(2020.1.21)

アカデミック スカラロボット C言語開発環境導入の手引き

ヴイストン株式会社

本資料は、アカデミック スカラロボット（以下「本製品」と記述）を WindowsPC から C 言語プログラミングする際の、開発環境の導入とサンプルソースのビルドの手順について説明したものです。

本製品は、PC と USB ポートで接続し、PC 側よりプログラムで通信制御します。PC 側のプログラムとしては、シミュレータ機能搭載のモーションエディタ「SCARA Programmer」を利用できますが、無償公開されているプログラム開発環境・通信ライブラリなどを利用して、C 言語などのプログラミングによって制御プログラムを作成することが可能です。

目次

開発環境の導入

本項目では、PC に開発環境を準備する手順について説明します。本製品では、C 言語の統合開発環境として、Microsoft Visual Studio が使用できます（それ以外の開発環境も存在しますが、本資料では説明を省きます）。2020 年 1 月現在、Microsoft Visual Studio Community が無償公開されており、こちらで開発が可能です。なお、以前のバージョンの Visual Studio をお持ちの方は、新しい Visual Studio のインストールは不要です。

1. お使いの PC に Visual Studio がインストールされていない場合は、下記 URL にアクセスし、「Visual Studio Community 2019」をインストールしてください。

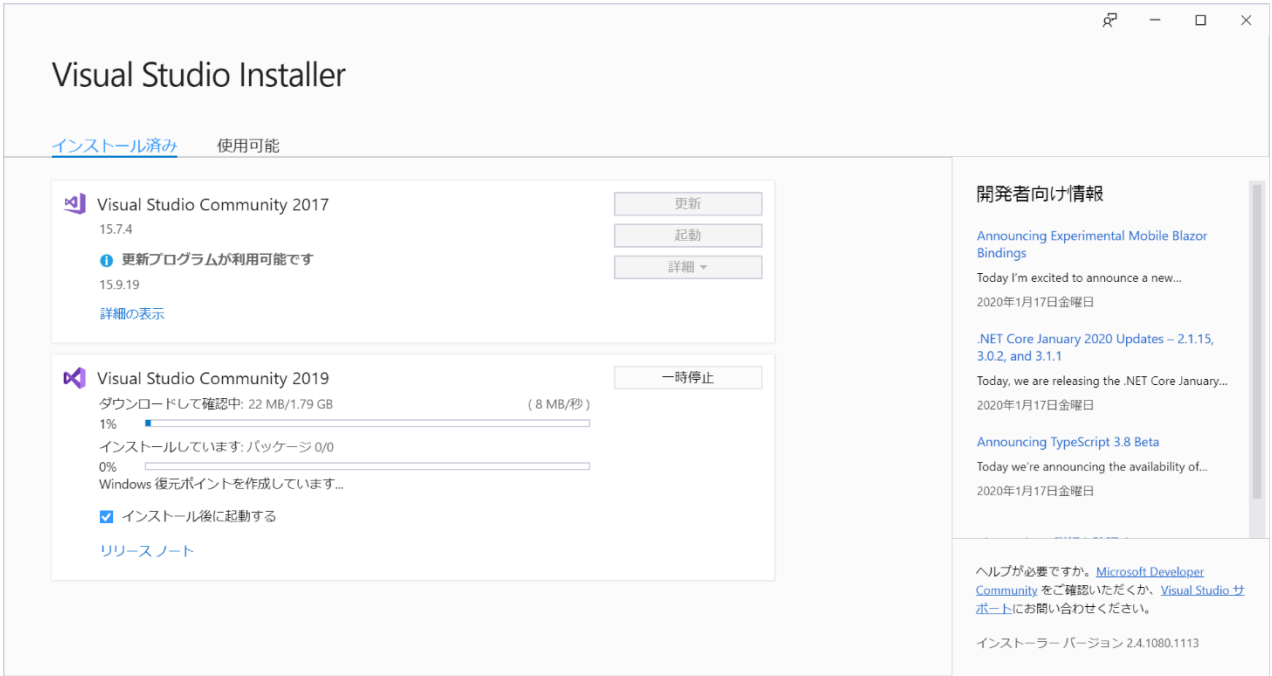
■Microsoft Visual Studio Community 配布 URL

https://visualstudio.microsoft.com/ja/downloads/

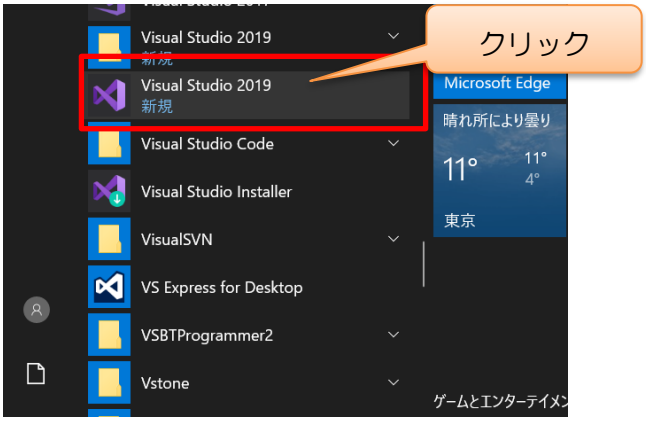
vs_Community.exe というファイルがダウンロードされるので実行してください。なお、インストール中はネットワークから逐次必要なファイルをダウンロードするため、必ずネットワークの接続環境をご用意ください。

