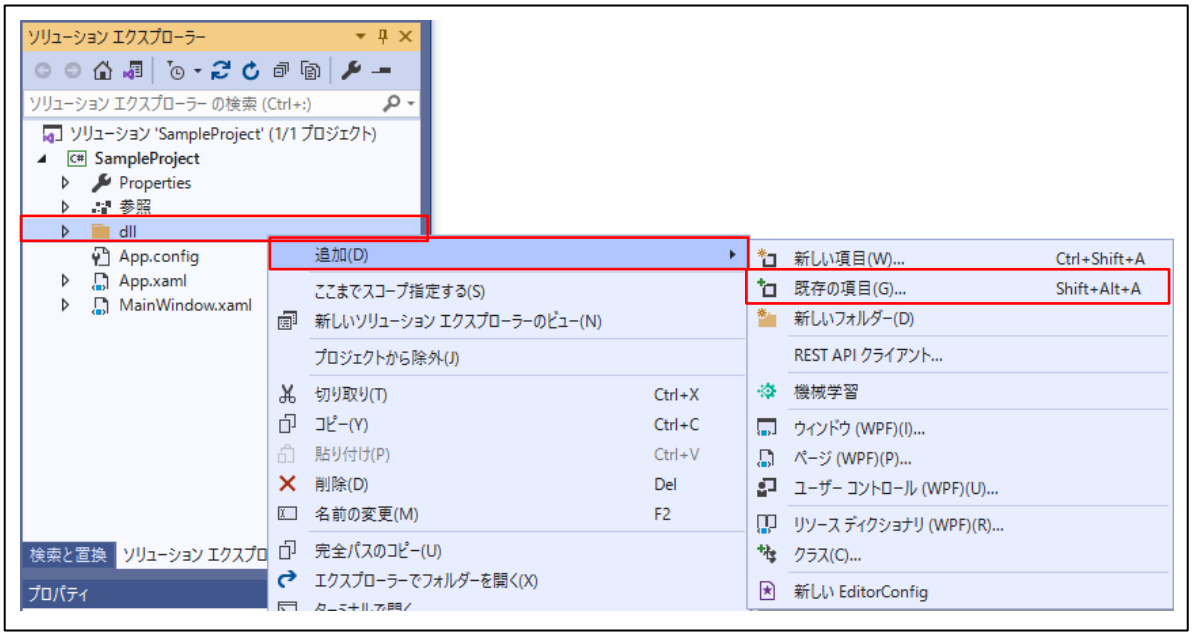


図 3-3 ソリューションエクスプローラーでの選択

既存項目の追加画面から「RMWCommunicationLibrary.dll」を選択し、「追加」ボタンを押下します。

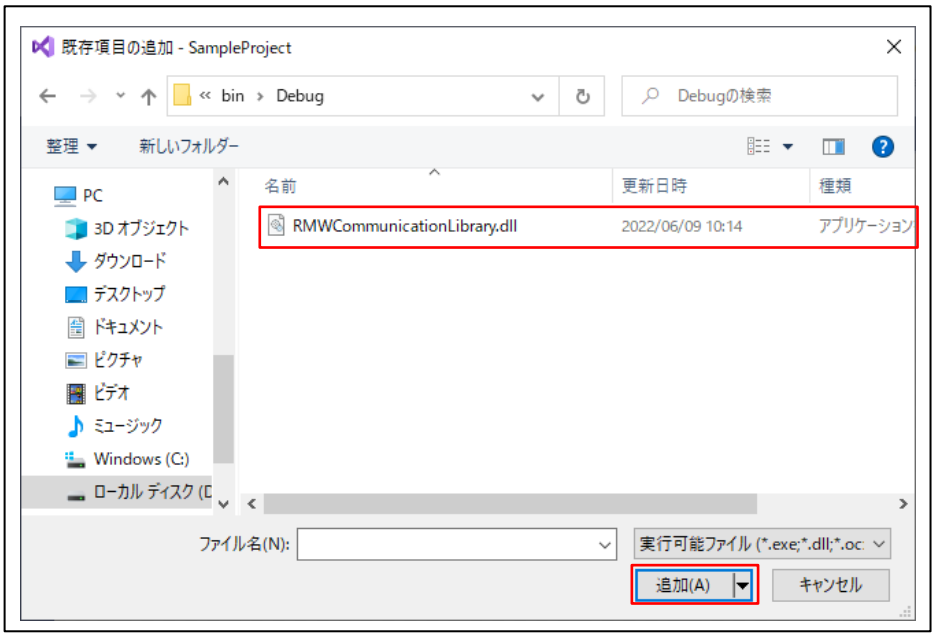


図 3-4 既存項目の追加画面での選択

ソリューションエクスプローラーに以下のように表示されていれば、プロジェクトへの DLL ファイルの登録は完了です。

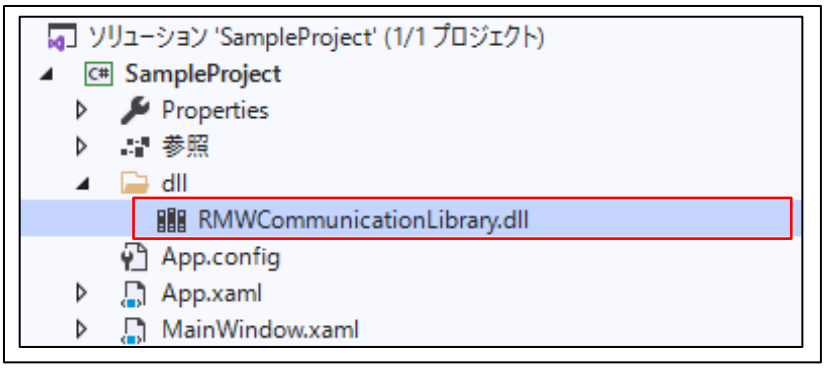


図 3-5 登録完了時の表示

(2) プロジェクトの参照設定に DLL を追加

画面上部から「プロジェクト」-「参照の追加」を選択します。

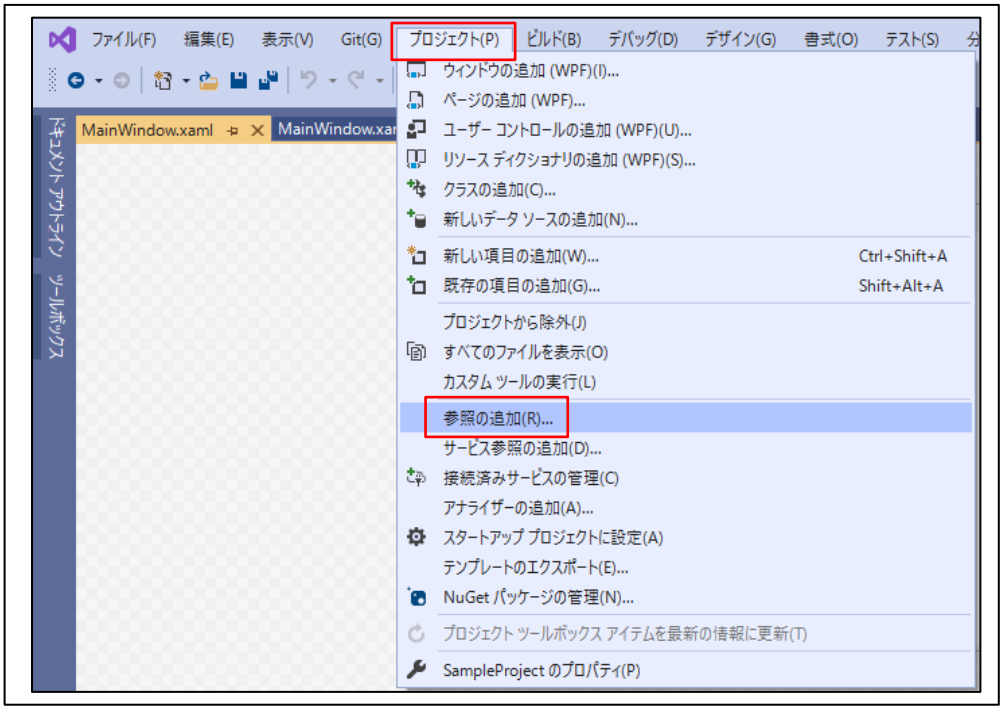


図 3-6 参照マネージャーの表示

参照マネージャーから「参照」ボタンを押下します。

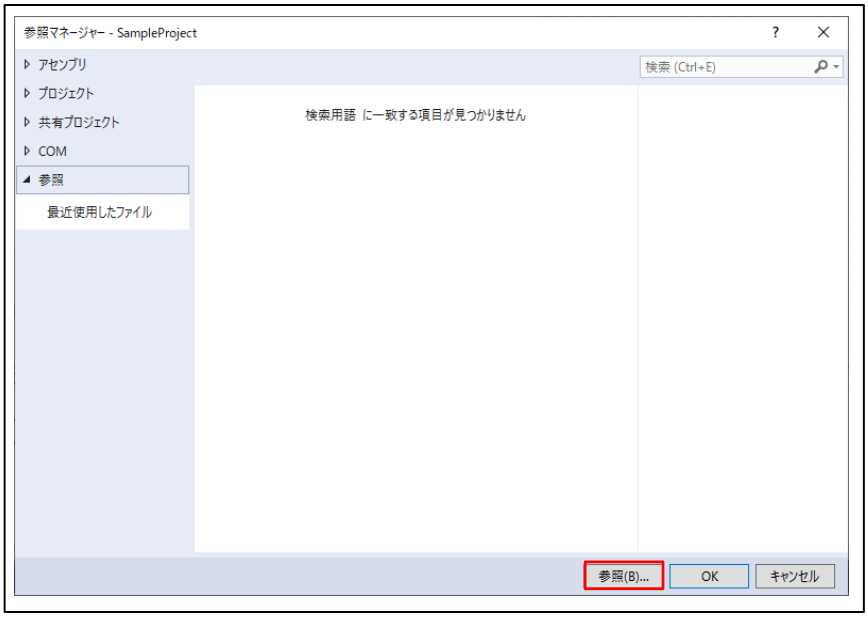


図 3-7 参照マネージャーでの選択

プロジェクトに追加した「RMWCommunicationLibrary.dll」を選択し、「追加」ボタンを押下します。

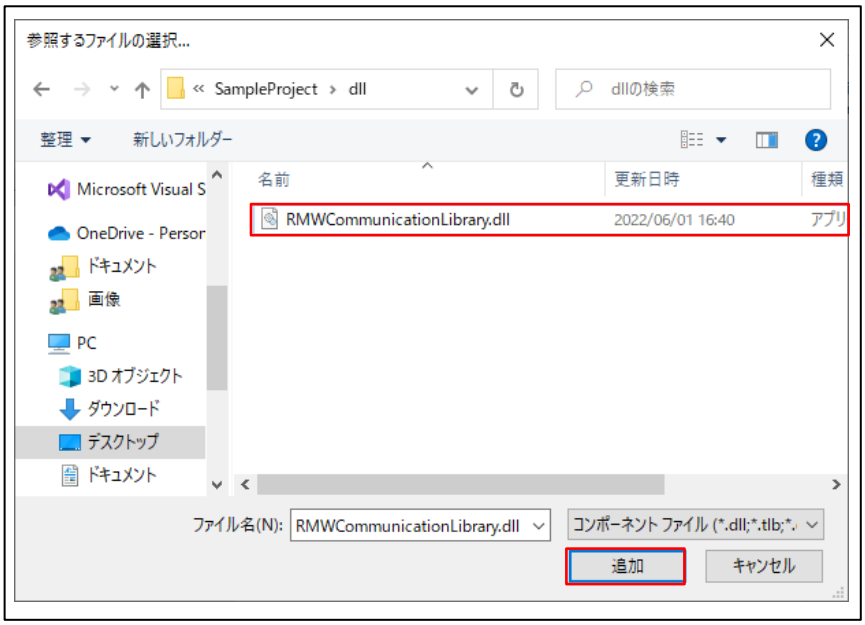


図 3-8 参照ファイルの選択

参照マネージャーで「OK」ボタンを押下します。

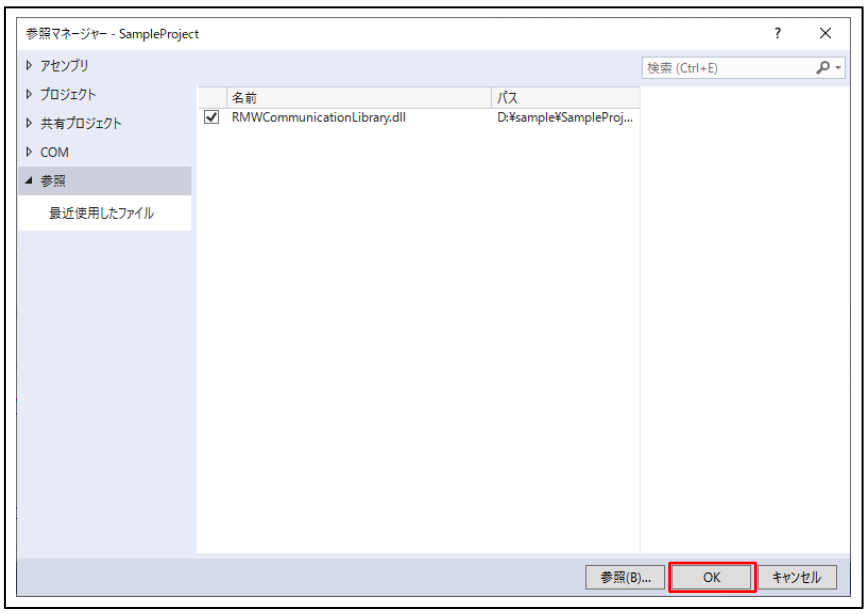


図 3-9 参照の登録

(3) プロジェクトの保存

画面上部から「ファイル」-「すべて保存」を選択します。

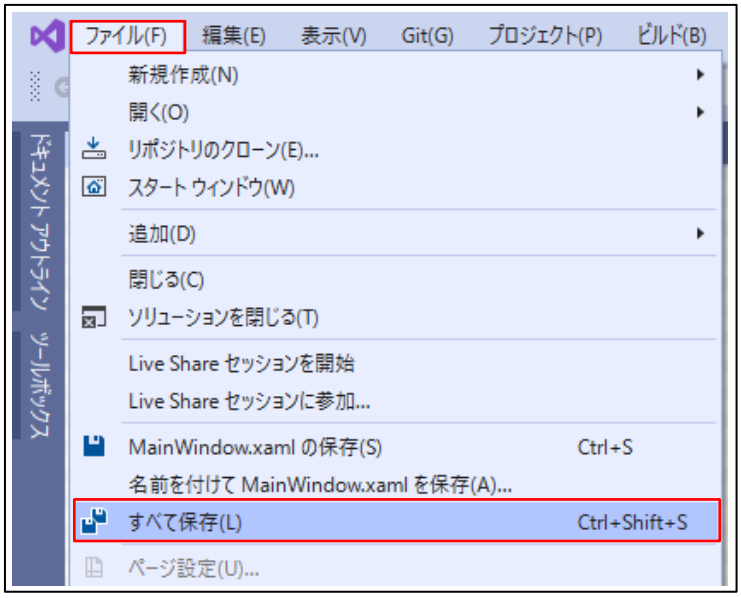


図 3-10 プロジェクトの保存

3.1.2 Excel への導入方法

DLL を Excel に導入する方法を説明します。

[重要！]DLL を更新する場合、開いている全ての Excel を閉じた後に、必ず「3.2.2Excel からの削除方法」の手順で DLL のファイルを削除してから、「1.1.1(1)DLL を任意のディレクトリに展開」を実行してください。

(1) DLL を任意のディレクトリに展開

DLL を任意のディレクトリに展開します。

図 3-11 DLL の展開

(2) レジストリに DLL を登録

スタートメニューから「Windows システム ツール」を選択します。

「コマンドプロンプト」を右クリックし、「その他」-「管理者として実行」を選択します。

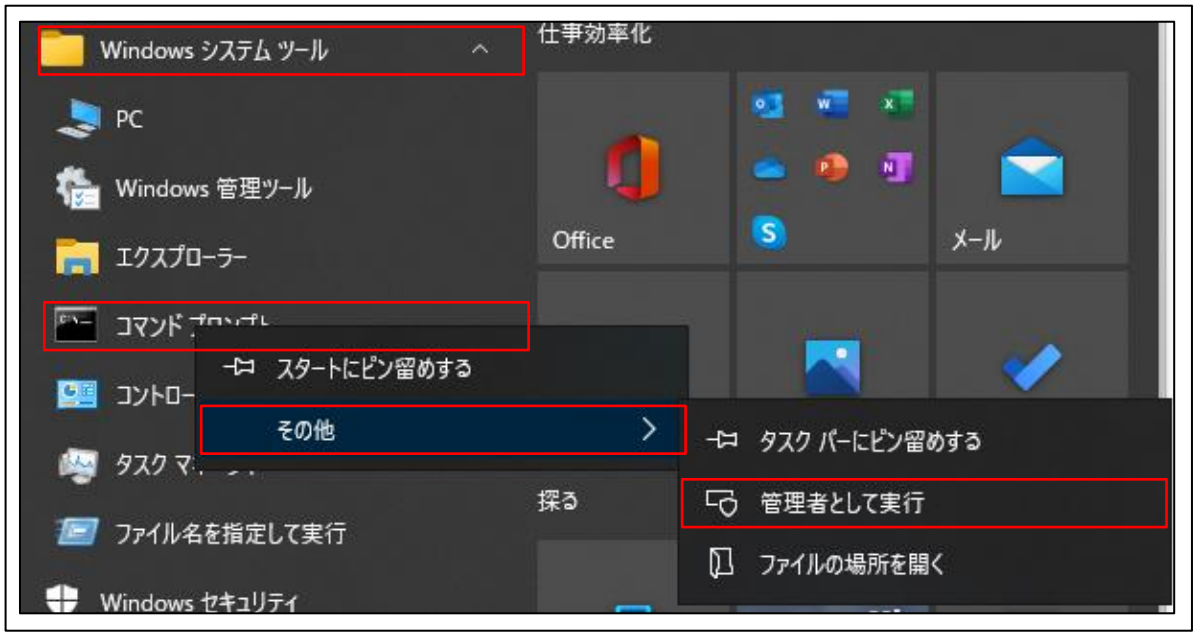


図 3-12 コマンドプロンプトの起動

コマンドプロンプトで、以下のコマンドを実行します。

【実行するコマンド】

(RegAsm.exe 格納パス)¥RegAsm.exe /tlb /codebase "(DLL 展開先パス)¥RMWCommunicationLibrary.dll"

RegAms.exe 格納パスは以下の通り。

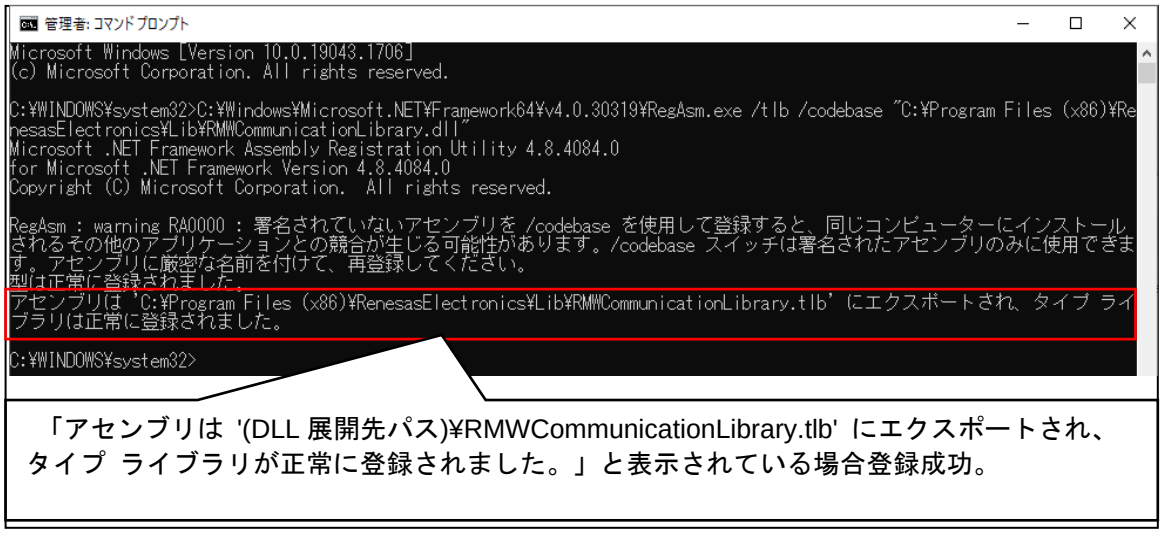
64bit Excel：C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319

32bit Excel：C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319

実行する RegAsm.exe は OS の種類ではなく、Excel のバージョンで変更します。

※Excel のバージョンについては、以下の Web ページを参照してください。

https://support.microsoft.com/ja-jp/office/%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%81%84%E3%82%8B-office-%E3%81%AE%E3%83%90%E3%83%BC%E3%82%B8%E3%83%A7%E3%83%B3%E3%82%92%E7%A2%BA%E8%AA%8D%E3%81%99%E3%82%8B%E6%96%B9%E6%B3%95-932788b8-a3ce-44bf-bb09-e334518b8b19



「アセンブリは '(DLL 展開先パス)¥RMWCommunicationLibrary.tlb' にエクスポートされ、タイプ ライブラリが正常に登録されました。」と表示されている場合登録成功。

図 3-13 コマンドプロンプトの実行画面

(3) Excel の参照設定に DLL を追加

Excel のリボンに「開発」タブが表示されていない場合、以下の Web ページを参照し、「開発」タブを表示してください。

https://support.microsoft.com/ja-jp/topic/-%E9%96%8B%E7%99%BA-%E3%82%BF%E3%83%96%E3%82%92%E8%A1%A8%E7%A4%BA%E3%81%99%E3%82%8B-e1192344-5e56-4d45-931b-e5fd9bea2d45

Excel のリボンに表示されている「開発」-「Visual Basic」を選択します。

図 3-14 Visual Basic 画面の表示

Visual Basic 画面の「ツール」-「参照設定」を選択します。

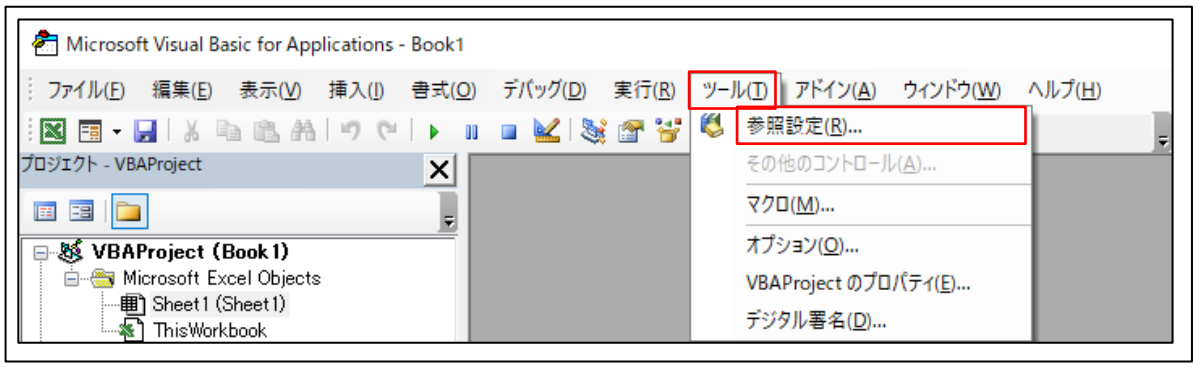


図 3-15 参照設定画面の表示

参照設定画面で「RMW Communication Library」をチェックし、「OK」ボタンを押下します。

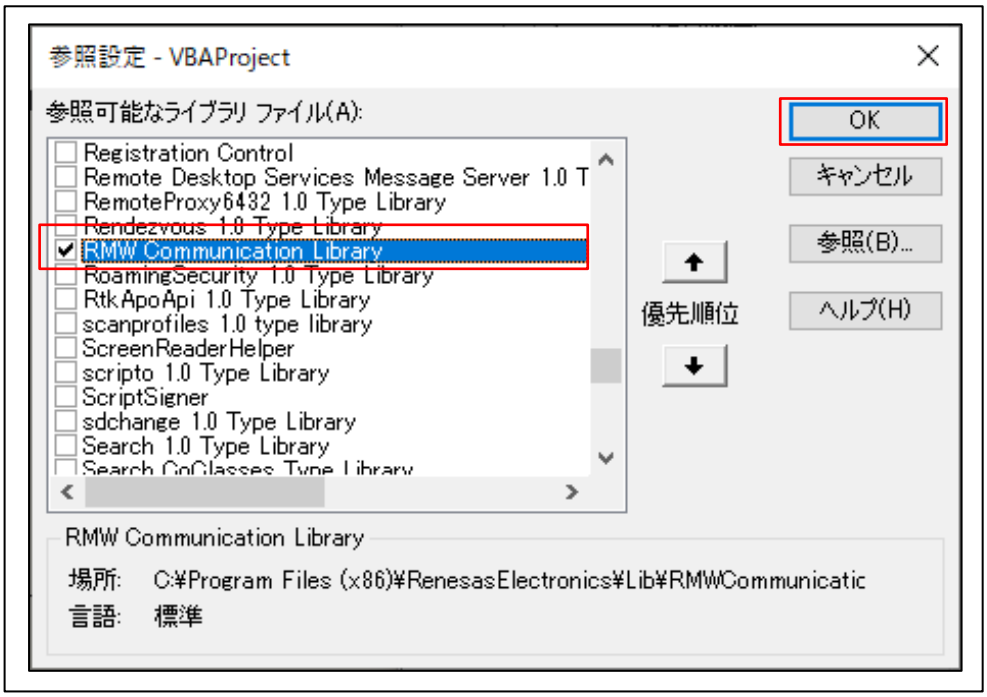


図 3-16 参照の登録

※上記画面に「RMW Communication Library」が表示されていない場合、DLL が PC に登録できていないため、「(2)レジストリに DLL を登録」を再度実行してください。