実行すると下記の画面を開きます。「C++によるデスクトップ開発」をクリックし、続いて「インストール」をクリックしてください。

インストールを開始し、画面に進行状況が表示されます。完了までに PC の再起動を要求される場合がありますので、指示に従って操作してください。



インストールが完了すると、Windows のスタートメニューに「Visual Studio 2019」が追加されます。起動するときはこちらをクリックします。



2.　　ロボットとの通信に必要なライブラリをインストールします。

ロボット本体との通信のために、Sillocon Laboratory 社「CP2110-USB-to-UART インターフェースライブラリ （以後「通信ライブラリ」と記述）」を使用します。下記 URL より、ライブラリをダウンロードして PC にインストールしてください。

■CP2110-USB-to-UART インターフェースライブラリ配布 URL

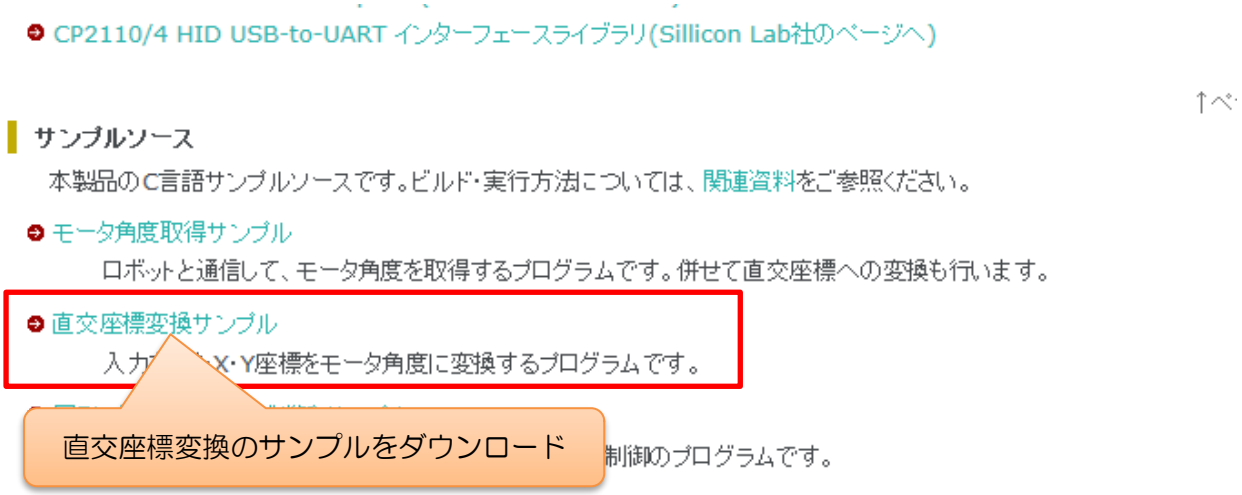
http://jp.silabs.com/products/interface/Pages/CP2110EK.aspx

インストーラのファイルをダウンロードしたら、実行して PC にライブラリをインストールしてください。また、後の設定でインストール先のフォルダを使用するので、インストールが完了したら確認してください（標準の設定では、「C:¥SiLabs¥MCU¥CP2110_4_SDK¥Library¥Windows¥」にインストールされます）。また、同じ web ページより、ライブラリに関する説明資料などをダウンロードしてください。