(4) ファイルの保存

Excel のリボンに表示されている「ファイル」を選択します。

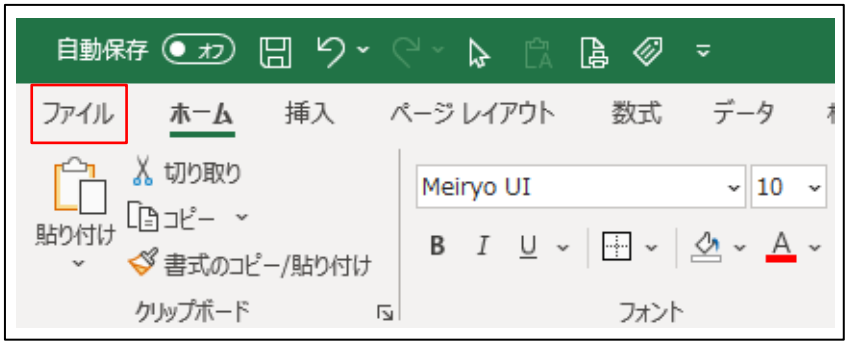


図 3-17 ファイルタブの選択

「上書き保存」または「名前を付けて保存」を選択します。

図 3-18 ファイルの保存

3.2 削除方法

3.2.1 Visual Studio からの削除方法

DLL を Visual Studio から削除する方法を説明します。

(1) プロジェクトの参照設定から DLL を解除

画面上部から「プロジェクト」-「参照の追加」を選択します。

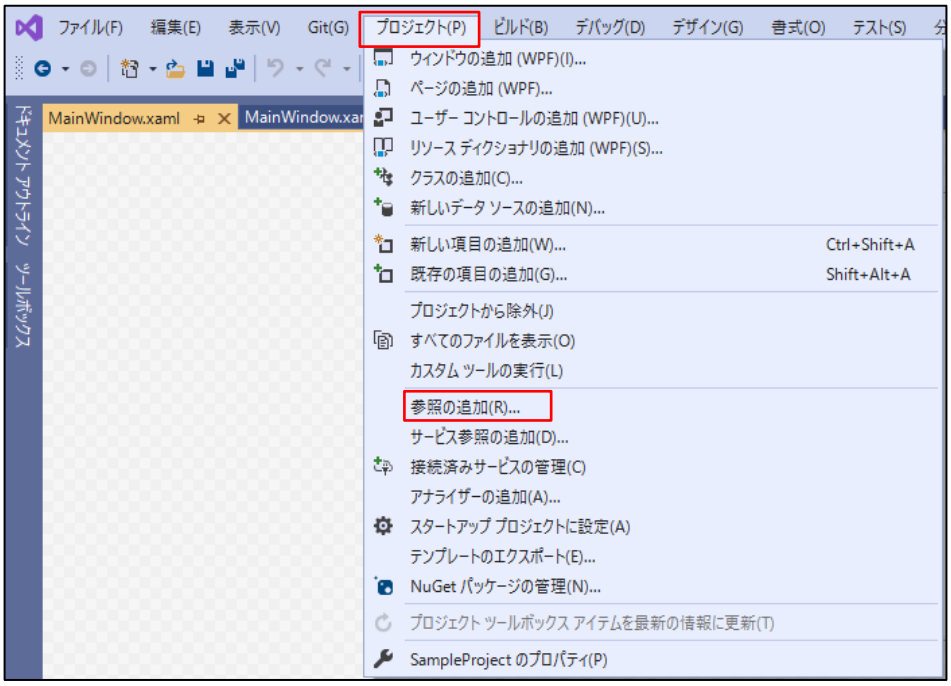


図 3-19 参照マネージャーの表示

参照マネージャーで「RMWCommunicationLibrary.dll」のチェックを外し、「OK」ボタンを押下します。

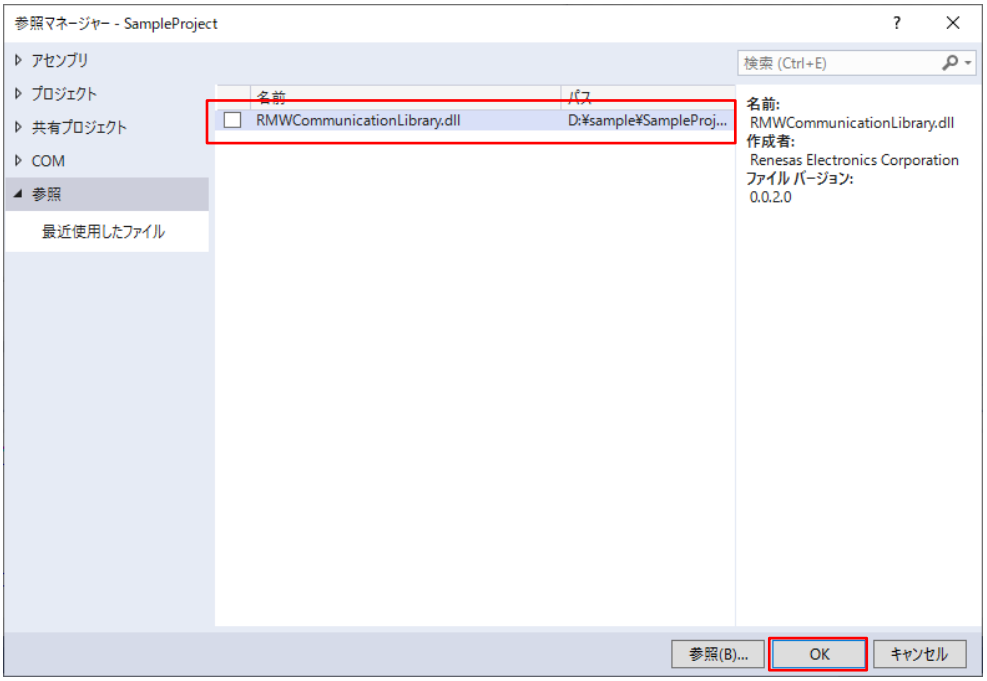


図 3-20 参照の登録解除

(2) プロジェクトから DLL を削除

画面上部から「表示」-「ソリューションエクスプローラー」を選択します。

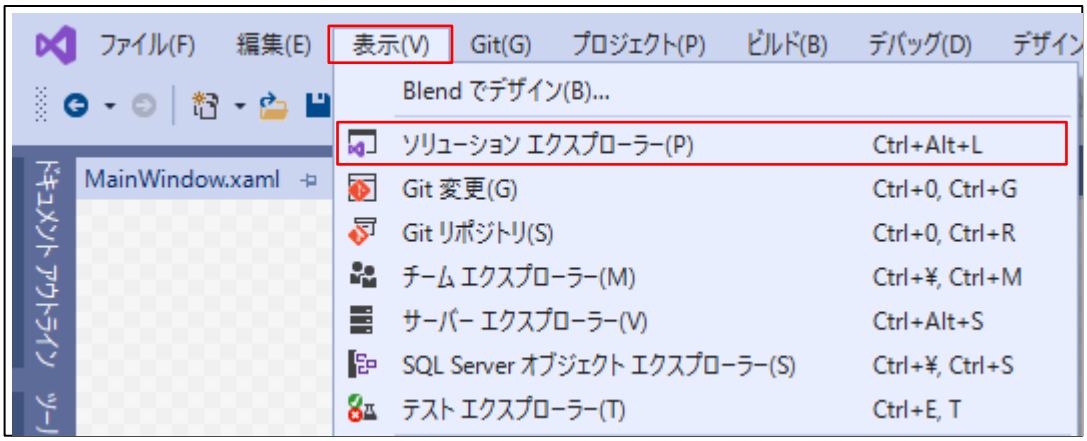


図 3-21 ソリューションエクスプローラーの表示

ソリューションエクスプローラーから、削除する DLL ファイルを右クリックし、「削除」を選択します。

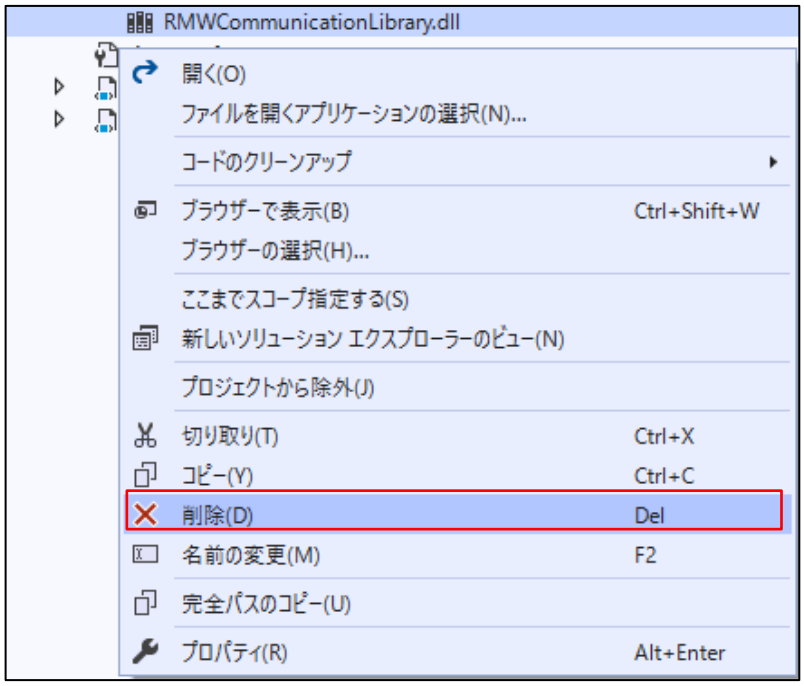


図 3-22 プロジェクトからファイルを削除

以下のメッセージが表示された場合、「OK」ボタンを押下します。

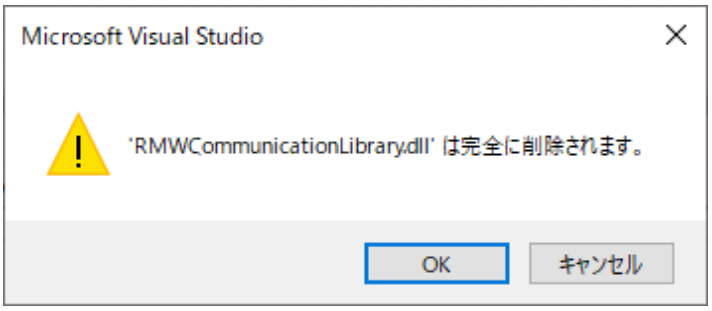


図 3-23 ファイル削除時の注意画面

(3) プロジェクトの保存

画面上部から「ファイル」-「すべて保存」を選択します。

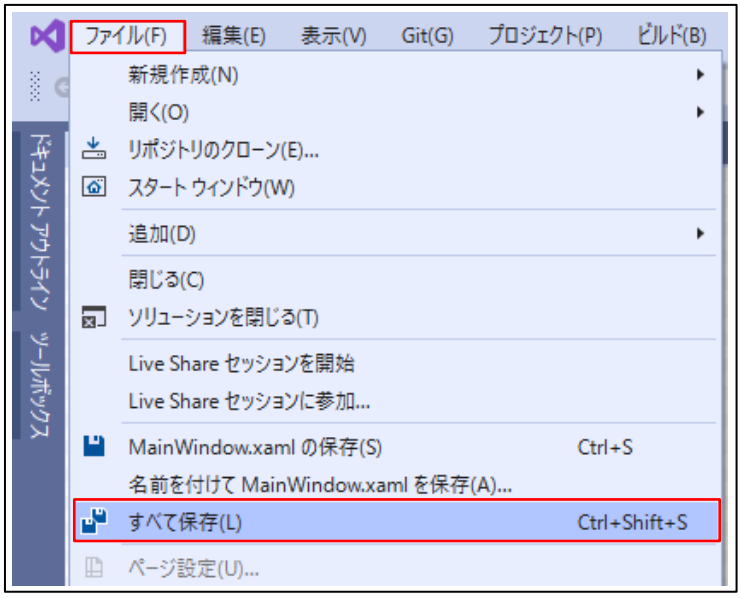


図 3-24 プロジェクトの保存

3.2.2 Excel からの削除方法

DLL を Excel から削除する方法を説明します。

[重要！] レジストリから DLL を解除する場合には、DLL ファイルが必要です。DLL 解除前に DLL ファイルの削除は行わないようにしてください。

(1) Excel の参照設定から DLL を解除

Excel のリボンに表示されている「開発」-「Visual Basic」を選択します。

図 3-25 Visual Basic 画面の表示

Visual Basic 画面の「ツール」-「参照設定」を選択します。

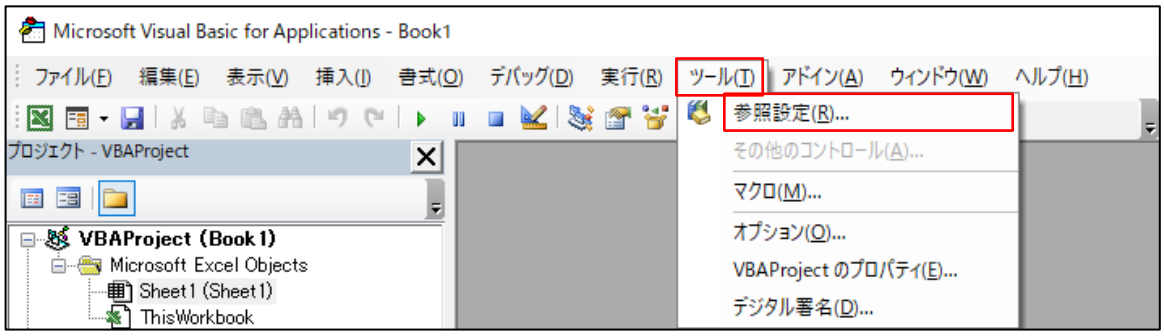


図 3-26 参照設定画面の表示

参照設定画面で「RMW Communication Library」のチェックを外し、「OK」ボタンを押下します。

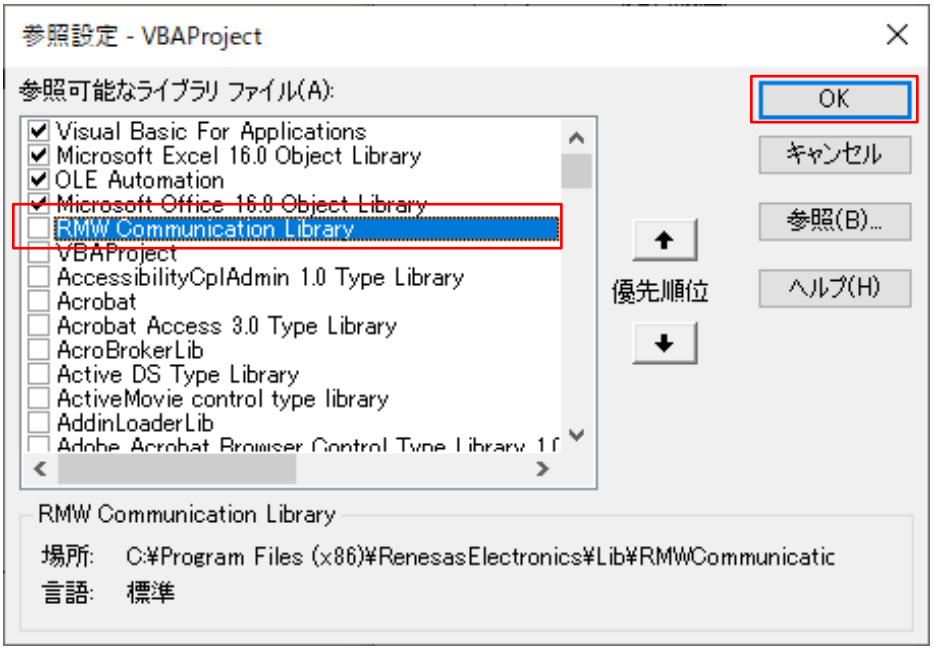


図 3-27 参照の登録解除

(2) ファイルの保存

Excel のリボンに表示されている「ファイル」を選択します。

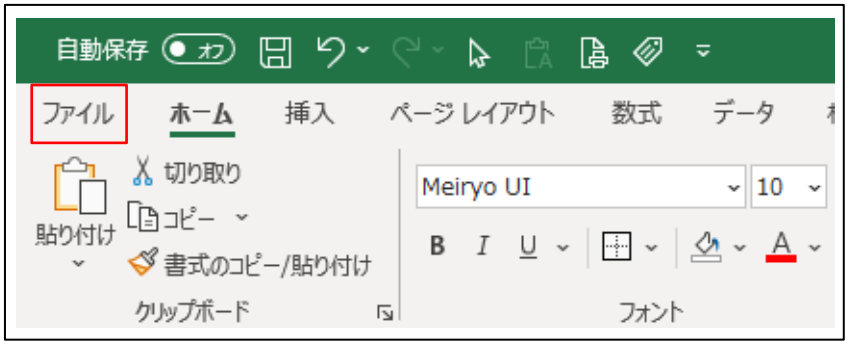


図 3-28 ファイルタブの選択

「上書き保存」または「名前を付けて保存」を選択します。

図 3-29 ファイルの保存

(3) レジストリから DLL を解除

スタートメニューから「Windows システム ツール」を選択します。

「コマンドプロンプト」を右クリックし、「その他」-「管理者として実行」を選択します。

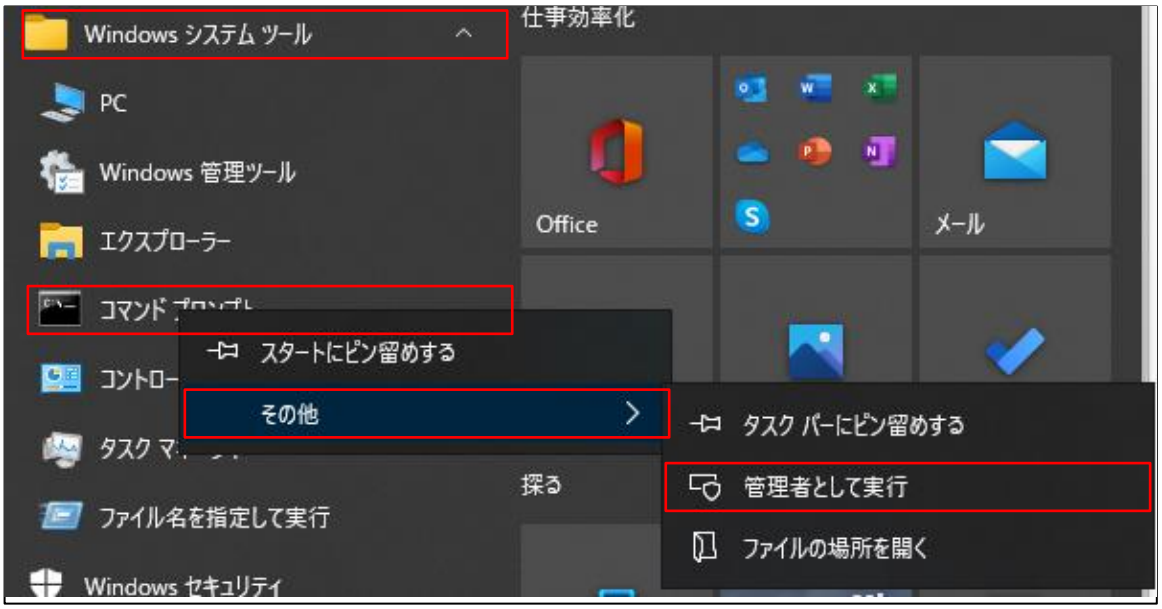


図 3-30 コマンドプロンプトの起動

コマンドプロンプトで、以下のコマンドを実行します。

【実行するコマンド】

(RegAsm.exe 格納パス)¥RegAsm.exe /tlb /unregister "(DLL 展開先パス)¥RMWCommunicationLibrary.dll"

RegAms.exe 格納パスは以下の通り。

64bit Excel：C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319

32bit Excel：C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319

実行する RegAsm.exe は OS の種類ではなく、Excel のバージョンで変更します。

※Excel のバージョンについては、以下の Web ページを参照してください。

https://support.microsoft.com/ja-jp/office/%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%81%84%E3%82%8B-office-%E3%81%AE%E3%83%90%E3%83%BC%E3%82%B8%E3%83%A7%E3%83%B3%E3%82%92%E7%A2%BA%E8%AA%8D%E3%81%99%E3%82%8B%E6%96%B9%E6%B3%95-932788b8-a3ce-44bf-bb09-e334518b8b19

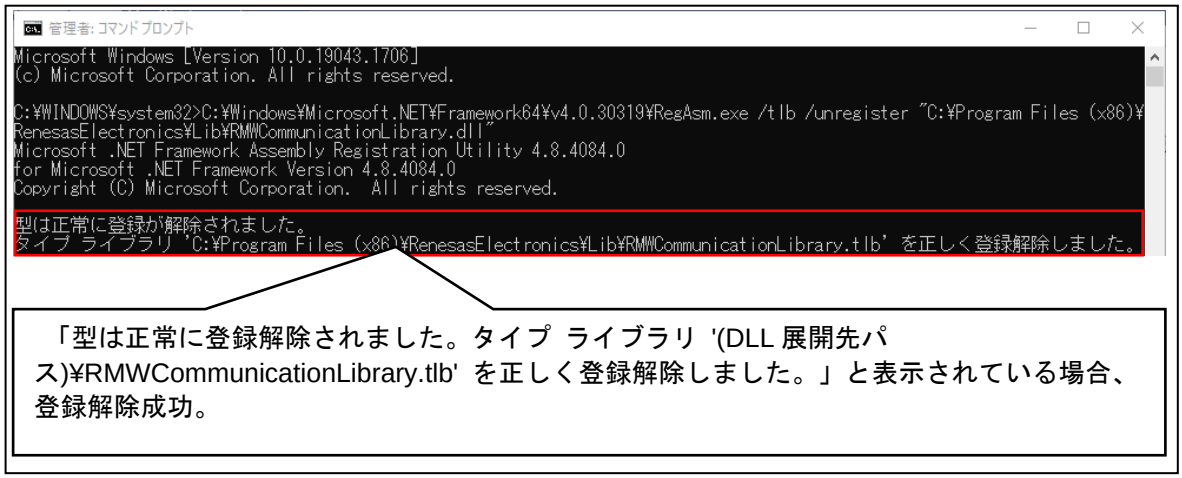


図 3-31 コマンドプロンプトの実行画面

(4) 展開した DLL を削除

導入時に展開した DLL ファイルを削除します。

※.dll 以外にもファイルが生成されていますが、登録時に生成されるファイルのため、合わせて削除します。

図 3-32 展開した DLL の削除

4. DLL の関数一覧

DLL の関数一覧を以下に示します。

表 4-1 関数一覧

クラス名	関数名	概要	引数	戻り値
ComCommunication	Connect	マイコンと接続する	第 1 引数：接続する COM ポート 第 2 引数：接続に使用するボーレート	接続処理の結果
	DisConnect	マイコンから切断する	-	切断処理の結果
	Read	マイコンから値を読み込む	第 1 引数：読み込んだ結果 第 2 引数：読み込むアドレス 第 3 引数：読み込むデータ型 第 4 引数：エンディアン	読み込み処理の結果
	Write	マイコンへ値を書き込む	第 1 引数：書き込むアドレス 第 2 引数：書き込む値 第 3 引数：書き込むデータ型 第 4 引数：エンディアン	書き込み処理の結果
	ScopeStart	Scope 処理を開始する	第 1 引数：トリガチャネル 第 2 引数：トリガレベル 第 3 引数：Scope 設定	Scope 処理の結果
	ScopeGetCondition	Scope 処理の状態を取得する	-	Scope 処理の状態
	ScopeGetData	Scope 処理のデータを取得する	第 1 引数：読み込みインデックス 第 2 引数：読み込みデータ数 第 3 引数：取得データ格納用変数 第 4 引数：エンディアン	データの取得状態
	ScopeStop	Scope 処理を停止する	-	停止処理の結果
MapConversion	ConvertMapToMemory	Map ファイルの変換結果をメモリ上に展開する	第 1 引数：マップファイルパス 第 2 引数：変数用プレフィックス 第 3 引数：配列用プレフィックス 第 4 引数：データ上限数 第 5 引数：データ変換数 第 6 引数：データ格納数 第 7 引数：データ格納用変数 第 8 引数：コンパイラ種別	変換処理の結果
	ConvertMapToCSV	Map ファイルの変換結果を CSV ファイルに出力する	第 1 引数：マップファイルパス 第 2 引数：変数用プレフィックス 第 3 引数：配列用プレフィックス 第 4 引数：出力ファイルパス 第 5 引数：上書きフラグ	変換処理の結果
ScopeSetting	SetProcessInfo	Scope 処理に必要なサンプリングタイム、ポジション、取得データ数等を設定する	第 1 引数：サンプル時間 第 2 引数：ポジション 第 3 引数：取得データ数 第 4 引数：トリガエッジ 第 5 引数：トリガモード	設定処理の結果

	AddChannelInfo	Scope 処理に必要なチャネル情報を追加する	第 1 引数：読み込むアドレス 第 2 引数：チャネル番号 第 3 引数：データ型 第 4 引数：スケール	追加処理の結果
	RemoveChannelInfo	引数で指定したチャネル情報を削除する	第 1 引数：削除するチャネル番号	削除処理の結果
	ClearChannelInfo	登録したチャネル情報を全て削除する	-	-

5. 各関数の機能説明

5.1 接続関数(Connect)

本関数は、引数で指定した COM ポート、ボーレートを使用し、マイコンと接続します。

既に接続した状態で本関数を再度実行した場合、通信を切断し再度接続処理を行います。

表 5-1 接続関数の引数

#	内容	型	I/O	有効値
1	COM ポート	string	I	Windows 上で認識している COM ポート番号
2	ボーレート	int	I	1～999,999

接続後、CPU 情報収集コマンドを送信し、返ってきた応答を判定し、接続の成否を判定します。

接続処理に成功した場合、戻り値を処理成功で返します。

接続処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-2 接続関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：COM ポート不正	COM ポートの設定を確認後、再接続する。
3			2：ボーレート不正	ボーレートの設定を確認後、再接続する。
4			3：接続失敗	ツール用通信ボードの電源が入っていることを確認後、再接続する。
5			4：コマンドの応答が不正	対応しているマイコンか確認後、再接続する。

5.2 切断関数(DisConnect)

本関数は、接続関数で接続したマイコンとの切断を行います。

切断処理に成功した場合、戻り値を処理成功で返します。

切断処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-3 切断関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：接続されていない	接続関数が実行されているか確認する。 ツール用通信ボードと接続されているか確認する。

5.3 読み込み関数(Read)

本関数は、【各データ型の読み込みイメージ】のように、引数で指定したアドレスを先頭アドレスとして、変数の型に応じたサイズ分、マイコンから値を読み込みます。

表 5-4 読み込み関数の引数

#	内容	型	I/O	有効値
1	読み込み結果	string	O	-
2	読み込むアドレス	int	I	0x00000000 ～ 0x7FFFFFFF
3	読み込むデータ型	enum	I	ReadUInt8、ReadInt8、ReadUInt16、ReadInt16、ReadUInt32、ReadInt32、ReadFloat、ReadBool、ReadLogic
4	エンディアン	enum	I	Little、Big

※読み込むデータ型とエンディアンについては、以下の enum 定義に対応する値を指定します。

表 5-5 読み込むデータ型の定義

#	enum 名	値	概要
1	ReadUInt8	0	unsigned char 型のデータを読み込む
2	ReadInt8	1	signed char 型のデータを読み込む
3	ReadUInt16	2	unsigned short 型のデータを読み込む
4	ReadInt16	3	signed short 型のデータを読み込む
5	ReadUInt32	4	unsigned long 型のデータを読み込む
6	ReadInt32	5	signed long 型のデータを読み込む
7	ReadFloat	6	float 型のデータを読み込む
8	ReadBool	7	bool 型のデータを読み込む
9	ReadLogic	8	logic 型のデータを読み込む

表 5-6 エンディアンの定義

#	enum 名	値	概要
1	Little	0	Little エンディアンで読み込む
2	Big	1	Big エンディアンで読み込む

表 5-7 対応している型とサイズ

#	型	サイズ
1	bool	1
2	logic	1
3	signed char	1
4	unsigned char	1
5	signed short	2
6	unsigned short	2
7	signed long	4
8	unsigned long	4
9	float	4

【各データ型の読み込みイメージ】

〇8Bit(1byte)の値を読み込み

1008 番地から 8bit(1byte)のデータを読み込む場合

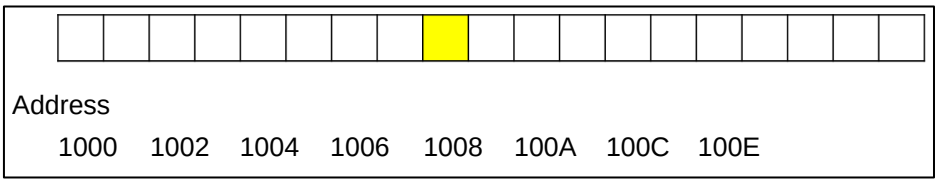


図 5-1 8Bit 読み込みイメージ

〇16Bit(2byte)の値を読み込み

1008 番地から 16bit(2byte)のデータを読み込む場合

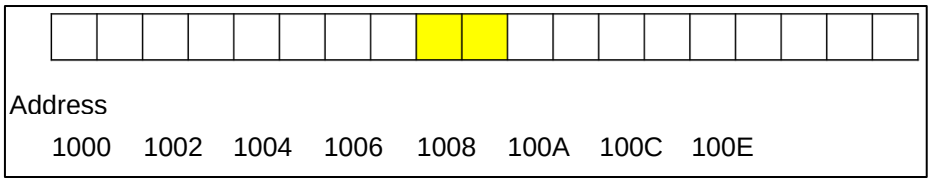


図 5-2 16Bit 読み込みイメージ

〇32Bit(4byte)の値を読み込み

1008 番地から 32bit(4byte)のデータを読み込む場合

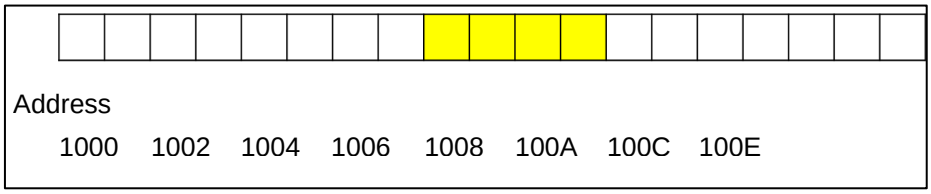


図 5-3 32Bit 読み込みイメージ

読み込んだデータは、以下の図のように、指定したエンディアンで変換します。

○リトルエンディアンを指定している場合

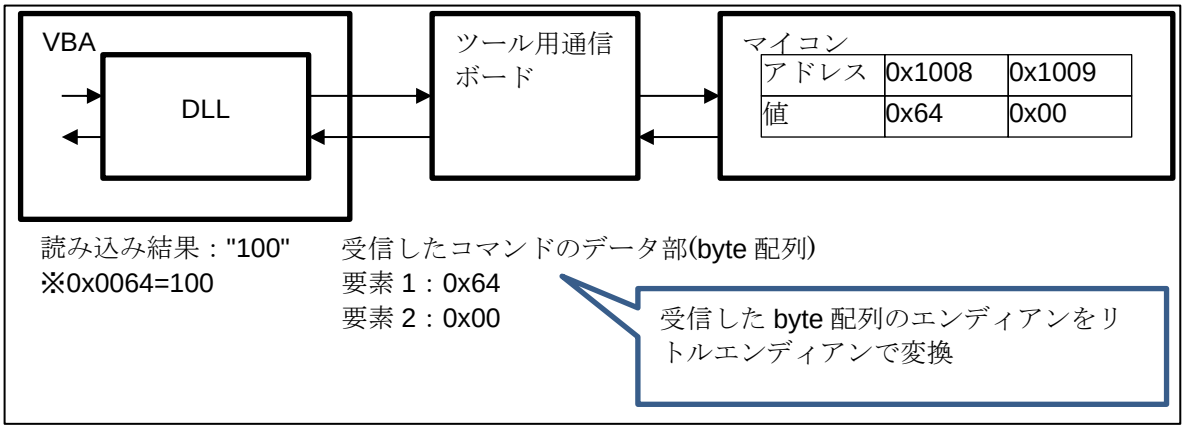


図 5-4 リトルエンディアン読み込みイメージ

○ビッグエンディアンを指定している場合

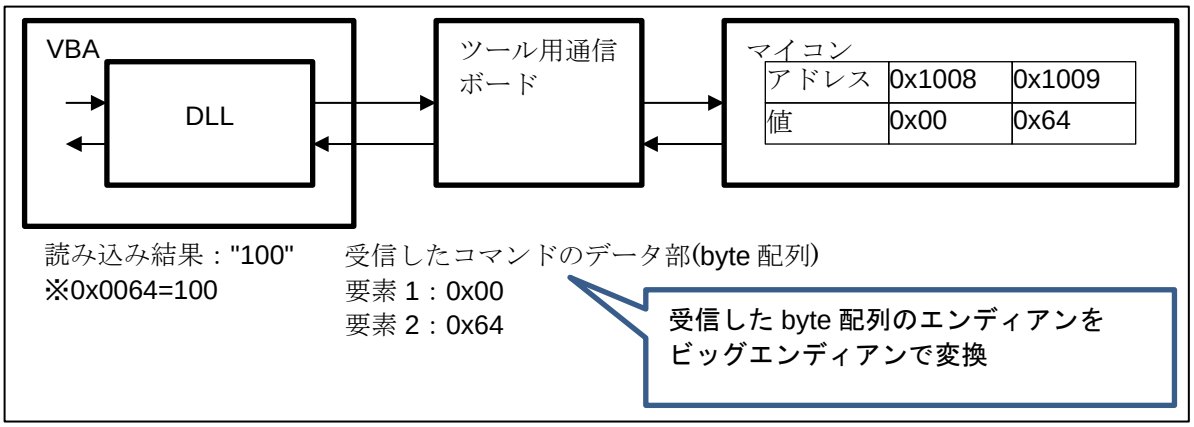


図 5-5 ビッグエンディアン読み込みイメージ

読み込み処理に成功した場合、引数に読み込んだ結果を格納し、戻り値を処理成功で返します。

読み込み処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-8 読み込み関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、読み込み関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、読み込み関数を実行する。
3			2：マイコンとの通信に失敗	接続関数を未実行の場合は、接続関数を実行後、読み込み関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、読み込み関数を実行する。
4			3：アドレスに指定された値が不正	アドレス設定の確認後、再度読み込み関数を実行する。
5			4：データ型が不正	データ型設定の確認後、再度読み込み関数を実行する。
6			5：Endian が不正	Endian 設定の確認後、再度読み込み関数を実行する。

5.4 書き込み関数(Write)

本関数は、【各データ型の書き込みイメージ】のように、引数で指定したマイコンのアドレスに引数で指定した値を書き込みます。

表 5-9 書き込み関数の引数

#	内容	型	I/O	有効値
1	書き込みアドレス	int	I	0x00000000 ～ 0x7FFFFFFF
2	書き込み値	string	I	（データ型により変動）
3	書き込みデータ型	enum	I	WriteUInt8、WriteInt8、WriteUInt16、WriteInt16、WriteUInt32、WriteInt32、WriteFloat、WriteBool、WriteLogic
4	エンディアン	enum	I	Little、Big

※書き込む値はデータ型により有効範囲が変わるため、文字列型で設定します。
書き込みデータ型とエンディアンについては、以下の enum 定義に対応する値を指定します。

表 5-10 書き込みデータ型の定義

#	enum 名	値	概要
1	WriteUInt8	0	unsigned char 型のデータを書き込む
2	WriteInt8	1	signed char 型のデータを書き込む
3	WriteUInt16	2	unsigned short 型のデータを書き込む
4	WriteInt16	3	signed short 型のデータを書き込む
5	WriteUInt32	4	unsigned long 型のデータを書き込む
6	WriteInt32	5	signed long 型のデータを書き込む
7	WriteFloat	6	float 型のデータを書き込む
8	WriteBool	7	bool 型のデータを書き込む
9	WriteLogic	8	logic 型のデータを書き込む

表 5-11 エンディアンの定義

#	enum 名	値	概要
1	Little	0	Little エンディアンで書き込む
2	Big	1	Big エンディアンで書き込む

表 5-12 対応している型とサイズ

#	型	サイズ
1	bool	1
2	logic	1
3	signed char	1
4	unsigned char	1
5	signed short	2
6	unsigned short	2
7	signed long	4
8	unsigned long	4
9	float	4

【各データ型の書き込みイメージ】

〇8Bit(1byte)の値を書き込み

1004 番地に 8bit(1byte)のデータ 0x01 を書きこむ場合

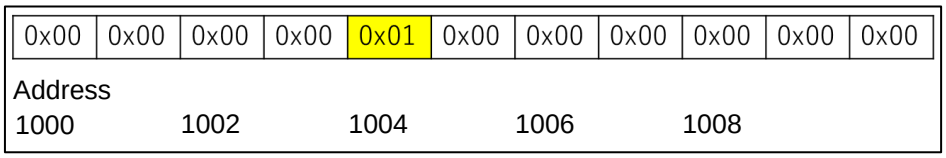


図 5-6 8Bit 書き込みイメージ

〇16Bit(2byte)の値を書き込み

1004 番地に 16bit(2byte)のデータ 0x0001 を書きこむ場合

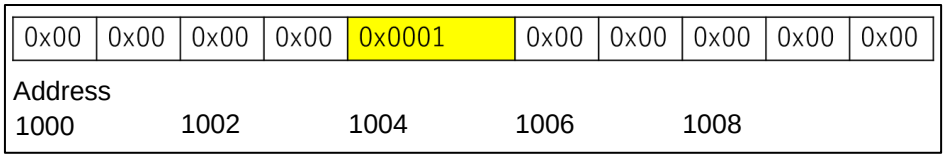


図 5-7 16Bit 書き込みイメージ

〇32Bit(4byte)の値を書き込み

1004 番地に 32bit(4byte)のデータ 0x0000 0001 を書き込む場合

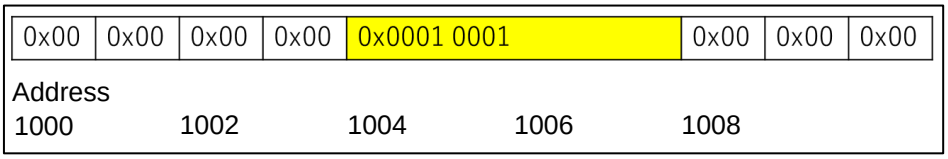


図 5-8 32Bit 書き込みイメージ

書き込むデータは、指定したエンディアンで byte 配列に変換して送信します。

○リトルエンディアンを指定している場合

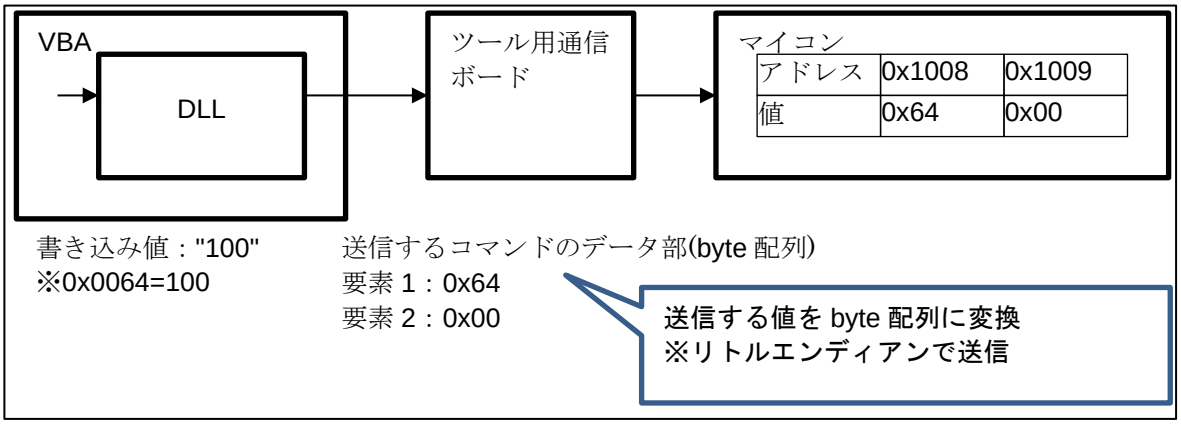


図 5-9 リトルエンディアン書き込みイメージ

○ビッグエンディアンを指定している場合

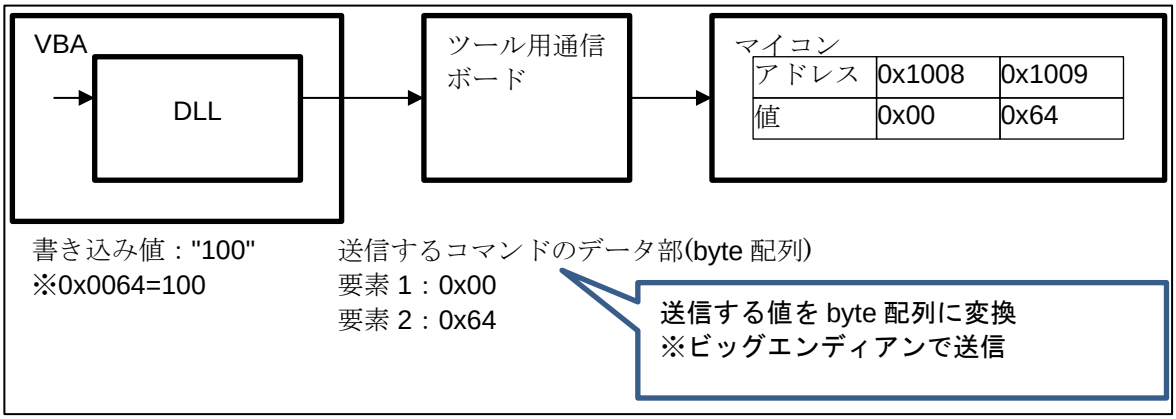


図 5-10 ビッグエンディアン書き込みイメージ

書き込み処理に成功した場合、戻り値を処理成功で返します。

書き込み処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-13 書き込み関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、書き込み関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、書き込み関数を実行する。
3			2：アドレスが不正	アドレス設定の確認後、再度書き込み関数を実行する。
4			3：書き込み値が範囲外	書き込み値設定の確認後、再度書き込み関数を実行する。
5			4：マイコンとの通信失敗	接続関数を未実行の場合は、接続関数を実行後、書き込み関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、書き込み関数を実行する。
6			5：データ型が不正	データ型設定の確認後、再度書き込み関数を実行する。
7			6：Endian が不正	Endian 設定の確認後、再度書き込み関数を実行する。

5.5 Scope 設定関数(SetProcessInfo)

本関数は、引数で指定した Scope 処理に必要な情報を登録します。

複数回呼び出した場合、後に呼び出した設定値に上書きされます。

表 5-14 Scope 設定関数の引数

#	内容	型	I/O	設定する値	有効値
1	サンプリングタイム	int	I	サンプリング時間（μs）	20～4000
2	ポジション	double	I	トリガを判定する位置	0～1000
3	取得レコード数	int	I	取得するレコード数	20～マイコン毎の上限値
4	トリガエッジ	enum	I	Rise、Down、Both	Rise、Down、Both
5	トリガモード	enum	I	Single、Normal、Auto	Single、Normal、Auto

※トリガエッジとトリガモードについては【トリガエッジについて】、【トリガモードについて】を参照してください。

【トリガエッジについて】

トリガエッジが「Rise(上がり検知)」の場合、トリガとなるチャネルの値がトリガレベルの値を上回った場合に、トリガ条件を満たしたものとして、各チャネルのデータを取得します。

トリガエッジが「Down(下り検知)」の場合、トリガとなるチャネルの値がトリガレベルの値を下回った場合に、トリガ条件を満たしたものとして、各チャネルのデータを取得します。

トリガエッジが「Both(両検知)」の場合、トリガとなるチャネルの値がトリガレベルの値を上回ったもしくは下回った場合に、トリガ条件を満たしたものとして、各チャネルのデータを取得します。

表 5-15 トリガエッジの定義

#	enum 名	値	概要
1	Rise	0	上がり検知
2	Down	1	下り検知
3	Both	2	両検知

【トリガモードについて】

トリガモードが「Single(1 回取得)」及び「Normal(トリガ条件を満たす度に取得)」の場合、トリガ条件が満たされた場合に、データを取得します。

トリガモードが「Auto(常時取得)」の場合、即時データを取得します。

表 5-16 トリガモードの定義

#	enum 名	値	概要
1	Single	0	トリガ条件が満たされた場合データ取得
2	Normal	1	トリガ条件が満たされた場合データ取得
3	Auto	2	即時データ取得

Scope 設定処理に成功した場合、戻り値を処理成功で返します。

Scope 設定処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-17 Scope 設定関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：トリガエッジが不正	トリガエッジ設定の確認後、Scope 設定関数を再実行する。
3			2：トリガモードが不正	トリガモード設定の確認後、Scope 設定関数を再実行する。

5.6 Scope チャネル追加関数(AddChannelInfo)

本関数は、引数で指定したチャネル情報を、収集するチャネル情報に追加します。

表 5-18 Scope チャネル追加関数の引数

#	内容	型	I/O	設定する値
1	アドレス情報	int	I	0x00000000 ～ 0x7FFFFFFF
2	チャネル番号	byte	I	（マイコンにより変動）
3	データの種類	enum	I	UINT8、INT8、UINT16、INT16、UINT32、INT32、FLOAT、BOOL、LOGIC
4	スケール情報	byte	I	1～255

※データの種類については、以下の enum 定義に対応する値を指定します。

表 5-19 データの種類の定義

#	enum 名	値	概要
1	UINT8	0	unsigned char 型のデータ
2	INT8	1	signed char 型のデータ
3	UINT16	2	unsigned short 型のデータ
4	INT16	3	signed short 型のデータ
5	UINT32	4	unsigned long 型のデータ
6	INT32	5	signed long 型のデータ
7	FLOAT	6	float 型のデータ
8	BOOL	7	bool 型のデータ
9	LOGIC	8	logic 型のデータ

Scope チャネル追加処理に成功した場合、戻り値を処理成功で返します。

Scope チャネル追加処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-20 Scope チャネル追加関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：データの種類が不正	データ型設定の確認後、チャネル情報追加関数を再実行する。
3			2：ボードが対応していないデータ型	データ型設定の確認後、チャネル情報追加関数を再実行する。

5.7 Scope チャネル削除関数(RemoveChannelInfo)

本関数は、引数で指定したチャネル番号のチャネル情報を削除します。

表 5-21 Scope チャネル削除関数の引数

#	内容	型	I/O	設定する値
1	削除するチャネル番号	byte	I	0～31(既に登録されているチャネル番号)

Scope チャネル削除処理に成功した場合、戻り値を処理成功で返します。

Scope チャネル削除処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-22 Scope チャネル削除関数の戻り値と対応

#	内容	型	戻り値
1	処理の結果	int	0：処理成功
2			1：指定されたチャネルが存在しない

5.8 Scope チャネル情報全削除関数(ClearChannelInfo)

本関数は、実行時点で登録されている全てのチャネル情報を削除します。

本関数は引数無しで実行し、戻り値も返しません。

5.9 Scope 処理の関数について

Scope 処理は、ScopeStart 関数、ScopeGetCondition 関数、ScopeGetData 関数、ScopeStop 関数を使用します。

表 5-23 Scope 処理に使用する関数

#	関数名	概要
1	ScopeStart	Scope 処理の開始
2	ScopeGetCondition	Scope 処理の状態を取得
3	ScopeGetData	Scope 処理で取得したデータを取得
4	ScopeStop	Scope 処理を停止

Scope 処理を実行する場合、以下のような順で各関数を実行します。

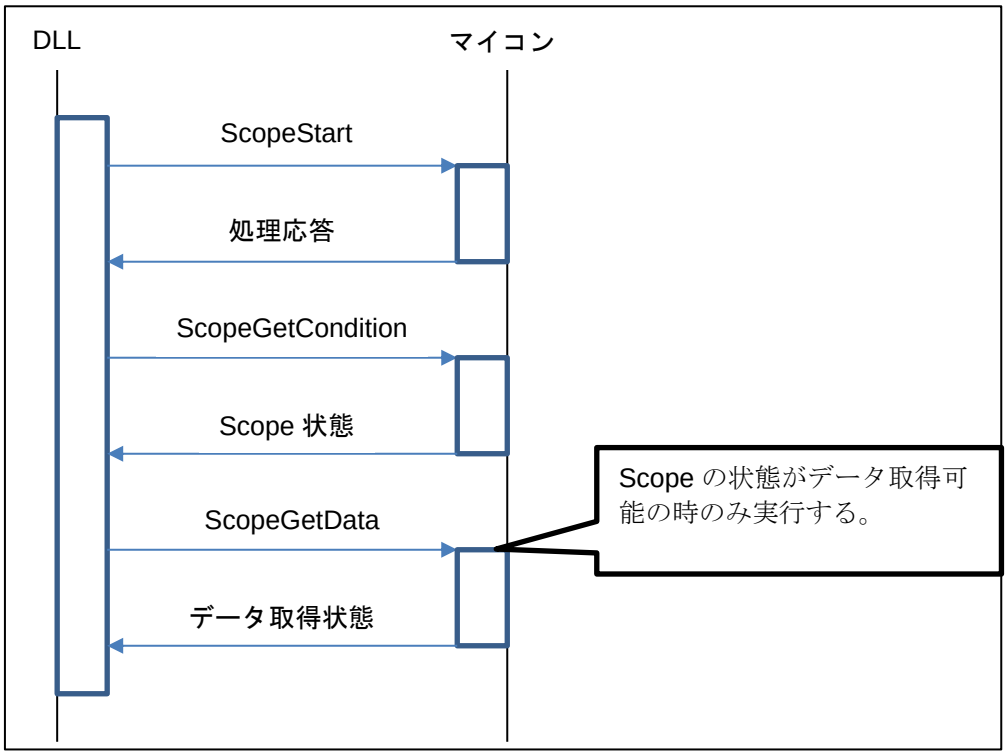


図 5-11 Scope 処理の流れのイメージ

上記の流れで、指定したチャネルの変数の値を連続的にマイコンから読み込みます。

受信したデータは、以下の図のように、指定したエンディアンで変換し、読み込み結果として返します。

○リトルエンディアンを指定している場合

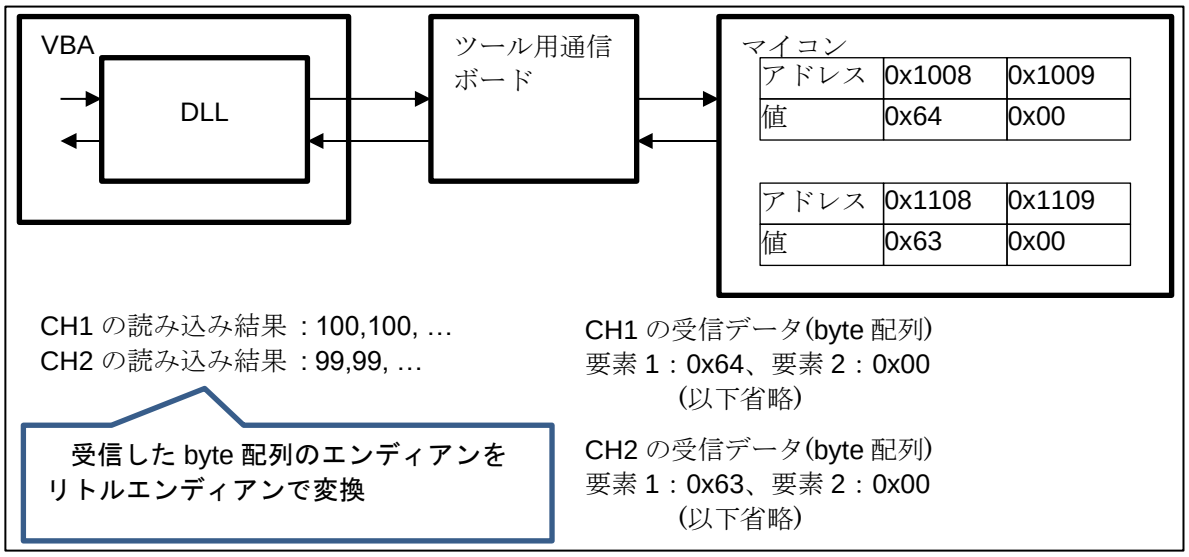


図 5-12 リトルエンディアン連続読み込みイメージ

○ビッグエンディアンを指定している場合

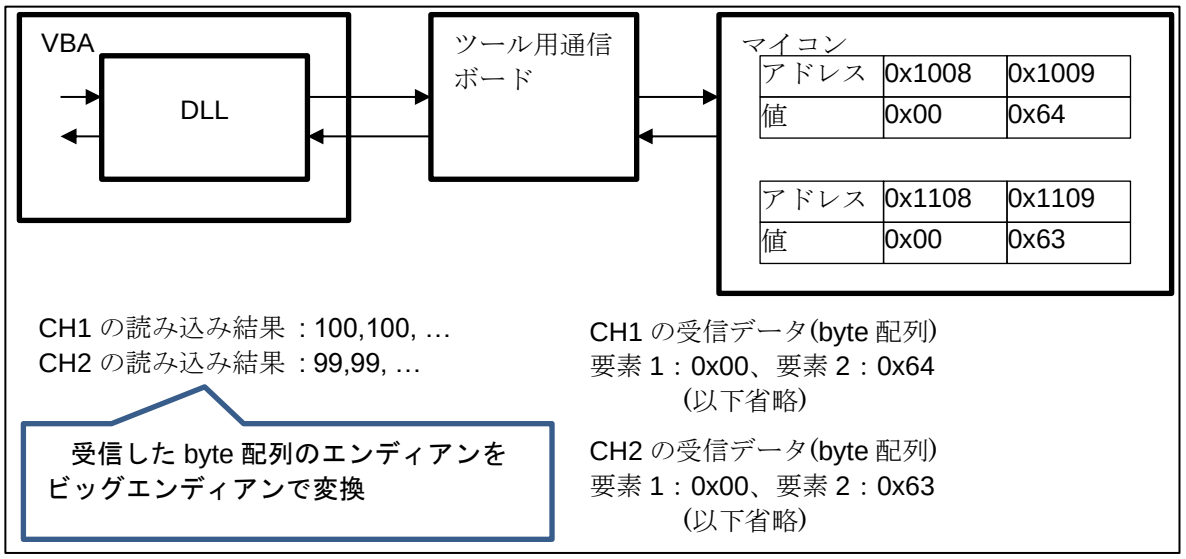


図 5-13 ビッグエンディアン連続読み込みイメージ

5.9.1 Scope 開始関数(ScopeStart)

本関数は、引数で指定した Scope の設定を用いて、チャネル情報に設定した変数の値を取得します。

引数の値をチェックし、正常な場合 Scope 処理を開始します。

表 5-24 Scope 開始関数の引数

#	内容	型	I/O	有効値
1	トリガチャネル	byte	I	接続処理時に取得したチャネルの最大数（マイコンにより変動）
2	トリガレベル	double	I	-3.40282346638529E+38～3.40282346638529E+38
3	Scope 処理の設定	ScopeSetting	I	サンプリングタイム、取得データ数、ポジション等の Scope 処理に必要な情報を設定

※ScopeSetting については【ScopeSetting クラスについて】を参照してください。

【ScopeSetting クラスについて】

ScopeSetting クラスは、Scope 処理に必要な情報及び取得するチャネルに関する設定を登録するクラスです。

ScopeStart 関数実行前に、以下の関数を用いて Scope 処理に必要な情報及び取得するチャネルに関する設定を行います。

Scope 処理に必要な情報の登録：「5.5Scope 設定関数(SetProcessInfo)」参照

チャネル情報の登録　　：「5.6Scope チャネル追加関数(AddChannelInfo)」参照

指定したチャネル情報の削除：「5.7Scope チャネル削除関数(RemoveChannelInfo)」参照

チャネル情報の全削除　　：「5.8Scope チャネル情報全削除関数(ClearChannelInfo)」参照

Scope 開始処理に成功した場合、戻り値を処理成功で返します。

Scope 開始処理に失敗した場合、戻り値を失敗原因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-25 Scope 開始関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：スコープ設定が不正	スコープ設定を確認後、Scope 開始関数を再実行する。
3			2：トリガレベルが不正	トリガレベル設定を確認後、Scope 開始関数を再実行する。
4			3：チャネル設定が不正	チャネル設定を確認後、Scope 開始関数を再実行する。
5			4：Position の設定が不正	ポジション設定を確認後、Scope 開始関数を再実行する。
6			5：レコード長の設定が不正	取得レコード数を確認後、Scope 開始関数を再実行する。
7			6：サンプリング周期が不正	サンプリング周期を確認後、Scope 開始関数を再実行する。
8			7：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。
9			8：コマンド通信失敗	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。

5.9.2 Scope 状態取得関数(ScopeGetCondition)

本関数は、Scope 開始関数にて開始した Scope 処理の状態を取得します。

Scope 開始関数が既に実行されている場合のみ、本関数を実行します。

本関数は引数なしで実行します。

Scope 状態取得処理に成功し、データ取得可能な場合、戻り値を処理成功で返します。

Scope 状態取得処理に成功したが、データ取得ができない状態の場合、戻り値をデータ取得準備中で返します。

Scope 状態取得処理に失敗した場合、戻り値を失敗要因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-26 Scope 状態取得関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：データ取得準備中	再度状態取得関数を実行する。
3			2：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数から再実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。
4			3：コマンド通信失敗	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。

5.9.3 Scope データ取得関数(ScopeGetData)

本関数は、Scope 処理にて収集したマイコンにバッファされているデータを取得します。

Scope 状態取得関数の処理に成功した場合(データ取得可能状態の場合)のみ、本関数を実行してください。

本関数は、マイコンのバッファの読み込み開始インデックス及び読み込みデータ数を指定してマイコンからデータを取得します。

表 5-27 Scope データ取得関数の引数

#	内容	型	I/O	有効値
1	読み込みインデックス	int	I	0 以上
2	読み込みデータ数	int	I	1 以上
3	取得データ格納変数	GetData[]	O	取得したデータを格納する構造体
4	エンディアン	enum	I	Little、Big