3. サンプルソースを製品 web ページよりダウンロードします。ここでは「直交座標変換」のサンプルをダウンロードしてください。

■アカデミック スカラロボット ダウンロードページ URL

http://www.vstone.co.jp/products/scara_robot/download.html



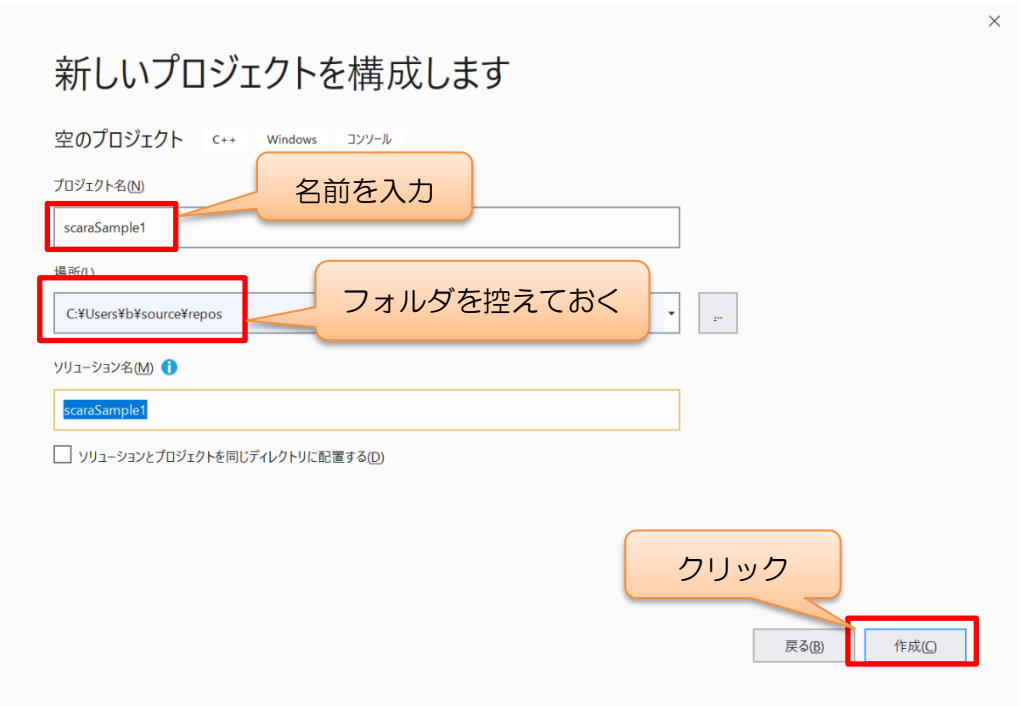
4. Visual Studio を起動し、新しいプロジェクトを作成します。

最初に起動すると、(Microsoft アカウントの入力を求められます。既にお持ちである、もしくは新規に作成される場合は、それぞれアカウント情報を入力してください。アカウントを利用しない場合は「後で行う」をクリックしてください。続く画面表示のスタイルなども適時好みのものをお選び下さい。

初回起動の設定が終わると下記の画面を表示します。「新しいプロジェクトの作成」をクリックしてください。

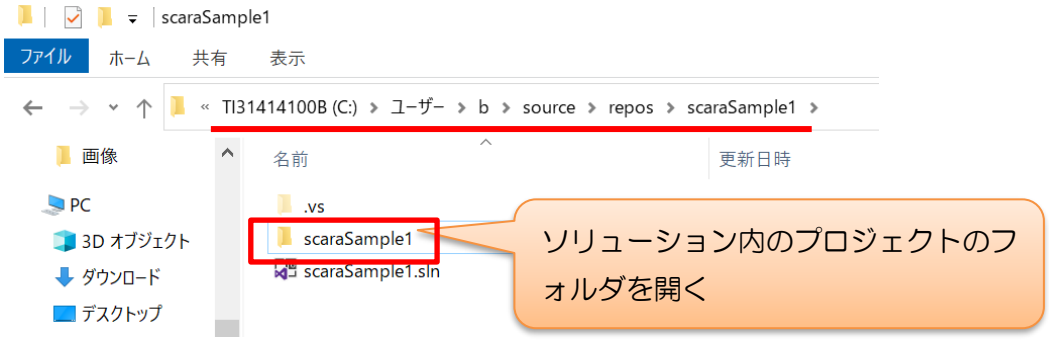
作成するプロジェクトの種類は、「空のプロジェクト」です。

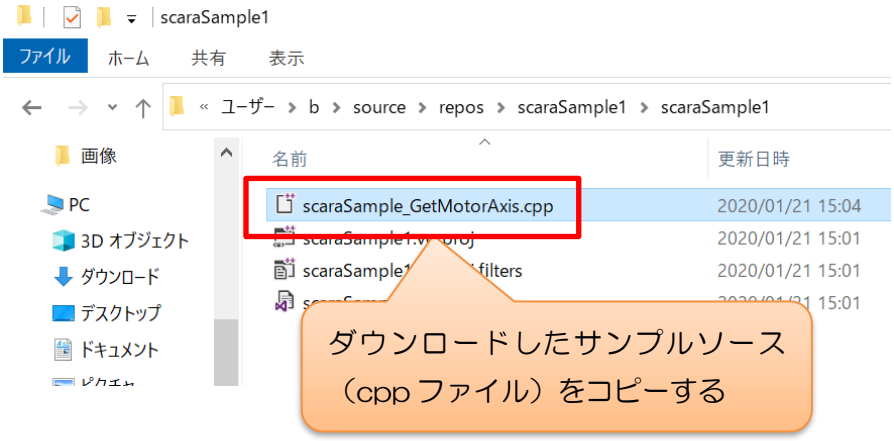
プロジェクト名を入力してください。また、ここで表示される「場所」は、実際にプロジェクト・ソリューションが作成されるフォルダです。この次に使用するため、この場所を控えておいてください。