※取得データ格納変数については【GetData 構造体について】を参照してください。
エンディアンについては、以下の enum 定義に対応する値を指定します。

表 5-28 エンディアンの定義

#	enum 名	値	概要
1	Little	0	Little エンディアンで書き込む
2	Big	1	Big エンディアンで書き込む

【GetData 構造体について】

GetData 構造体は取得したデータ、取得したデータ数、取得するデータの最大数を設定する構造体です。

表 5-29 GetData 構造体に定義している変数

#	変数名	型	I/O	概要	有効値
1	ScopeData	double[]	I/O	取得したデータを格納する配列	(値の設定は関数内で行う)
2	DataCount	int	O	取得したデータ数	(値の設定は関数内で行う)
3	CountLimit	int	I	取得するデータの最大数	1 以上

ScopeGetData 関数実行前に本構造体に対して以下の設定を行います。

(1)ScopeData を初期化する。

(2)CountLimit に(1)で割り当てた配列の要素数以下の値を設定する。

※ScopeGetData 関数は CountLimit で指定した値までのデータ数を配列に格納します。
必ず設定した配列の要素数以下の値を設定してください。

上記の操作を各チャネル分行います。

Scope データ取得処理に成功し、取得するデータの最大数までデータの取得が完了している場合、戻り値を処理成功で返します。

データの取得については、引数で指定したエンディアンで受信したデータを変換後、変数に格納します。

Scope データ取得処理に成功したが、取得するデータの最大数までデータの取得が完了していない場合、再度、Scope データ取得関数を実行してください。

Scope データ取得処理に失敗した場合、戻り値を失敗要因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-30 Scope データ取得関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功(データ取得完了)	-
2			1：読み込み範囲の設定が不正	読み込みインデックス、読み込みデータ数の値を確認する。
3			2：取得データ格納配列の設定が不正(最大取得数に値が入っていない)	取得データ格納変数の設定を確認後、データ取得関数を再実行する。
4			3：状態がデータ取得可能ではない	状態取得関数を実行し、データ取得可能になるまで待つ。
5			4：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数から再実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。
6			5：エンディアン設定が不正	エンディアン設定を確認後、データ取得関数を再実行する。
7			6：Scope 設定が未設定	Scope 開始関数から実行する。
8			7：コマンド通信失敗	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。

5.9.4 Scope 停止関数(ScopeStop)

本関数は、Scope 開始関数で開始した Scope 処理を停止します。

Scope 開始関数を実行後、停止させるタイミングでのみ本関数を実行してください。

本関数実行後、再度 Scope 処理を行う場合は、Scope 開始関数から実行してください。

本関数は引数なしで実行します。

Scope 停止処理に成功した場合、戻り値を処理成功で返します。

Scope 停止処理に失敗した場合、戻り値を失敗要因に対応する値で返します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-31 Scope 停止関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：マイコンと接続されていない	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を再実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。
3			2：コマンド通信失敗	接続関数を未実行の場合は、接続関数を実行後、Scope 開始関数を再実行する。 接続関数を実行済みの場合は、ツール用通信ボードの接続を確認し、接続関数を実行し、Scope 開始関数を実行する。

5.10 Map ファイル変換機能について

本機能は、指定された Map ファイルを変換し、メモリ上に展開、もしくは CSV ファイルに出力します。

対応しているコンパイラは、以下の表の通りです。

表 5-32 対応コンパイラ

#	コンパイラ
1	CC
2	CA

メモリ上に展開する処理及び CSV ファイルを出力する処理が実行された時に共通で、Map ファイルの読み込み及び CSV 形式(カンマ区切り)への変換処理を実行します。

CSV 形式への変換処理が完了後、実行された関数に応じてメモリ上に展開するか、CSV 形式のファイルに出力します。

5.10.1 Map ファイル変換メモリ展開関数(ConvertMapToMemory)

本関数は、引数で指定した Map ファイルの内容を変換し、引数で指定した変換結果格納用の変数に格納します。

また、Map ファイルを解析し変換した変数の数については、引数で指定した変換数の変数に格納します。

本関数は、変換結果格納用の変数の上限数より Map ファイルに登録されている変数が多い場合、可能な限り変換結果格納用の変数に値を格納します。

変換結果格納用の変数に格納した変数の数については、引数で指定した格納数の変数に格納します。

データ上限、変換数、格納数の関係については以下を参照してください。

・データ上限より変換数が少ない場合(データ上限：5、変換数：3)

以下の場合、変換数には Map ファイルから変換できた数が入ります。

変換数と格納数には 3 が設定されます。

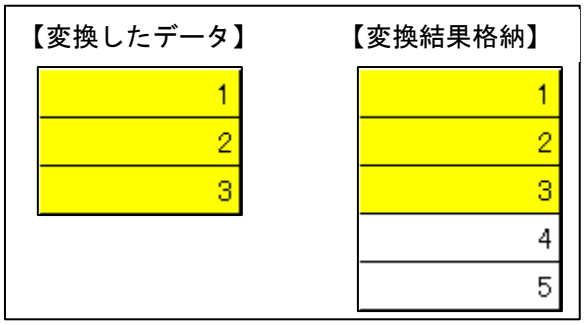


図 5-14 データ上限、変換数、格納数の関係(データ上限>変換数)

・データ上限より変換数が多い場合(データ上限：5、変換数：7)

以下の場合、変換したデータ数よりデータ上限が少ないため、一部のデータは変換結果格納用変数に設定されません。

変換数には 7、格納数には 5 が設定されます。

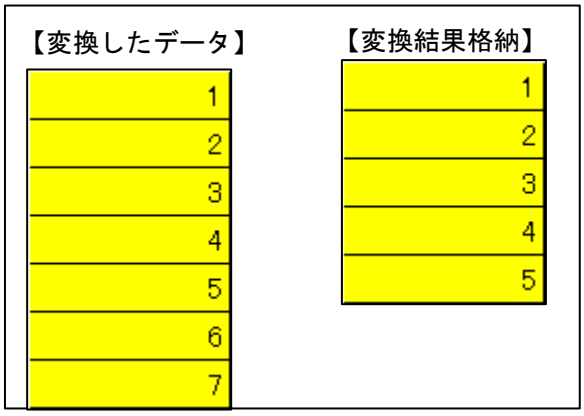


図 5-15 データ上限、変換数、格納数の関係(データ上限<変換数)

表 5-33 Map ファイル変換メモリ展開関数の引数

#	変数名	型	I/O	有効値
1	マップファイルパス	string	I	実際に存在する Map ファイルのパス
2	変数用プレフィックス	VariablePrefixDefined	I	型毎にプレフィックス情報を設定するクラス
3	配列用プレフィックス	ArrayPrefixDefined	I	型毎にプレフィックス情報を設定するクラス
4	データ上限	int	I	1 以上
5	変換数	int	O	（値の設定は関数内で行う）
6	格納数	int	O	（値の設定は関数内で行う）
7	変換結果格納用	ConvData	O	（値の設定は関数内で行う）
8	コンパイラ種別	byte	O	（値の設定は関数内で行う）

※VariablePrefixDefined、ArrayPrefixDefined、ConvData については【VariablePrefixDefined クラスについて】、【ArrayPrefixDefined クラスについて】、【ConvData クラスについて】を参照してください。

【VariablePrefixDefined クラスについて】

VariablePrefixDefined クラスは、変数の型判定用のプレフィックスを設定するクラスです。

表 5-34 VariablePrefixDefined クラスの変数

#	変数名	型	I/O	概要	有効値	設定例
1	UINT8	string	I	UINT8 と判定する prefix 情報を定義する	CSV 形式の文字列	g_u1_,com_u1
2	INT8	string	I	INT8 と判定する prefix 情報を定義する		g_s1_
3	UINT16	string	I	UINT16 と判定する prefix 情報を定義する		g_u2_,com_u2_
4	INT16	string	I	INT16 と判定する prefix 情報を定義する		g_s2_
5	UINT32	string	I	UINT32 と判定する prefix 情報を定義する		g_u4_,com_u4
6	INT32	string	I	INT32 と判定する prefix 情報を定義する		g_s4_
7	Float	string	I	Float と判定する prefix 情報を定義する		g_f4_

型毎にプレフィックス情報を定義します。

1 つの型に対して複数のプレフィックス情報を定義する場合、CSV 形式(カンマ区切り)で値を設定します。

プレフィックスの定義がない場合、サイズのみで判断するため、全てサイズに応じた singed 型になります。

表 5-35 サイズ毎の型

サイズ	型
1	INT8
2	INT16
4	FLOAT

【ArrayPrefixDefined クラスについて】

ArrayPrefixDefined クラスは、配列の型判定用のプレフィックスを設定するクラスです。

表 5-36 ArrayPrefixDefined クラスの変数

#	変数名	型	I/O	概要	有効値	設定例
1	UINT8	string	I	UINT8 の配列と判定する prefix 情報を定義する	CSV 形式の文字列	g_Array_u1_
2	INT8	string	I	INT8 の配列と判定する prefix 情報を定義する		g_Array_s1_
3	UINT16	string	I	UINT16 の配列と判定する prefix 情報を定義する		g_Array_u2_
4	INT16	string	I	INT16 の配列と判定する prefix 情報を定義する		g_Array_s2_
5	UINT32	string	I	UINT32 の配列と判定する prefix 情報を定義する		g_Array_u4_
6	INT32	string	I	INT32 の配列と判定する prefix 情報を定義する		g_Array_s4_
7	Float	string	I	Float の配列と判定する prefix 情報を定義する		g_Array_f4_

型毎にプレフィックス情報を定義します。

1 つの型に対して複数のプレフィックス情報を定義する場合、CSV 形式(カンマ区切り)で値を設定します。

プレフィックスの定義がない場合、サイズのみで判断するため、全てサイズに応じた singed 型になります。

表 5-37 サイズ毎の型

サイズ	型
1	INT8
2	INT16
4	FLOAT

【ConvData クラスについて】

ConvData クラスは、変換した各データを格納するクラスです。

表 5-38 ConvData クラスの変数

#	変数名	型	I/O	概要	有効値
1	AddressName	string	O	変数のアドレスを 8 桁の 16 進数で格納する	（値の設定は関数内で行う）
2	VariableName	string	O	変数名を格納する	（値の設定は関数内で行う）
3	DataType	byte	O	0～6 のデータ型の数値を格納する	（値の設定は関数内で行う）

表 5-39 データ型の値

値	内容
0	UINT8
1	INT8
2	UINT16
3	INT16
4	UINT32
5	INT32
6	FLOAT

Map ファイルの変換及び変数への格納処理に成功した場合、戻り値を処理成功で返します。

Map ファイルの変換及び変数への格納処理に失敗した場合、戻り値を失敗要因に対応する値で返します。

Map ファイルの変換中にエラーが発生した場合は処理を終了するため、エラー発生までの変換結果は、引数で指定した変換結果格納用の変数には格納されません。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-40 Map ファイル変換メモリ展開関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：Map ファイルが存在しない	引数の Map ファイルのパスを確認する。
3			2：Map ファイルが読み込めない	引数で指定した Map ファイルが他から開かれていないか、読み込み専用になっていないか確認する。
4			3：Map ファイルが対応していないコンパイラから出力されている	引数で指定した Map ファイルが対応したコンパイラから出力されているか確認する。
5			4：Map ファイル内の変数設定が不正	引数で指定した Map ファイルが正しいものか確認する。
6			5：Map ファイル内のアドレスが 16 進数に変換できない	引数で指定した Map ファイルが正しいものか確認する。
7			6：Map ファイル内のサイズが 16 進数に変換できない	引数で指定した Map ファイルが正しいものか確認する。
8			7：Map ファイル変換異常	引数で指定した Map ファイルが正しいものか確認する。
9			8：上限数が無効	引数で指定した上限数の設定を確認する。
10			9：格納用配列の定義が不正	引数で指定した格納用配列の定義を確認する。

5.10.2 Map ファイル変換 CSV 出力関数(ConvertMapToCSV)

本関数は、引数で指定した Map ファイルの内容を変換し、引数で指定したファイルパスに変換した CSV 形式(カンマ区切り)でファイルを保存します。

表 5-41 Map ファイル変換 CSV 出力関数の引数

#	変数名	型	I/O	有効値
1	マップファイルパス	string	I	実際に存在する Map ファイルのパス
2	変数用プレフィックス	VariablePrefixDefined	I	型毎にプレフィックス情報を設定するクラス
3	配列用プレフィックス	ArrayPrefixDefined	I	型毎にプレフィックス情報を設定するクラス
4	出力パス	string	I	禁則文字の入っていないファイルパス
5	上書きフラグ	bool	I	true：上書き許可 false：上書き禁止

※VariablePrefixDefined、ArrayPrefixDefined 構造体については、「5.10.1Map ファイル変換メモリ展開関数(ConvertMapToMemory)」を参照してください。

表 5-42 ファイルパスの禁則文字

#	使用不可文字	備考
1	¥	ファイル名、フォルダ名には使用不可。 パスの区切り文字としては入力可能。
2	/	−
3	:	ファイル名、フォルダ名には使用不可。 パスの区切り文字としては入力可能。
4	*	−
5	?	−
6	“	−
7	<>	−
8	|	−

Map ファイルの変換及び CSV ファイルの出力処理に成功した場合、戻り値を処理成功で返します。

出力パスに既にファイルが存在する場合、上書きフラグが true であれば、ファイルを上書きし、上書きフラグが false であれば、エラーを返します。

CSV は以下のフォーマットで出力します。

表 5-43 CSV のフォーマット

No	項目名	出力例	内容
1	コンパイラ固有文字列	Compiler：0	ファイルの先頭に、CC の場合 0、CA の場合 3 を出力
2	変数アドレス	00000401	変数のアドレスを 8 桁の 16 進数で出力
3	変数名	g_u1_motor_status	変数名を出力
4	データ型	0	データ型を表す数値を出力

表 5-44 データ型の値

値	内容
0	UINT8
1	INT8
2	UINT16
3	INT16
4	UINT32
5	INT32
6	FLOAT

CSV の出力例を以下に示します。

```
Compiler Type：0
Address,Variable Name,Data Type
00000401,g_u1_motor_status,0
00000402,gui_u1_active_gui,1
00000403,com_u1_sw_userif,0
00000404,g_u1_sw_userif,0
00000405,com_u1_mode_system,0
00000406,g_u1_mode_system,0
00000408,com_u1_direction,0
00000409,com_u1_enable_write,0
0000040a,g_u1_enable_write,0
00000432,g_u2_conf_sw_ver,2
00000434,g_u2_max_speed_rpm,2
00000436,com_s2_ref_speed_rpm,3
00000438,com_u2_max_speed_rpm,2
0000043a,com_u2_overspeed_limit_rpm,2
0000043c,com_u2_offset_calc_time,2
0000043e,com_u2_mtr_pp,2
00000440,com_u2_id_up_speed_rpm,2
00000442,com_u2_id_down_speed_rpm,2
00000456,g_u2_conf_hw,2
00000458,g_u2_conf_sw,2
```

Map ファイルの変換及び CSV ファイルへの出力処理に失敗した場合、戻り値を失敗要因に対応する値で返します。

Map ファイルの変換中にエラーが発生した場合、ファイルを出力せずに終了します。

エラー発生時は、戻り値を確認して、以下の対応を行ってください。

表 5-45 Map ファイル変換 CSV 出力関数の戻り値と対応

#	内容	型	戻り値	対応
1	処理の結果	int	0：処理成功	-
2			1：Map ファイルが存在しない	引数の Map ファイルのパスを確認する。
3			2：Map ファイルが読み込めない	引数で指定した Map ファイルが他から開かれていないか、読み込み専用になっていないか確認する。
4			3：Map ファイルが対応していないコンパイラから出力されている	引数で指定した Map ファイルが対応したコンパイラから出力されているか確認する。
5			4：Map ファイル内の変数設定が不正	引数で指定した Map ファイルが正しいものか確認する。
6			5：Map ファイル内のアドレスが 16 進数に変換できない	引数で指定した Map ファイルが正しいものか確認する。
7			6：Map ファイル内のサイズが 16 進数に変換できない	引数で指定した Map ファイルが正しいものか確認する。
8			7：Map ファイル変換異常	引数で指定した Map ファイルが正しいものか確認する。
9			8：出力パスが未設定	引数で指定している出力パスに値を設定する。
10			9：出力パスに無効な文字が含まれている	引数で指定した出力パスに禁則文字が使用されていないか確認する。
11			10：出力パスが出力できないパスを指定している	引数で指定した出力パスを別のパスに変更する。
12			11：ファイルが既に存在する ※上書きフラグが false で出力先にファイルが既に存在する場合発生。	ファイルの上書きを許可する、もしくは、出力先を変更する。
13			12：ファイル出力失敗	出力先に指定したファイルが出力可能なフォルダか、上書きする場合、他の機能で使用していないか確認する。