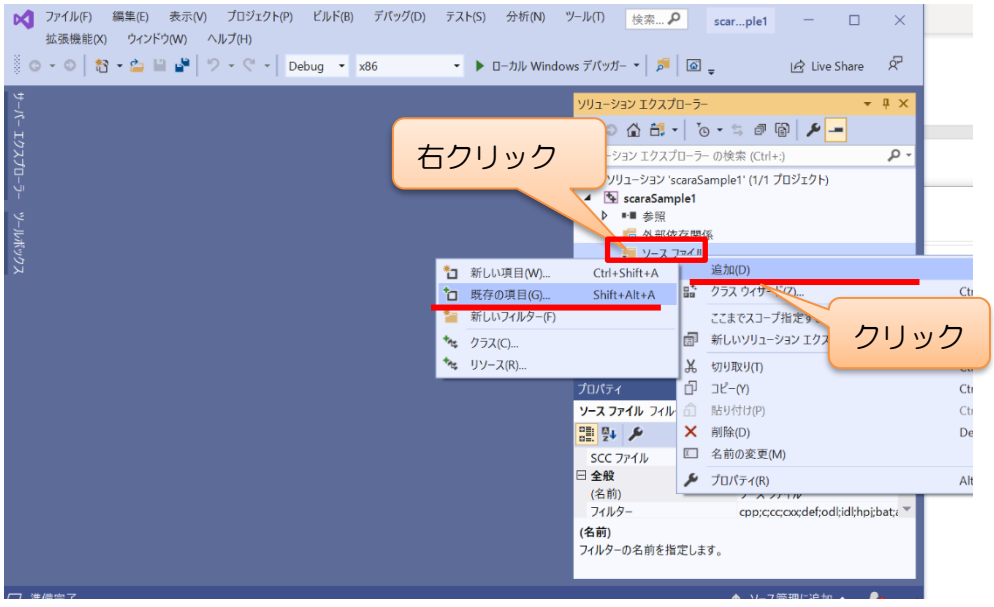
5.　ダウンロードしたソースをプロジェクトに追加してください。

エクスプローラーより、プロジェクト・ソリューションが作成されたフォルダを開いてください。先ほどの「場所」に表示されたのがソリューションのフォルダ、その中にある、先ほど設定したプロジェクト名と同じフォルダがプロジェクトのフォルダです。プロジェクトのフォルダに、ダウンロードしたサンプルソースをコピーしてください。

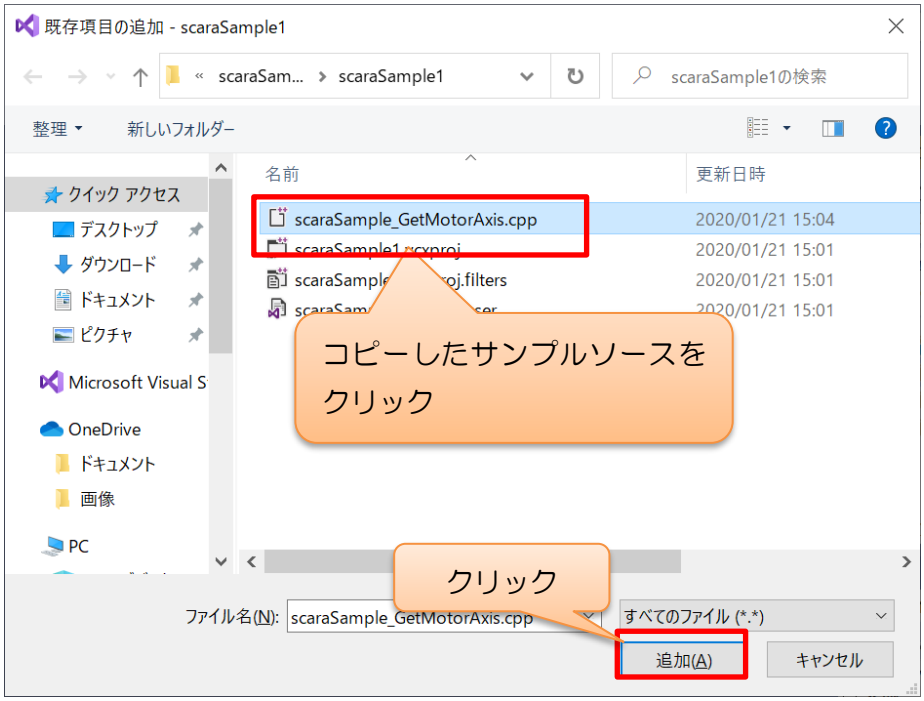


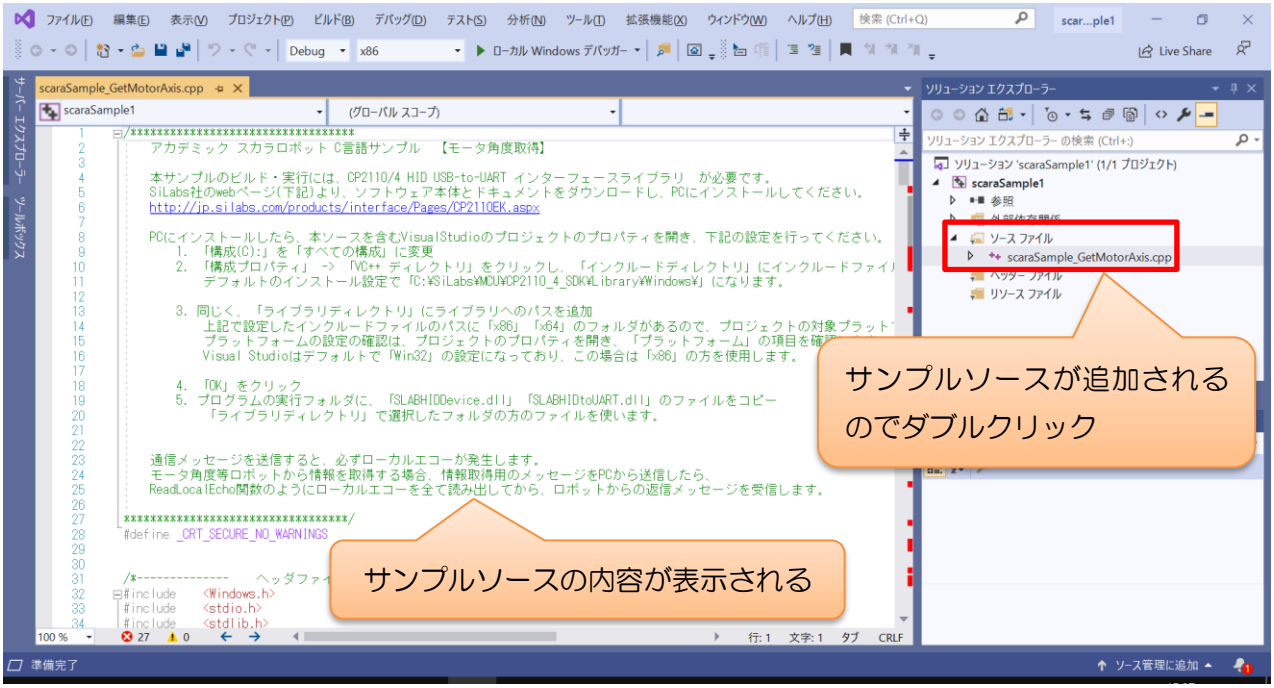


ファイルをコピーしたら、VisualStudio の「ソリューションエクスプローラー」（ツリー状の表示部分）より、作成したプロジェクトの「ソースファイル」のフォルダを右クリックします。クリックするとポップアップメニューを表示するので、「追加」→「既存の項目」をクリックしてください。



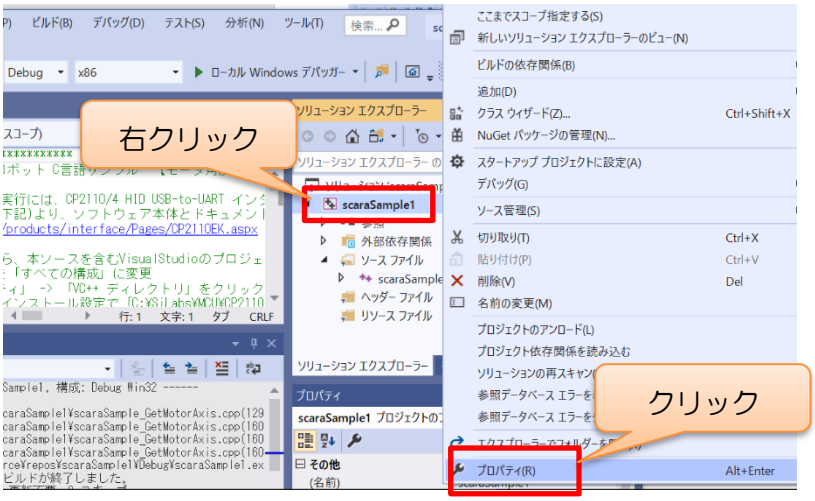
クリックするとファイル選択ダイアログを開くので、先ほどコピーしたサンプルソースファイルを選んで「追加」をクリックします。ファイルを選択すると、ソリューションエクスプローラーの「ソースファイル」のフォルダに、選択したファイルが追加されます。