6. 各関数の使用方法

6.1 DLL 使用前の準備

6.1.1 各デバイスの接続

パソコンとツール用通信ボードを USB ケーブルで接続し、ツール用通信ボードとマイコンを専用ケーブルで接続します。

ツール用通信ボードとの接続方法については、ツール用通信ボードのマニュアルを参照してください。

6.1.2 COM ポートの確認

DLL の接続関数実行のために、引数として設定する COM ポートを確認しておく必要があります。

以下の手順で COM ポートを確認します。

スタートメニューを右クリックし、「デバイスマネージャー」を選択します。

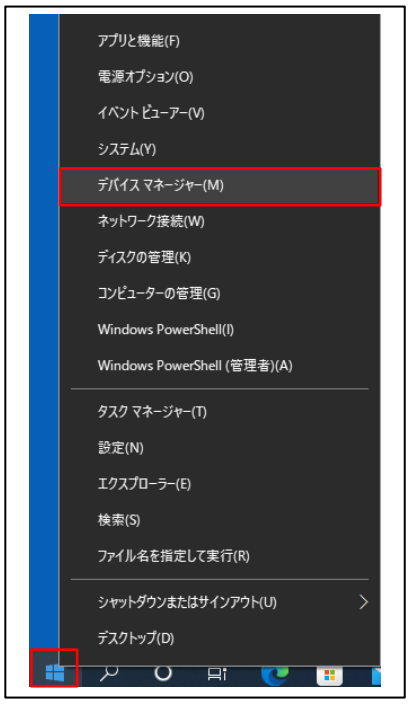


図 6-1 デバイスマネージャーの起動

「ポート(COM と LPT)」をダブルクリックし、ポート一覧を表示します。

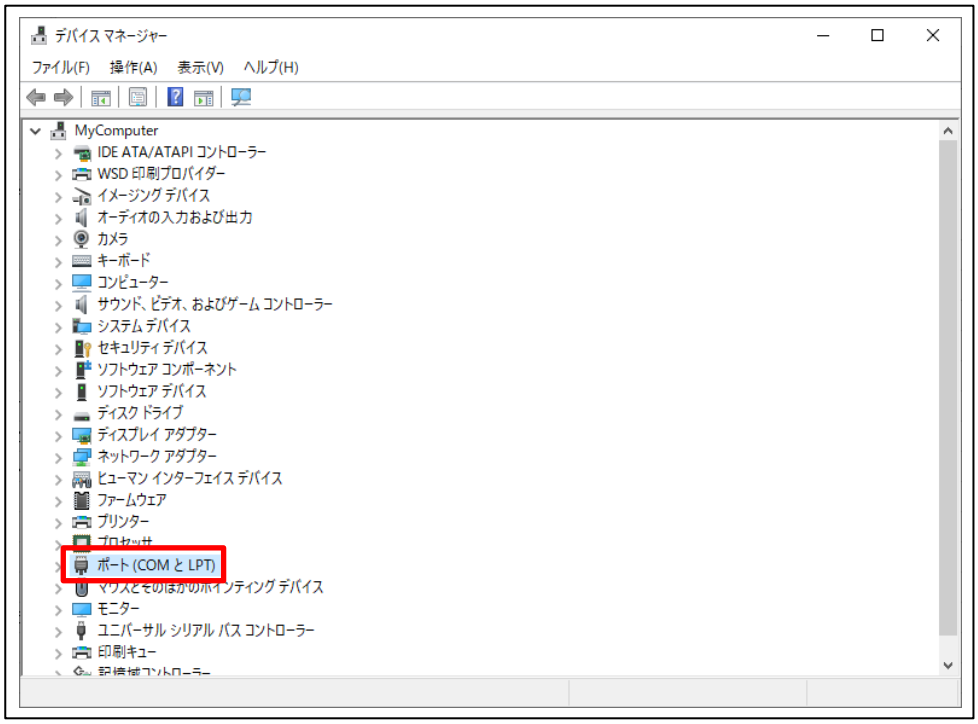


図 6-2 デバイスマネージャーの表示

「USB シリアルデバイス(COM～)」の括弧で囲まれた部分が、COM ポート番号(画像の例の場合"COM3")になります。

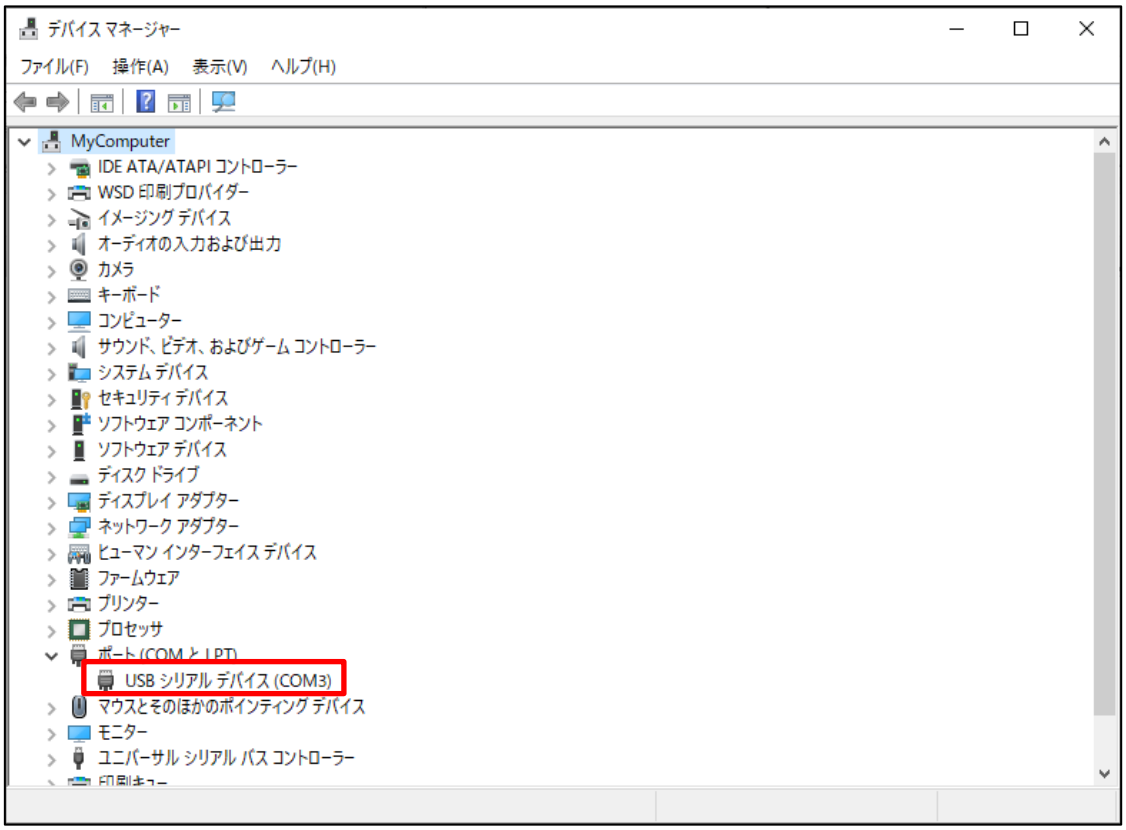


図 6-3 COM ポートの確認

6.2 Excel VBA で DLL を使用する場合の注意点

(1) Integer 型の使用について

int 型の引数が含まれる関数を Excel VBA から使用する場合は、引数として設定する値は Integer 型ではなく、Long 型で指定してください。

(2) enum 型の使用について

DLL で定義している enum 型は Excel VBA からは参照できません。

enum 型の引数が含まれる関数を Excel VBA から使用する場合は、設定したい値を Long 型で指定してください。

(3) DoEvents の使用について

Excel VBA で DoEvents 関数を使用すると、OS に制御を戻し、保留となっているイベントの処理(Excel 上での操作等)が実行されます。

以下の例のように、ループ等にて時間がかかる処理を行う場合は、処理の途中で DoEvents 関数を使用してください。

（例）

Scope 開始後データ取得可能になるまで ScopeGetCondition 関数をループ処理内で使用し、定期的に状態を確認する処理

（使用方法）

ループ処理内の ScopeGetCondition 関数実行前に DoEvents 関数を使用して、他の操作によるイベントを処理させる。

```
' Loop until the data acquisition state is completed.
Do
    ' Executes processes stored in the OS every 100 loops.
    ' Prevents DoEvents from running in large numbers.
    If count > 100 Then

        ' Return control to the OS and let it handle pending Excel operations and other events.
        DoEvents
        count = 0
    Else
        count = count + 1
    End If

    ' Execute the ScopeGetCondition function of the RMW communication library.
    condition = comLib.ScopeGetCondition()
Loop While condition = 1

' If the status of Scope is processing completion (data acquisition ready), data acquisition is performed.
If condition = 0 Then

    ' Execute the ScopeGetData function of the RMW communication library.
    getDataResult = comLib.ScopeGetData(100, 100, getDatas, 0)
End If
```

図 6-4 DoEvents の使用例

7. サンプルプログラムの使用方法

本 DLL を使用するためのサンプルプログラム「RMWCommLibClientSample.xlsm」について説明します。

7.1 概要

本サンプルプログラムは、本 DLL の各関数を VBA から実行し、各関数の処理結果を Excel 上に表示するプログラムです。

7.2 動作環境

本サンプルプログラムの動作には、以下の環境が必要です。

表 7-1 動作環境

#	項目名	値
1	OS	Windows 10 のみ
2	.Net Framework	.Net Framework 4.6.1 以上
3	Excel ファイル形式	Excel マクロ有効ブック(*.xlsm)

7.3 クイックスタート

7.3.1 サンプルプログラムの準備

Excel で本 DLL を使用する際には、DLL の登録が必要です。DLL の登録は「3.1.2Excel への導入方法」を参照してください。

また、本サンプルプログラムを使用するためには Excel のマクロ機能を有効にする必要があります。

7.3.2 サンプルプログラムの操作手順

本サンプルプログラムでマイコンとの接続から、変数値を Read するまでの手順を説明します。

(1)接続先のマイコンが通信可能な状態であることを確認してください

(2) COM ポート番号を入力して、通信ボードと接続します(COM ポート番号の確認は 6.1.2COM ポートの確認を参照してください)

(a)COM ポート番号を入力

(b)Connect ボタンをクリック

(c)成功すると Success が表示

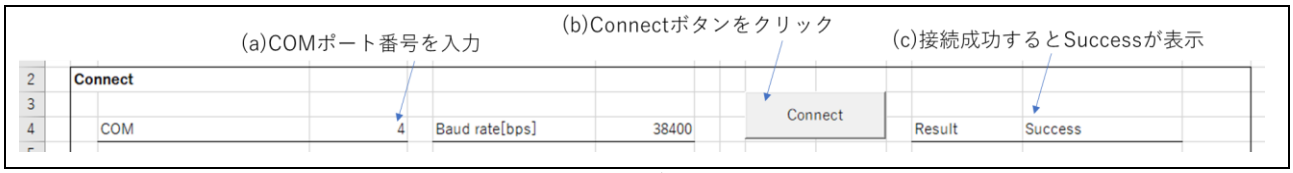


図 7-1 通信ボードと接続

(3)マイコンに書き込まれているプログラムの map ファイルを読み込みます

(a)map ファイルを選択

(b)MapToMem ボタンをクリック

(c)成功すると Success が表示

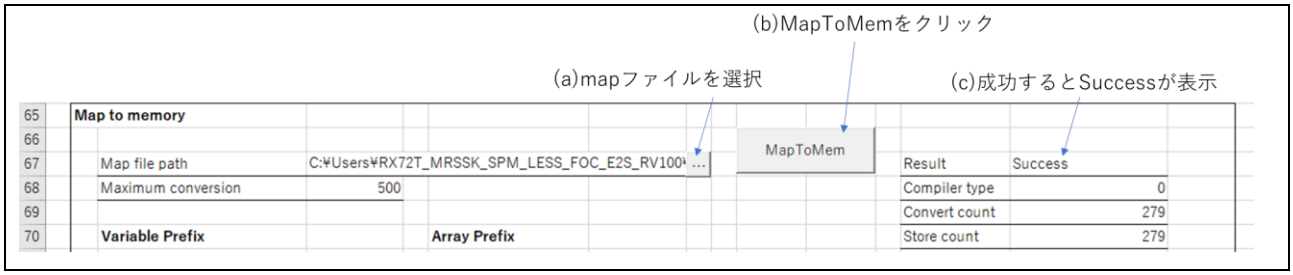


図 7-2 map ファイル読み込み

(4)変数を選択し、選択した変数の値を読み込みます

(a)Read する変数を選択

(b)Read ボタンをクリック

(c)成功すると Success が表示

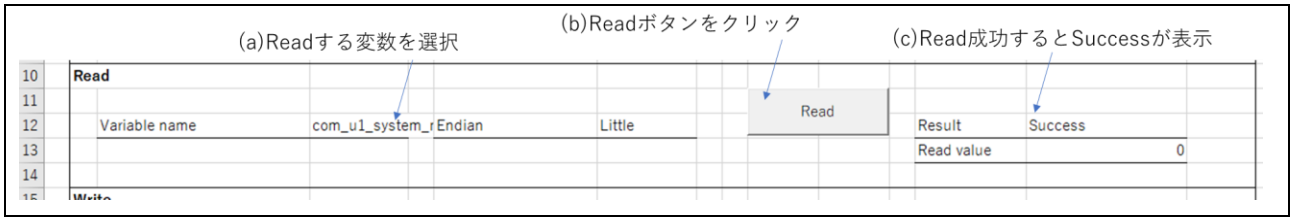


図 7-3 変数値の Read

7.4 動作説明

7.4.1 sample シート

sample シートは、本サンプルプログラムの各機能を実行するボタンを配置したメインシートです。

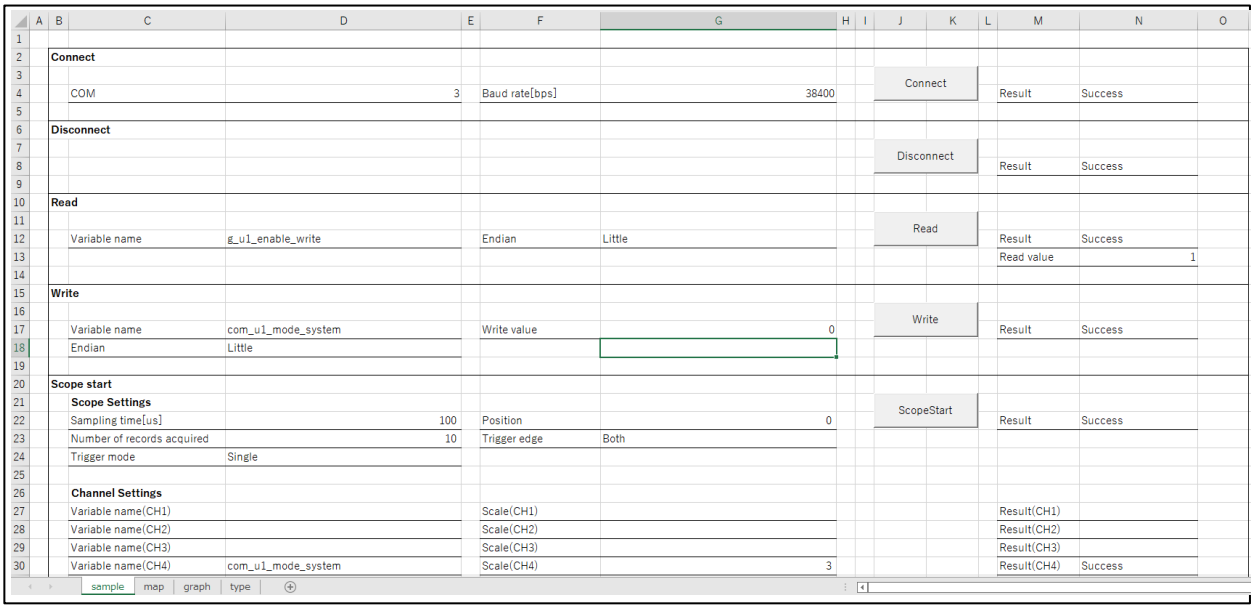


図 7-4 sample シート(全体)

各機能は罫線によって区切られており、各機能ボタンの左側に入力、右側に出力セルがまとめて配置されています。

図 7-5 sample シート(Raad 機能)

入力情報が不正の場合、以下のような表示となります(情報不正のセルが赤色になり、エラーメッセージを表示)ので、入力情報の見直しをしてください。

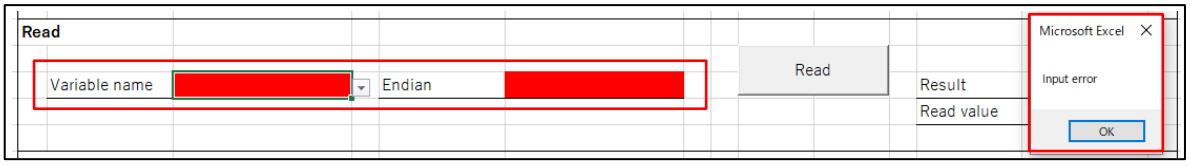


図 7-6 入力情報が不正

Read 機能や Write 機能等の、マイコンのデータを読み込み／書き込みする機能では、各セルでの変数入力やアドレス変換のために Map ファイル情報が必要です。

「Map to memory 機能」により、Map ファイル情報を読み込んだ後、各機能を使用してください。

また、Excel を閉じる操作を行うと、Sheet_Finalization 関数にて Scope 処理の停止及びマイコンとの切断を行います。閉じる操作をキャンセルした場合には、再度マイコンとの接続から実施してください。

(1) Connect 機能

以下の表に示す各セルに、必要な情報を入力し、Connect ボタンを押下することで、ConnectButton_Click 関数を実行し、マイコンと接続します。

表 7-2 Connect 機能の入力内容

#	入力セル	入力内容
1	COM	接続する COM ポート番号
2	Baud rare[bps]	ボーレート値

以下の表に示す各セルに、DLL の Connect 関数の処理結果を出力します。

表 7-3 Connect 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 IllegalCOMPort：COM ポート不正 IllegalBaudrate：ボーレート不正 ConnectFailed：接続失敗 SendCommendFailed：コマンドの応答が不正

(2) Disconnect 機能

Disconnect ボタンを押下することで、DisconnectButton_Click 関数を実行し、接続されているマイコンと切断します。

以下の表に示す各セルに、DLL の DisConnect 関数の処理結果を出力します。

表 7-4 Disconnect 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 NotConnect：接続されていない

(3) Read 機能

以下の表に示す各セルに、必要な情報を入力し、Read ボタンを押下することで、ReadButton_Click 関数を実行し、接続されているマイコンのデータを読み込みます。

表 7-5 Read 機能の入力内容

#	入力セル	入力内容
1	Variable name	読み込む変数
2	Endian	Little：リトルエンディアンで読み込み Big：ビッグエンディアンで読み込み

以下の表に示す各セルに、DLL の Read 関数の処理結果を出力します。

表 7-6 Read 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 NotConnect：マイコンと接続されていない CommunicationFailed：マイコンとの通信に失敗 IllegalAddress：アドレスに指定された値が不正 IllegalDataType：データ型が不正 IllegalEndianType：Endian が不正
2	Read value	読み込んだデータ

(4) Write 機能

以下の表に示す各セルに、必要な情報を入力し、Write ボタンを押下することで、WriteButton_Click 関数を実行し、接続されているマイコンにデータを書き込みます。

表 7-7 Write 機能の入力内容

#	入力セル	入力内容
1	Variable name	書き込む変数
2	Write value	書き込むデータ
3	Endian	Little：リトルエンディアンで書き込み Big：ビッグエンディアンで書き込み

以下の表に示す各セルに、DLL の Write 関数の処理結果を出力します。

表 7-8 Write 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 NotConnect：マイコンと接続されていない IllegalAddress：アドレスが不正 IllegalValue：書き込み値が範囲外 CommunicationFailed：マイコンとの通信失敗 IllegalDataType：データ型が不正 IllegalEndianType：Endian が不正

(5) Scope start 機能

以下の表に示す各セルに、必要な情報を入力し、ScopeStart ボタンを押下することで、ScopeStartButton_Click 関数が実行され、SetProccess 関数、AddChannels 関数、StartScope 関数を順に実行します。ただし、各関数の処理に失敗した場合は、その時点で処理が終了します。

表 7-9 Scope start 機能(Scope Settings)の入力内容

#	入力セル	入力内容
1	Sampling time[us]	サンプリング時間（μs）
2	Position	トリガを判定する位置
3	Number of records acquired	取得するレコード数
4	Trigger edge	Rise：上がり検知 Fall：下り検知 Both：両検知
5	Trigger mode	Single：1 回取得 Normal：トリガ条件を満たす度に取得 Auto：常時取得

表 7-10 Scope start 機能(Channel Settings)の入力内容

#	入力セル	入力内容
1	Variable name(CHx)	CHx で読み込む変数
2	Scale(CHx)	CHx のスケール ※トリガレベルにスケールを乗算した値がトリガレベルとしてマイコンに反映される

表 **7-11 Scope start 機能(Scope Start)**の入力内容

#	入力セル	入力内容
1	Trigger channel	トリガとなるチャネル番号
2	Trigger level	トリガとなる値

SetProccess 関数は、以下の表に示す各セルに、DLL の Scope 設定関数の処理結果を出力します。

表 7-12 Scope start 機能(Scope Settings)の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 IllegalTriggerEdge：トリガエッジが不正 IllegalTriggerMode：トリガモードが不正

AddChannels 関数は、以下の表に示す各セルに、DLL の Scope チャネル追加関数の処理結果を出力します。

表 7-13 Scope start 機能(Channel Settings)の出力内容

#	出力セル	出力内容
1	Result(CHx)	Success：処理成功 IllegalDataType：データの種類が不正 DisableDataType：ボードが対応していないデータ型

StartScope 関数は、以下の表に示す各セルに、DLL の Scope 開始関数の処理結果を出力します。

表 7-14 Scope start 機能(Scope Start)の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 IllegalScopeSetting：スコープ設定が不正 IllegalTriggerLevel：トリガレベルが不正 IllegalChannelSetting：チャネル設定が不正 IllegalPositionSetting：Position の設定が不正 IllegalRecordCount：レコード長の設定が不正 IllegalSamplingTime：サンプリング周期が不正 NotConnect：マイコンと接続されていない SendCommandFailed：コマンド通信失敗

(6) Scope get condition 機能

ScopeGetCondition ボタンを押下することで、ScopeGetConditionButton_Click 関数を実行し、Scope 処理の状態を取得します。

以下の表に示す各セルに、DLL の Scope 状態取得関数の処理結果を出力します。

表 7-15 Scope get condition 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 PreparingToAcquireData：データ取得準備中 NotConnect：マイコンと接続されていない SendCommandFailed：コマンド通信失敗

Scope 処理が開始状態のときに、処理結果が処理成功になると、Scope get data 機能によるデータの取得が可能になります。

(7) Scope get data 機能

以下の表に示す各セルに、必要な情報を入力し、ScopeGetData ボタンを押下することで、ScopeGetDataButton_Click 関数を実行し、Scope 処理において収集したデータを取得します。

本機能は、Scope get condition 機能が処理成功となった場合のみ実行してください。

表 7-16 Scope get data 機能の入力内容

#	入力セル	入力内容
1	Number of data acquired	取得するデータ数
2	Endian	Little：リトルエンディアンで読み込み Big：ビッグエンディアンで読み込み

以下の表に示す各セルに、DLL の Scope データ取得関数の処理結果を出力します。

また、データ取得完了すると、graph シートに取得したデータ出力とグラフ描画を行います。

表 7-17 Scope get data 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功（データ取得完了） IncorrectReadRange：読み込み範囲の設定が不正 IllegalGetData：取得データ格納配列の設定が不正 DataAcquisitionNotPossible：状態がデータ取得可能ではない NotConnect：マイコンと接続されていない IllegalEndianType：エンディアン設定が不正 ScopeSettingNothing：Scope 設定が未設定 SendCommandFailed：コマンド通信失敗

(8) Scope stop 機能

ScopeStop ボタンを押下することで、ScopeStopButton_Click 関数を実行し、開始した Scope 処理を停止します。

以下の表に示す各セルに、DLL の Scope 停止関数の処理結果を出力します。

表 7-18 Scope stop 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 NotConnect：マイコンと接続されていない SendCommandFailed：コマンド通信失敗

(9) Map to csv 機能

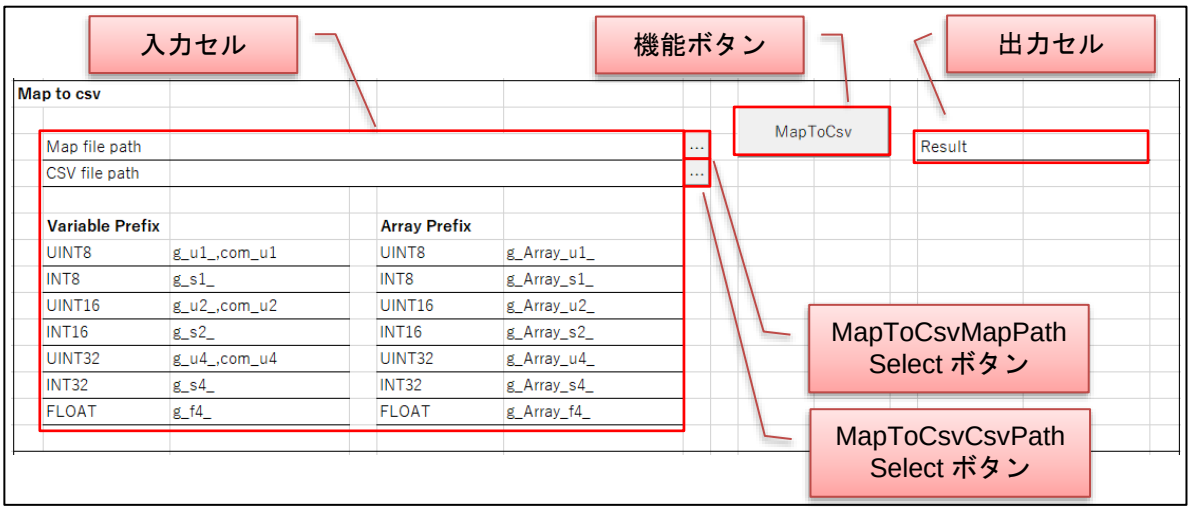


図 7-7 Map to csv 機能

以下の表に示す各セルに、必要な情報を入力し、MapToCsv ボタンを押下することで、MapToCsvButton_Click 関数を実行し、Map ファイルの内容を CSV 形式(カンマ区切り)に変換、指定したパスにファイルを保存します。