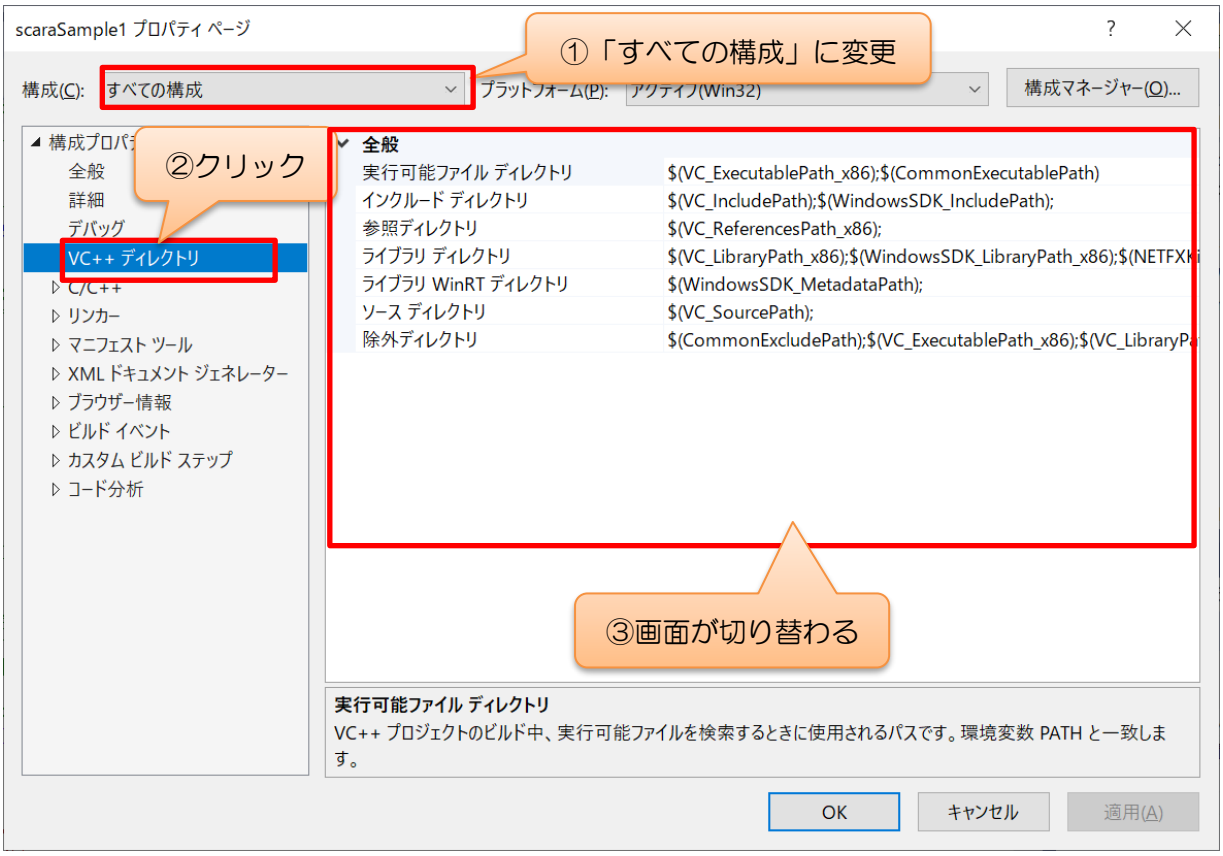


6.　プロジェクトの参照フォルダの設定で、通信ライブラリのインストールフォルダを参照します。

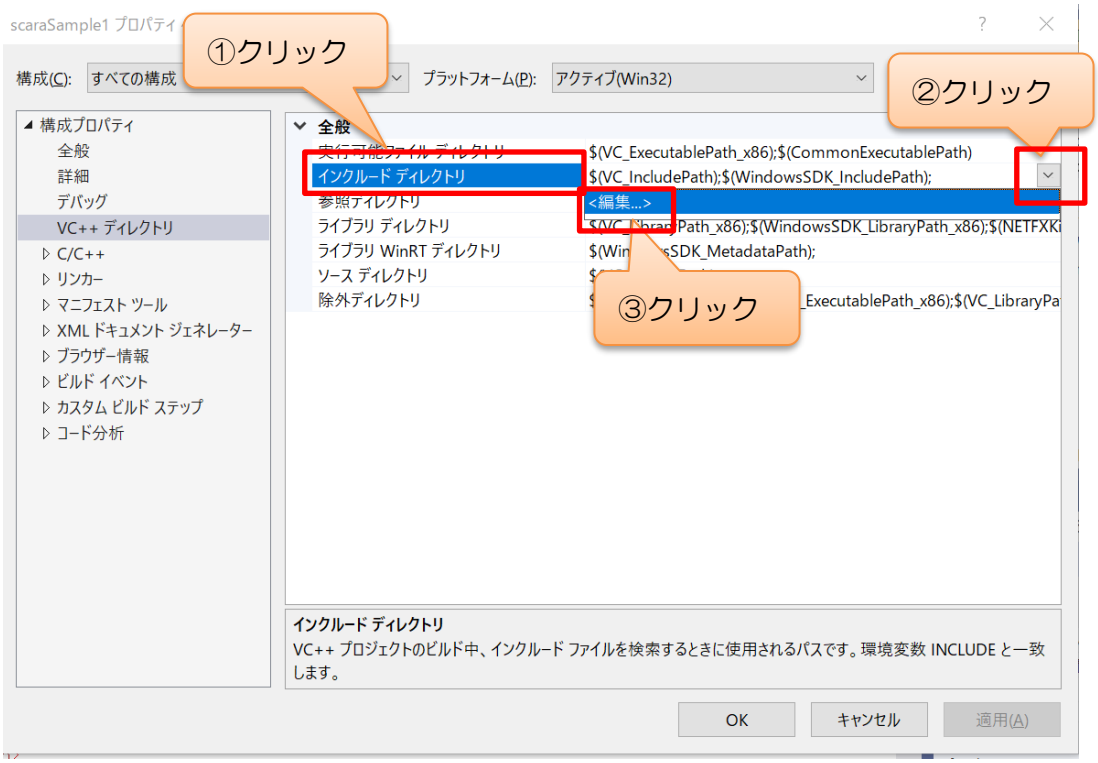
ソリューションエクスプローラーより、プロジェクト名の項目を右クリックし、表示されるメニューより「プロパティ」をクリックしてください。



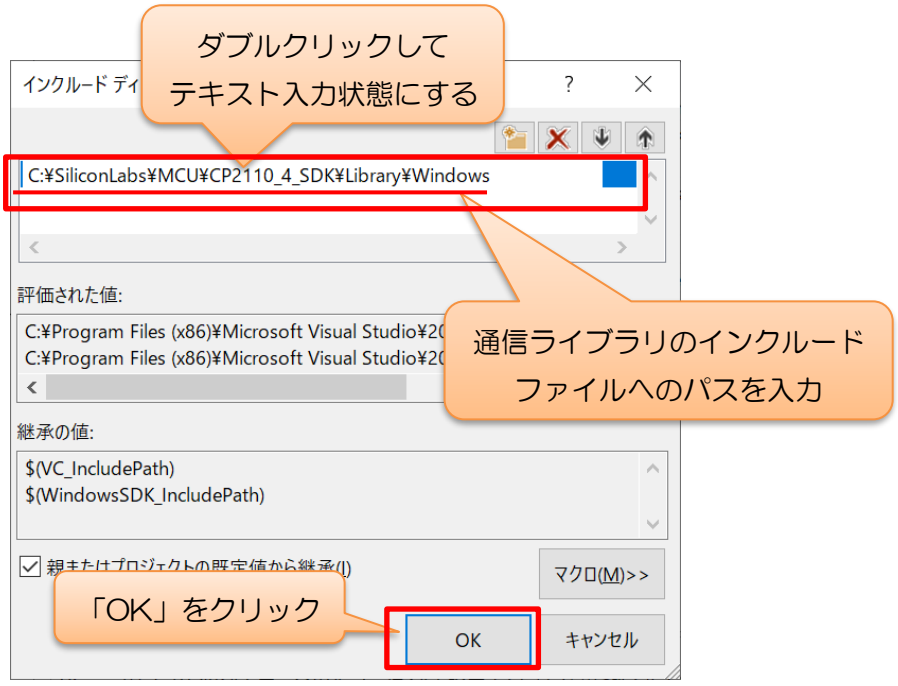
クリックするとプロジェクトのプロパティ設定ダイアログを開きます。ダイアログ左上の「構成」を「すべての構成」に変更し、続いてダイアログ左のツリーより「構成プロパティ」→「VC++ ディレクトリ」を選択してください。



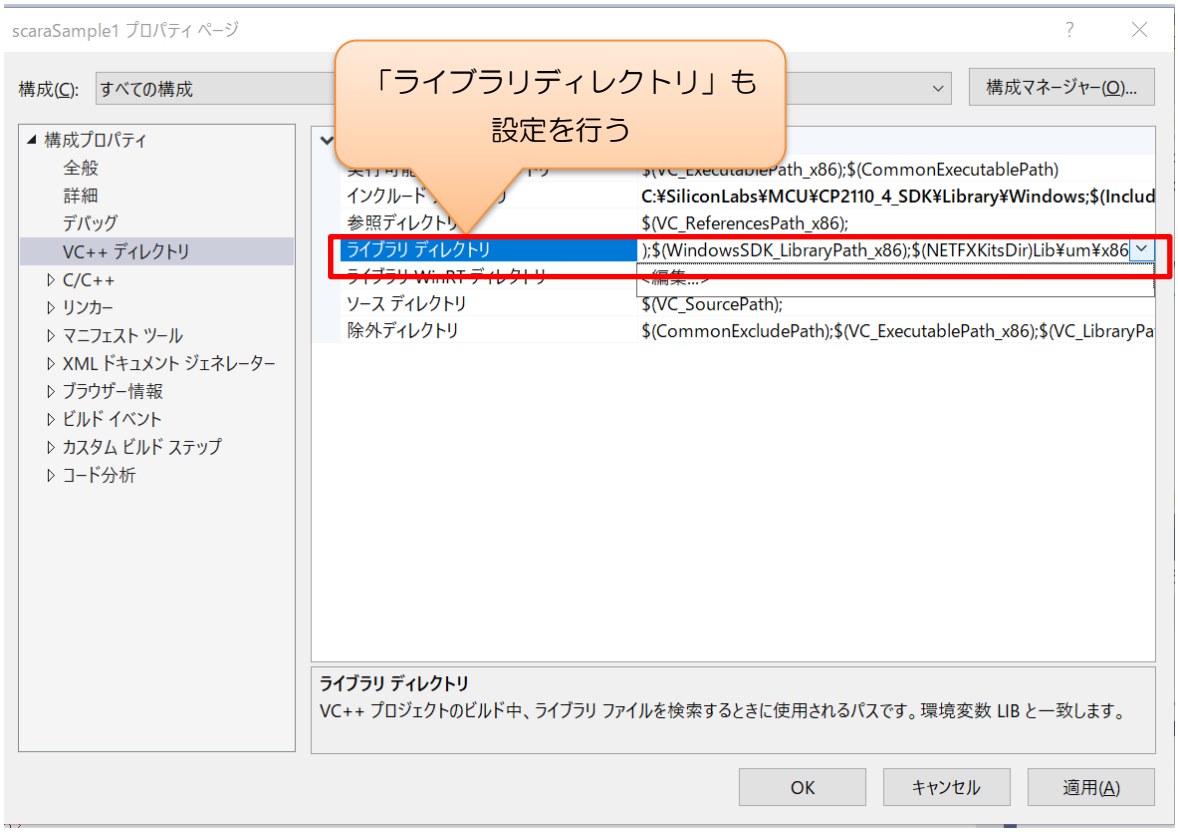
選択するとダイアログ右の設定画面が切り替わるので、「インクルードディレクトリ」の項目をクリックしてください。クリックすると項目の右端に【▼】のマークが表示されるのでクリックし、更に「<編集…>」をクリックします。