表 7-19 Map to csv 機能の入力内容

#	入力セル	入力内容
1	Map file path	実際に存在する Map ファイルのパス
2	CSV file path	禁則文字の入っていない出力ファイルパス ※すでにファイルが存在する場合は上書き
3	Variable Prefix	各データ型における Variable Prefix 情報
4	Array Prefix	各データ型における Array Prefix 情報

以下の表に示す各セルに、DLL の Map ファイル変換 CSV 出力関数の処理結果を出力します。

表 7-20 Map to csv 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 MapFileNotFound：Map ファイルが存在しない MapFileCanNotRead：Map ファイルが読み込めない MapFileIsOutputFromAnUnsupportedCompiler：Map ファイルが対応していないコンパイラから出力されている IncorrectVariableSettingsInTheMapFile：Map ファイル内の変数設定が不正 AddressesCannotBeConvertedToHexadecimal：Map ファイル内のアドレスが 16 進数に変換できない SizeCannotBeConvertedToHexadecimal：Map ファイル内のサイズが 16 進数に変換できない MapFileConversionError：Map ファイル変換異常 OutputPathIsNotSet：出力パスが未設定 OutputPathContainsInvalidCharacters：出力パスに無効な文字が含まれている OutputPathIsAPathThatCannotBeOutput：出力パスが出力できないパスを指定している FileAlreadyExists：ファイルが既に存在する FileOutputFailure：ファイル出力失敗

(a) Map to csv 機能用 Map ファイル選択機能

MapToCsvMapPathSelect ボタンを押下することで、MapToCsvMapPathSelectButton_Click 関数を実行し、ファイル選択ダイアログを開きます。開いたダイアログから Map ファイルを選択することで、CSV ファイルに変換する Map ファイルを選択します。

以下の表に示すセルに、選択した Map ファイルのパスを出力します。

表 7-21 Map to csv 機能用 Map ファイル選択機能の出力内容

#	出力セル	出力内容
1	Map file path	CSV ファイルに変換する Map ファイル

(b) Map to csv 機能用 Csv ファイル出力先指定機能

MapToCsvCsvPathSelect ボタンを押下することで、MapToCsvCsvPathSelectButton_Click 関数を実行し、パス指定ダイアログが開きます。開いたダイアログから、変換した CSV ファイルを出力するパスを指定します。

以下の表に示すセルに、指定したパスを出力します。

表 7-22 Map to csv 機能用 Csv ファイル出力先指定機能の出力内容

#	出力セル	出力内容
1	CSV file path	変換した CSV ファイルを出力するパス

(10) Map to memory 機能

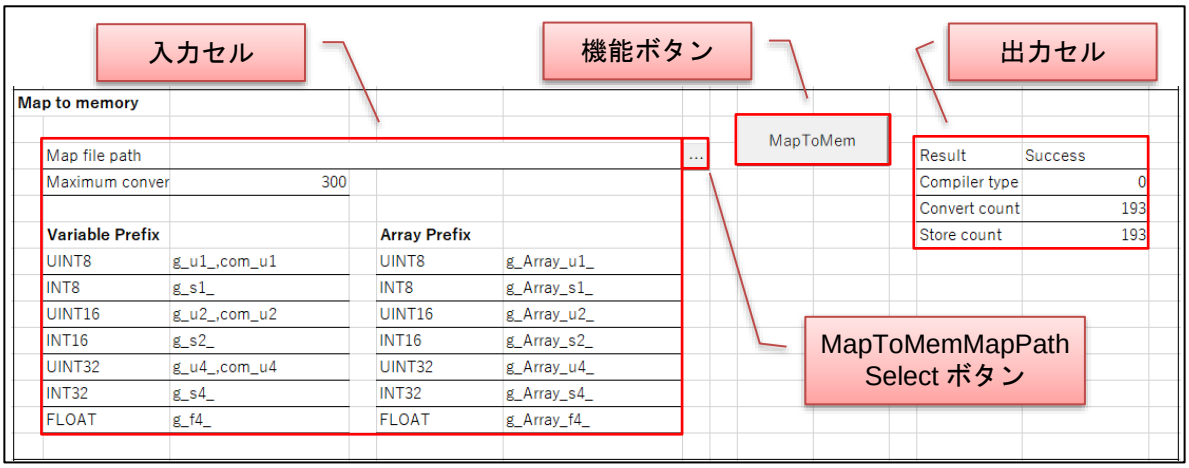


図 7-8 Map to memory 機能

以下の表に示す各セルに、必要な情報を入力し、MapToMem ボタンを押下することで、MapToMemButton_Click 関数を実行し、Map ファイルの内容を map シートに展開します。

表 7-23 Map to memory 機能の入力内容

#	入力セル	入力内容
1	Map file path	実際に存在する Map ファイルのパス
2	Maximum conversion	変換上限数
3	Variable Prefix	各データ型における Variable Prefix 情報
4	Array Prefix	各データ型における Array Prefix 情報

以下の表に示す各セルに、DLL の Map ファイル変換メモリ展開関数の処理結果を出力します。

表 7-24 Map to memory 機能の出力内容

#	出力セル	出力内容
1	Result	Success：処理成功 MapFileNotFound：Map ファイルが存在しない MapFileCanNotRead：Map ファイルが読み込めない MapFileIsOutputFromAnUnsupportedCompiler：Map ファイルが対応していないコンパイラから出力されている IncorrectVariableSettingsInTheMapFile：Map ファイル内の変数設定が不正 AddressesCannotBeConvertedToHexadecimal：Map ファイル内のアドレスが 16 進数に変換できない SizeCannotBeConvertedToHexadecimal：Map ファイル内のサイズが 16 進数に変換できない MapFileConversionError：Map ファイル変換異常 InvaldDataLimitNum：上限数が無効 InvalidStorageArrayForConversionResult：格納用配列の定義が不正
2	Compiler type	コンパイラ種別
3	Convert count	変換数
4	Store count	格納数
5	map シート	メモリに展開された Map ファイル情報 A 列：変数 B 列：アドレス C 列：データ型

(a) Map to memory 機能用 Map ファイル選択機能

MapToMemMapPathSelect ボタンを押下することで、MapToMemMapPathSelectButton_Click 関数を実行し、ファイル選択ダイアログを開きます。開いたダイアログから Map ファイルを選択することで、CSV ファイルに変換する Map ファイルを選択します。

以下の表に示すセルに、選択した Map ファイルのパスを出力します。

表 7-25 Map to memory 機能用 Map ファイル選択機能出力内容

#	出力セル	出力内容
1	Map file path	map シートに展開する Map ファイル

(11) Auto read 機能

以下の表に示す各セルに、必要な情報を入力し、AutoReadStart ボタンを押下することで、AutoReadButton_Click 関数を実行し、接続されているマイコンのデータを定期的に読み込みます。

表 7-26 Auto read 機能の入力内容

#	入力セル	入力内容
1	Variable name	読み込む変数
2	Endian	Little：リトルエンディアンで読み込み Big：ビッグエンディアンで読み込み
3	Sampling time[ms]	読み込み周期

以下の表に示す各セルに、Read 関数の処理結果を出力します。

出力内容は、Read 関数が実行されるたびに、上書きされます。

※セルの編集中は、出力内容の更新は行われません。

表 7-27 Auto read 機能の出力内容

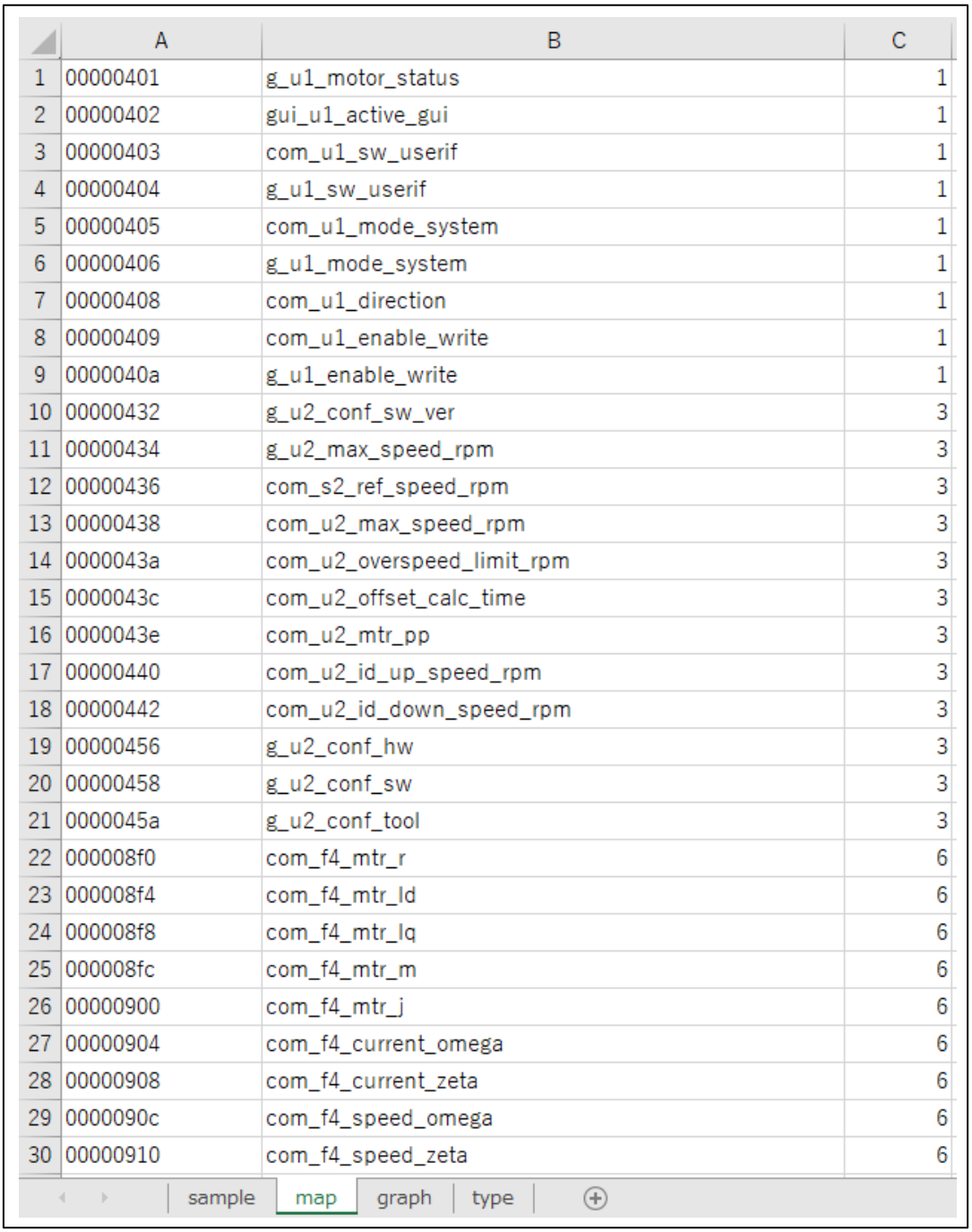
#	出力セル	出力内容
1	Result	Success：処理成功 NotConnect：マイコンと接続されていない CommunicationFailed：マイコンとの通信に失敗 IllegalAddress：アドレスに指定された値が不正 IllegalDataType：データ型が不正 IllegalEndianType：Endian が不正
2	Read value	読み込んだデータ

7.4.2 map シート

map シートは、Map to memory 機能によりメモリに展開した Map ファイル情報を表示します。

本シートに展開された変数は、sample シートの変数を入力するセルのドロップダウンリストに登録されます。

また、本シートは、変数の読み込み及び書き込みを行う際の、アドレスやデータ型の参照先として使用されます。



	A	B	C
1	00000401	g_u1_motor_status	1
2	00000402	gui_u1_active_gui	1
3	00000403	com_u1_sw_userif	1
4	00000404	g_u1_sw_userif	1
5	00000405	com_u1_mode_system	1
6	00000406	g_u1_mode_system	1
7	00000408	com_u1_direction	1
8	00000409	com_u1_enable_write	1
9	0000040a	g_u1_enable_write	1
10	00000432	g_u2_conf_sw_ver	3
11	00000434	g_u2_max_speed_rpm	3
12	00000436	com_s2_ref_speed_rpm	3
13	00000438	com_u2_max_speed_rpm	3
14	0000043a	com_u2_overspeed_limit_rpm	3
15	0000043c	com_u2_offset_calc_time	3
16	0000043e	com_u2_mtr_pp	3
17	00000440	com_u2_id_up_speed_rpm	3
18	00000442	com_u2_id_down_speed_rpm	3
19	00000456	g_u2_conf_hw	3
20	00000458	g_u2_conf_sw	3
21	0000045a	g_u2_conf_tool	3
22	000008f0	com_f4_mtr_r	6
23	000008f4	com_f4_mtr_ld	6
24	000008f8	com_f4_mtr_lq	6
25	000008fc	com_f4_mtr_m	6
26	00000900	com_f4_mtr_j	6
27	00000904	com_f4_current_omega	6
28	00000908	com_f4_current_zeta	6
29	0000090c	com_f4_speed_omega	6
30	00000910	com_f4_speed_zeta	6

図 7-9 map シート

7.4.3 graph シート

graph シートは、scope 機能を使用して取得したデータ及びグラフ作成用に加工したデータを表示します。

また、グラフ作成用データから作成したグラフを表示します。

※以下の図の赤枠で囲まれた領域に取得したデータ、青枠で囲まれた領域にグラフ作成用データを表示します。

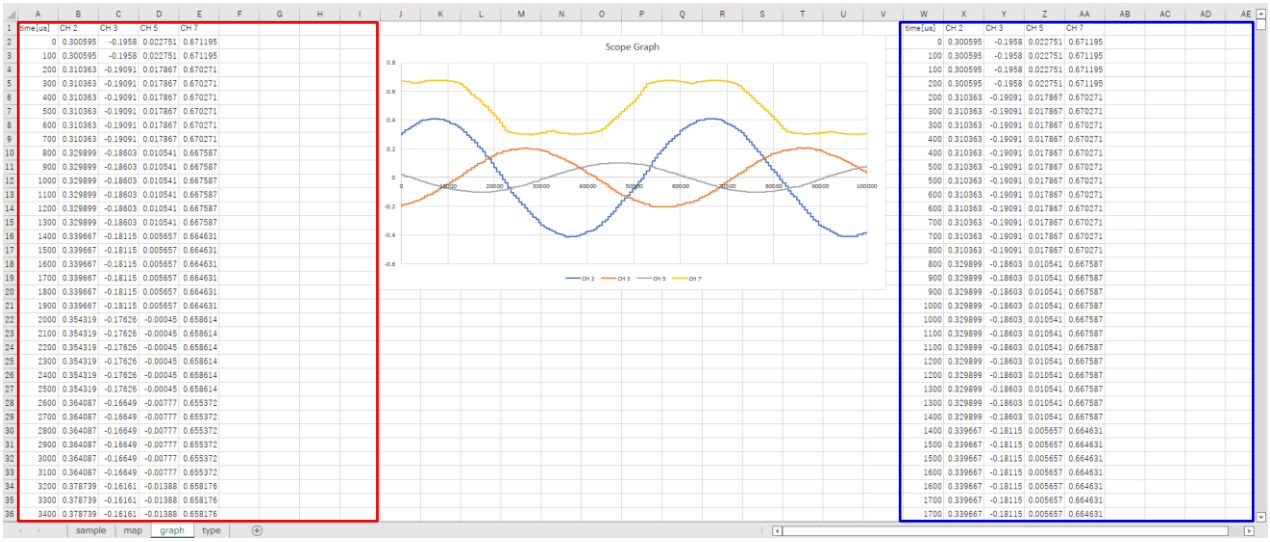


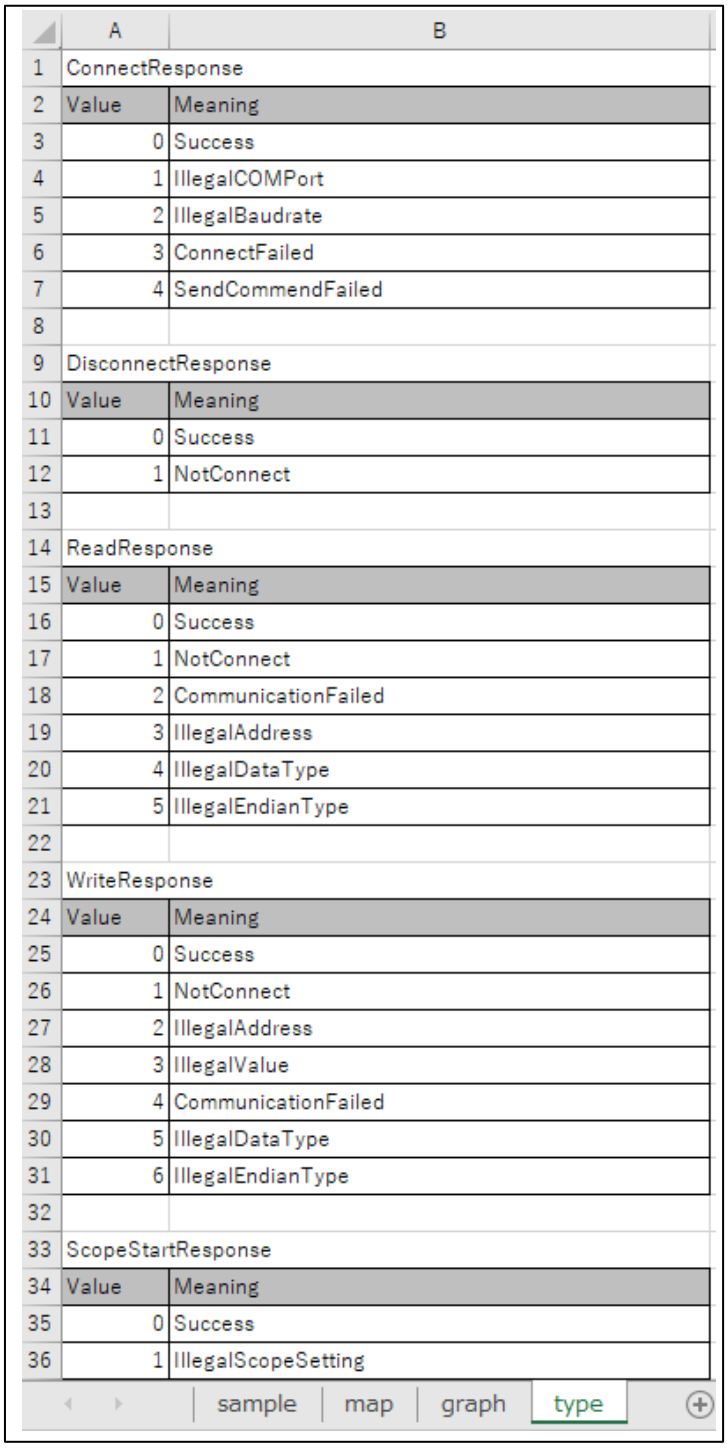
図 7-10 graph シート

7.4.4 type シート

type シートは、ライブラリ側で定義されている enum 型の文字列(列挙子)と値の対応を纏めたシートです。

VBA からは、各関数の入出力値としてライブラリ側で定義されている enum 情報が参照できません。

そのため、各関数の入出力値は数値として扱う必要があるため、本シートの情報を使用して、入力値への変換及び処理結果の意味への変換を行います。



	A	B
1	ConnectResponse	
2	Value	Meaning
3	0	Success
4	1	IllegalCOMPort
5	2	IllegalBaudrate
6	3	ConnectFailed
7	4	SendCommendFailed
8		
9	DisconnectResponse	
10	Value	Meaning
11	0	Success
12	1	NotConnect
13		
14	ReadResponse	
15	Value	Meaning
16	0	Success
17	1	NotConnect
18	2	CommunicationFailed
19	3	IllegalAddress
20	4	IllegalDataType
21	5	IllegalEndianType
22		
23	WriteResponse	
24	Value	Meaning
25	0	Success
26	1	NotConnect
27	2	IllegalAddress
28	3	IllegalValue
29	4	CommunicationFailed
30	5	IllegalDataType
31	6	IllegalEndianType
32		
33	ScopeStartResponse	
34	Value	Meaning
35	0	Success
36	1	IllegalScopeSetting

図 7-11 type シート

7.5 カスタマイズ例

本サンプルプログラムのカスタマイズ例として、Read 機能の複製方法について説明します。

7.5.1 Read 機能の複製方法

Read 機能を複製する手順について説明します。

表 7-28 Read 機能の複製の手順

#	手順内容
1	デザインモード起動
2	画面のコピー
3	セルの名前を定義
4	ボタンの名称を定義
5	新しいボタンのイベントを作成
6	イベント内の処理をコピー
7	入出力セルの名前を、定義した名前に変更
8	デザインモード終了

(1) デザインモード起動

Excel 上部の開発タブの「デザインモード」ボタン押下によりデザインモードに切り替えます。

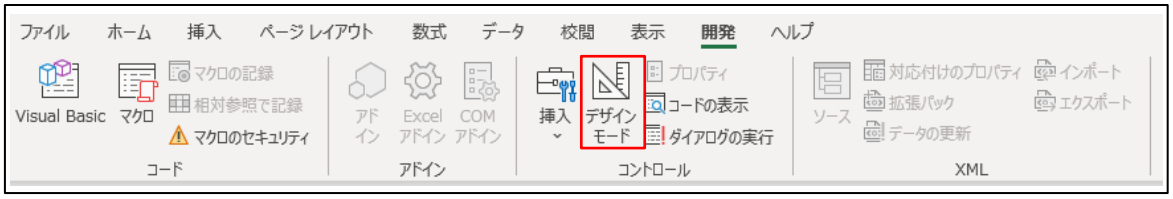


図 7-12 デザインモード起動方法

(2) 画面のコピー

Sample シートの Read 機能に関係する箇所をコピーし、任意の位置に張り付けます。

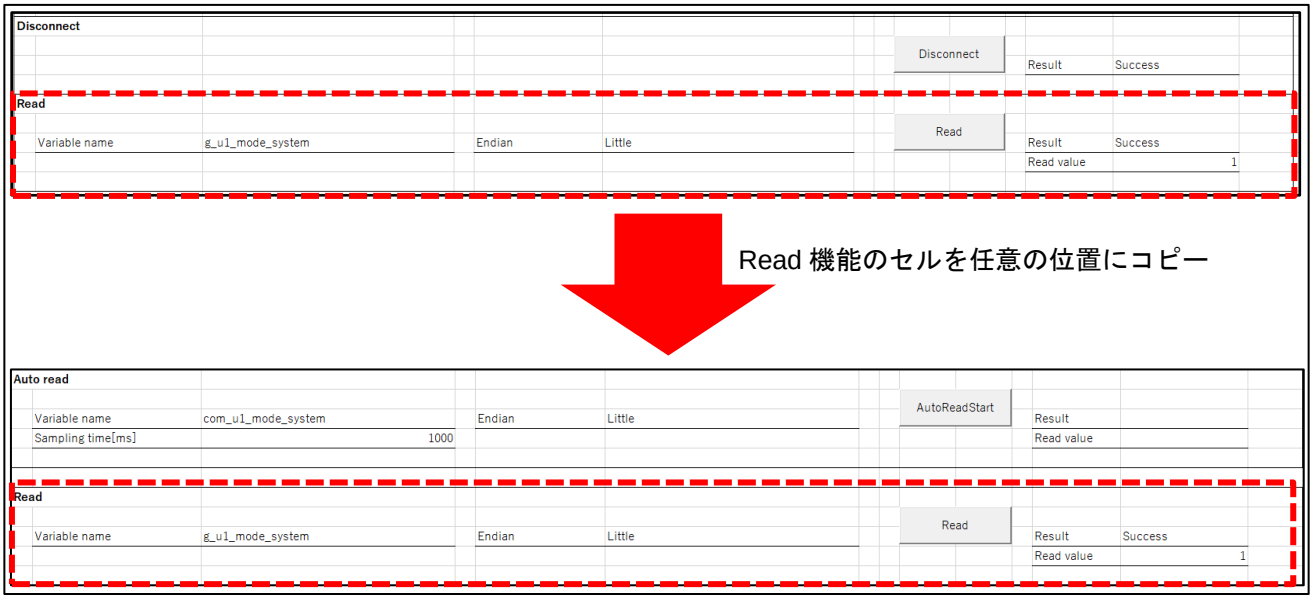


図 7-13 画面コピーの例

(3) セルの名前を定義

各機能の処理において、入力情報を取得するセルや処理結果を出力するセルは、定義されたセルの名前により指定しています。セルをコピーした場合、セルの名前はコピーされません。そのため、張り付け先の入出力セルに、名前の定義を追加します。

セルの名前は、セルを右クリックし、メニューから「名前の定義」を選択することで開く「新しい名前」画面から定義します。

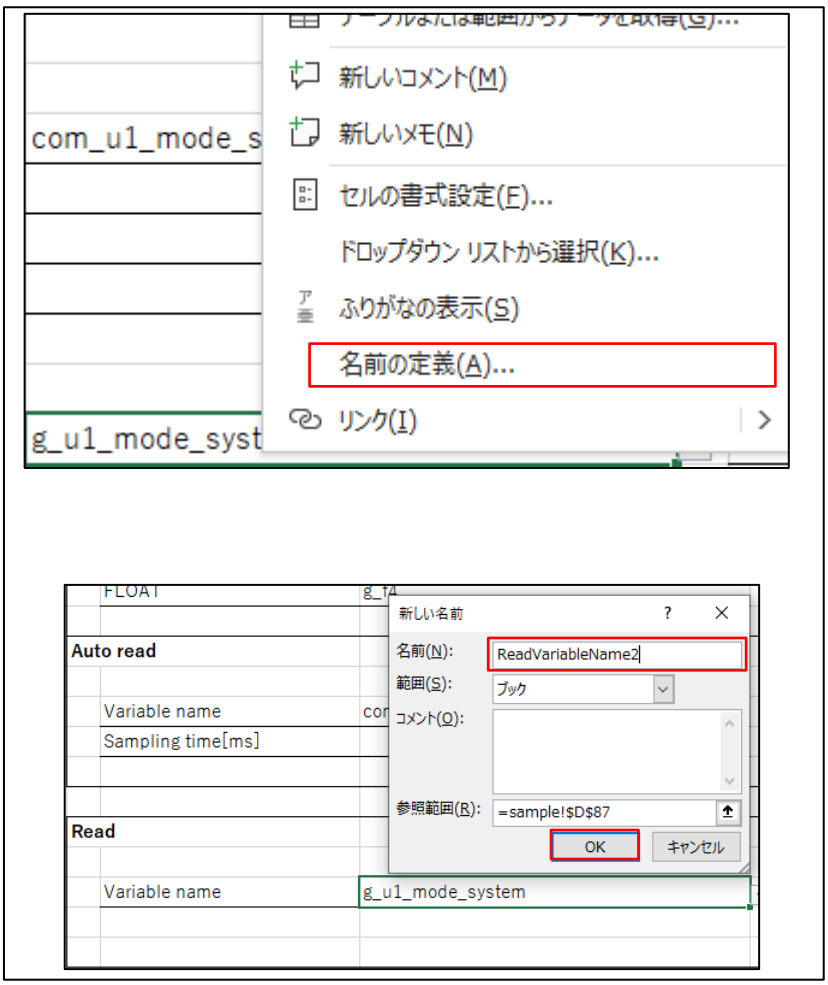


図 7-14 セルの名前の定義方法

(4) ボタンの名称を定義

「(2)画面のコピー」で張り付けた Read ボタンの名称を変更します。

Read ボタン上で右クリックして、表示されたメニューからプロパティを選択します。

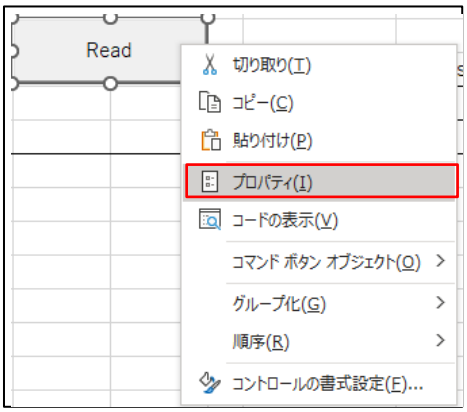


図 7-15 オブジェクトプロパティ画面の表示方法

プロパティ画面のオブジェクト名を任意の名前に変更します。

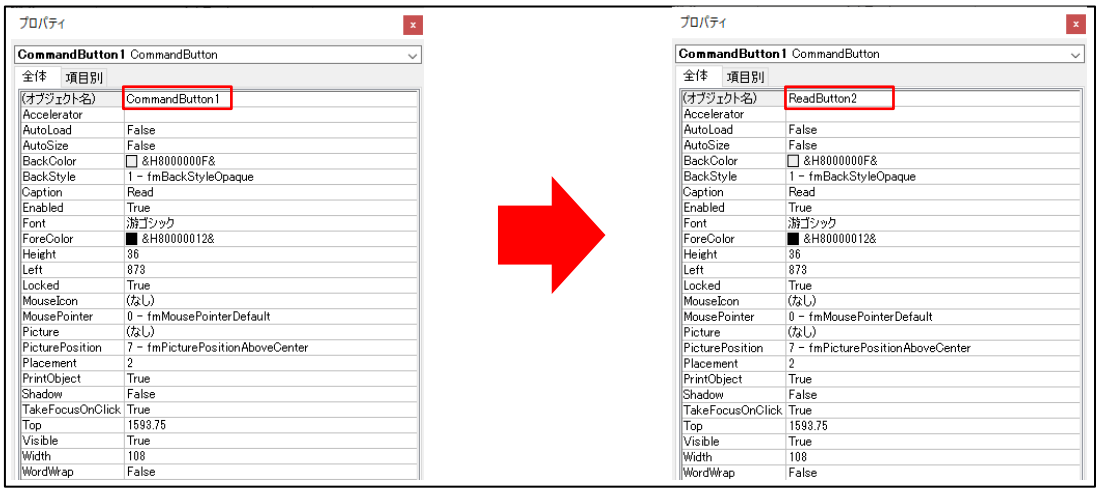


図 7-16 オブジェクト名の変更方法

(5) 新しいボタンのイベントを作成

「(4)ボタンの名称を定義」で名称を変更したボタンの上でダブルクリックすることで、そのボタンのクリックイベントが SampleSheet モジュールに追加され、追加されたイベントがボタンに登録されます。

イベント名は「(オブジェクト名)_Click」で登録されます。

```
Private Sub ReadButton2_Click()
End Sub
```

図 7-17 新しいボタンのイベントを作成する方法

(6) イベント内の処理をコピー

「(5) 新しいボタンのイベントを作成」で登録されたイベントに、ReadButton_Click イベントの処理をコピーします。

```
Private Sub ReadButton2_Click()
    Dim result As Long
    Dim variableName As String
    Dim readData As String
    Dim endianType As Byte
    Dim address As Long
    Dim dataType As Long

    ' Initialize the result display cell.
    Range("ReadResult") = ""
    Range("ReadValue") = ""

    ' Input validation
    If VaridationValues(READ_CELL_NAME_LIST, "Input error") <> VALIDATION_VALUES_SUCCESS_VALUE Then
        Exit Sub
    End If

    ' Initialize variables.
    readData = ""

    ' Retrieve the input contents.
    variableName = Range("ReadVariableName")
    endianType = ConvertTypeToValue(Range("ReadEndian"), "EEndianType")

    ' Obtain the address and data type from the variable name.
    ' If the acquisition is successful, read processing is performed.
    If GetAddress(variableName, address, dataType) = GET_ADDRESS_SUCCESS_VALUE Then
        ' Execute the Read function of the RMW communication library.
        result = comLib.Read(readData, address, dataType, endianType)
        ' Convert and display processing results.
        Range("ReadResult") _
            = ConvertResultValueToMean(result, "ReadResponse")

        ' If the read process is successful, the read value is displayed.
        If result = LIBRARY_PROCESS_SUCCESS_VALUE Then
            Range("ReadValue") = readData
        End If
    End If
End Sub
```

図 7-18 イベント内の処理をコピー

(7) 入力情報チェック用のリスト情報コピー

Common モジュールの入力情報チェック用のリスト情報をコピーし、リスト情報の名称及び「(3)セルの名前を定義」で定義したセル名に変更します。

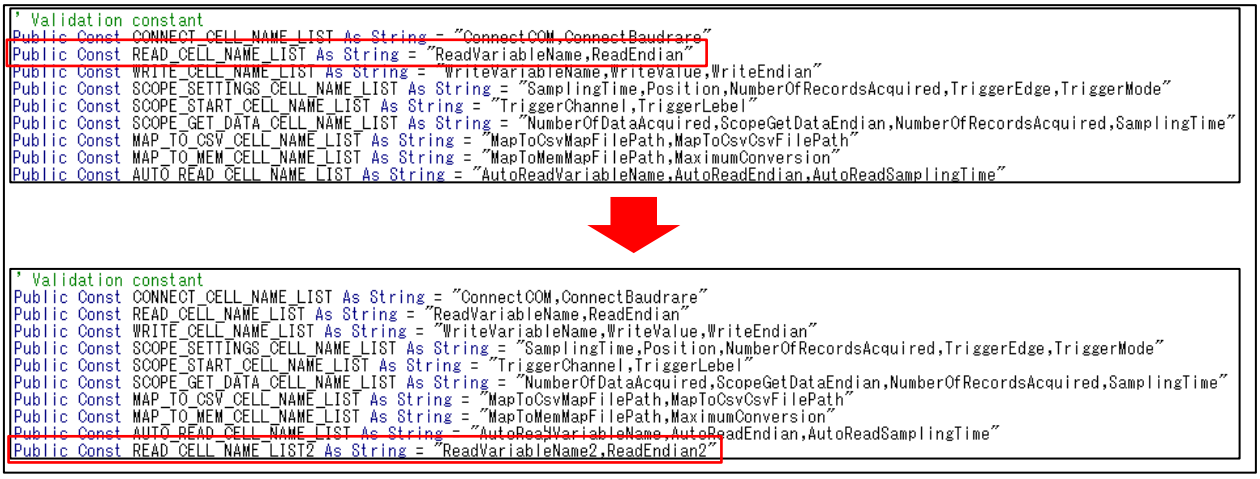


図 7-19 入力情報チェック用のリスト情報をコピー

(8) 入出力セルの名前を定義した名前に変更

「(6)イベント内の処理をコピー」でコピーしてきた処理内でセルを参照している箇所のセル名を、「(3)セルの名前を定義」で定義したセル名に変更します。

また、入力情報チェック用のリスト情報を参照している箇所の名称を「(7)入力情報チェック用のリスト情報コピー」で追加した名称に変更します。

※以下の図の赤枠で囲まれた箇所がセルを参照している箇所、青枠で囲まれた箇所が入力情報チェック用リスト情報を参照している箇所です。

```
Private Sub ReadButton2_Click()
    Dim result As Long
    Dim variableName As String
    Dim readData As String
    Dim endianType As Byte
    Dim address As Long
    Dim dataType As Long

    ' Initialize the result display cell.
    Range("ReadResult2") = ""
    Range("ReadValue2") = ""

    ' Input validation
    If ValidationValues(READ_CELL_NAME_LIST2, "Input error") <> VALIDATION_VALUES_SUCCESS_VALUE Then
        Exit Sub
    End If

    ' Initialize variables.
    readData = ""

    ' Retrieve the input contents.
    variableName = Range("ReadVariableName2")
    endianType = ConvertTypeToValue(Range("ReadEndian2"), "EEndianType")

    ' Obtain the address and data type from the variable name.
    ' If the acquisition is successful, read processing is performed.
    If GetAddress(variableName, address, dataType) = GET_ADDRESS_SUCCESS_VALUE Then
        ' Execute the Read function of the RMW communication library.
        result = comLib.Read(readData, address, dataType, endianType)
        ' Convert and display processing results.
        Range("ReadResult2") _
            = ConvertResultValueToMean(result, "ReadResponse")

        ' If the read process is successful, the read value is displayed.
        If result = LIBRARY_PROCESS_SUCCESS_VALUE Then
            Range("ReadValue2") = readData
        End If
    End If
End Sub
```

図 7-20 イベント内処理の変更箇所

(9) デザインモード終了

Excel 上部の開発タブの「デザインモード」ボタン押下によりデザインモードを終了します。

図 7-21 デザインモードの終了方法

7.6 サンプルプログラムのモジュール構成

サンプルプログラムの各モジュールに実装されている関数一覧を以下に示します。

表 7-29 サンプルプログラムのモジュール構成

モジュール名	関数名	概要	引数	戻り値
ThisWorkbook	Workbook_Open	ブックが開かれた際に発生するイベント。 SampleSheet の Sheet_Initialization 関数を実行する。	なし	なし
	Workbook_BeforeClose	ブックが閉じる前に発生するイベント。 SampleSheet の Sheet_Finalization 関数を実行する。	第 1 引数：True を設定することで閉じる操作を停止する変数	なし
SampleSheet	Sheet_Initialization	SampleSheet モジュールの初期化を行う。 AutoRead ボタンを初期化する。	なし	なし
	Sheet_Finalization	SampleSheet モジュールの終了処理を行う。 Scope 処理の停止及びマイコンと切断する。	なし	なし
	ConnectButton_Click	Connect ボタン押下時のイベント。 入力情報をセルから取得し、DLL の Connect 関数を実行し、結果をセル上に出力する。	なし	なし
	DisconnectButton_Click	Disconnect ボタン押下時のイベント。 DLL の Disconnect 関数を実行し、結果をセル上に出力する。	なし	なし
	ReadButton_Click	Read ボタン押下時のイベント。 入力情報をセルから取得し、DLL の Read 関数を実行し、結果をセル上に出力する。	なし	なし
	WriteButton_Click	Write ボタン押下時のイベント。 入力情報をセルから取得し、DLL の Write 関数を実行し、結果をセル上に出力する。	なし	なし
	ScopeStartButton_Click	ScopeStart ボタン押下時のイベント。	なし	なし

		SetProcess 関数と、AddChannels 関数と StartScope 関数を実行する。		
	SetProccess	入力情報をセルから取得し、DLL の SetProcessInfo 関数を実行し、結果をセル上に出力する。	第 1 引数：スコープ設定	処理結果
	AddChannels	各チャネルの入力情報をセルから取得し、チャネル数分 DLL の AddChannelInfo 関数を実行し、結果をセル上に出力する。	第 1 引数：スコープ設定	処理結果
	StartScope	入力情報をセルから取得し、DLL の StartScope 関数を実行し、結果をセル上に出力する。	第 1 引数：スコープ設定	なし
	ScopeGetConditionButton_Click	ScopeGetCondition ボタン押下時のイベント。 入力情報をセルから取得し、DLL の ScopeGetCondition 関数を実行し、結果をセル上に出力する。	なし	なし
	ScopeGetDataButton_Click	ScopeGetData ボタン押下時のイベント。 入力情報をセルから取得し、DLL の ScopeGetData 関数を実行し、結果をセル上に出力する。	なし	なし
	ClearGraphData	Scope 機能で取得したデータを表示しているセルをクリアする。	なし	なし
	ScopeStopButton_Click	ScopeStop ボタン押下時のイベント。 入力情報をセルから取得し、DLL の ScopeStop 関数を実行し、結果をセル上に出力する。	なし	なし
	MapToCsvMapPathSelectButton_Click	MapToCsvMapPathSelect ボタン押下時のイベント。 ファイル選択ダイアログで Map ファイルを選択し、結果をセル上に出力する。	なし	なし
	MapToCsvCsvPathSelectButton_Click	MapToCsvCsvPathSelect ボタン押下時のイベント。 ファイル指定ダイアログで CSV ファイルを出力するパスを指定し、結果をセル上に出力する。	なし	なし
	MapToCsvButton_Click	MapToCsv ボタン押下時のイベント。 入力情報をセルから取得し、DLL の ConvertMapToCSV 関数を実行し、結果をセル上に出力する。	なし	なし
	MapToMemMapPathSelectButton_Click	MapToMemMapPathSelect ボタン押下時のイベント。 ファイル選択ダイアログで Map ファイルを選択し、結果をセル上に出力する。	なし	なし
	MapToMemButton_Click	MapToMem ボタン押下時のイベント。 入力情報をセルから取得し、DLL の ConvertMapToMemory 関数を実行し、結果をセル上に出力する。	なし	なし

	AutoReadButton_Click	AutoRead ボタン押下時のイベント。 AutoReadStartButton 押下時：入力情報をセルから取得し、TimerModule の TimerStart 関数を実行する。 AutoReadStopButton 押下時：AutoReadStop 関数を実行する。	なし	なし
	AutoRead	AutoReadStartButton 押下時に取得した情報で DLL の Read 関数を実行し、結果をセル上に出力する。	なし	なし
	AutoReadStop	TimerModule の TimerStop 関数を実行する。	なし	なし
Common	GetAddress	Map シートを検索し、変数名からアドレスおよび、データ型を取得する。	第 1 引数：変数名 第 2 引数：アドレス格納用変数 第 3 引数：データ型格納用変数	処理結果
	VaridationValues	セルの入力値チェックを行う。	第 1 引数：セル名 第 2 引数：エラーメッセージ	処理結果
	SelectFilePath	Map ファイルの選択を行う。	なし	選択したパス
	GetSaveFilePath	CSV ファイルのパスの指定を行う。	なし	指定したパス
	ConvertTypeToValue	Type シートの各表を検索し、型名から値に変換する。	第 1 引数：type の文字列 第 2 引数：検索する表のレンジ名称	処理結果の値
	ConvertResultValueToMean	Type シートの各表を検索し、値から型名に変換する。	第 1 引数：type の値 第 2 引数：検索する表のレンジ名称	処理結果の文字列
TimerModule	TimerProc	SampleSheet の AutoRead 関数を実行する。	なし	なし
	TimerStart	TimerProc 関数を周期実行するタイマーを起動する。	第 1 引数：周期	なし
	TimerStop	タイマーを停止する。	なし	なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.6.30	—	初版発行

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。