クリックすると、ディレクトリの設定ダイアログを開きます。ダイアログ上側の余白部分をダブルクリックするとテキスト入力の状態になるので、先ほど確認した通信ライブラリのインクルードファイルが含まれたフォルダを開きます。このフォルダは、デフォルトの設定では「C:¥SiliconLabs¥MCU¥CP2110_4_SDK¥Library¥Windows」になります。

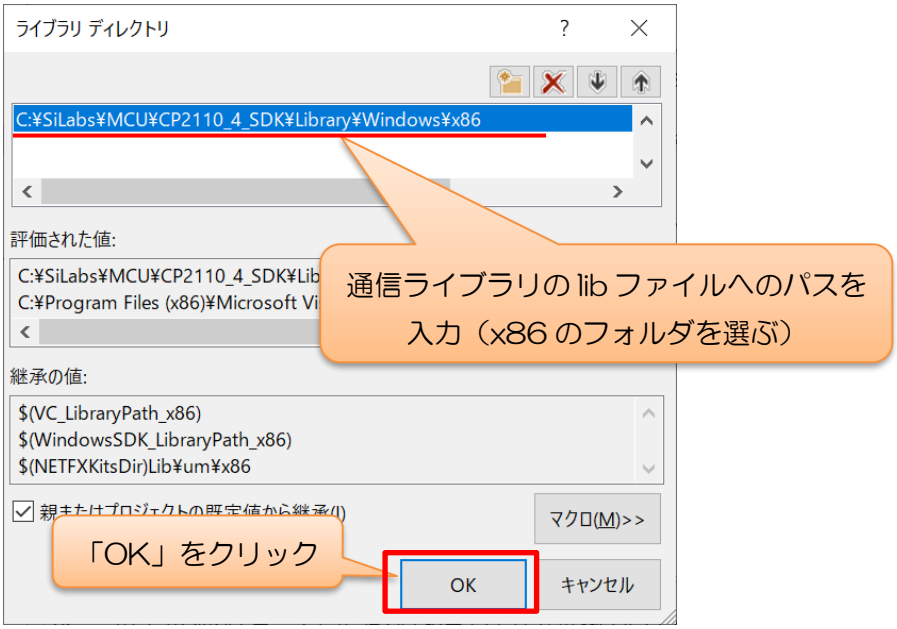


インクルードディレクトリを設定したら、続いて「ライブラリ ディレクトリ」を同様に設定します。



先ほどと同じ方法でフォルダの選択画面まで開き、通信ライブラリの lib ファイルが含まれたフォルダを開きます。このフォルダは、デフォルトの設定では「C:¥SiliconLabs¥MCU¥CP2110_4_SDK¥Library¥Windows¥x86」になります。

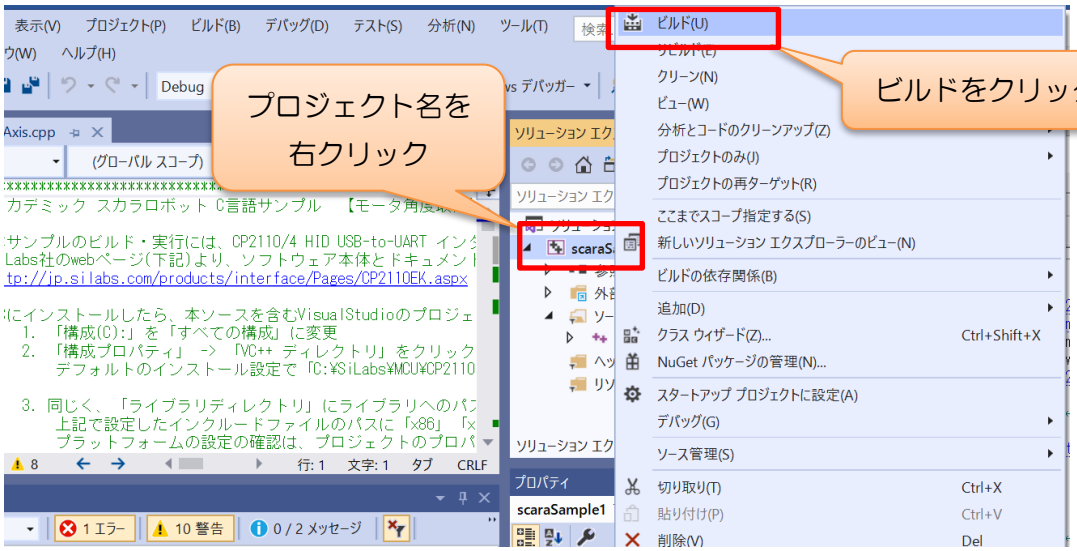
※ lib ファイルは、x64 のフォルダにも同名の物が存在しますが、Visual Studio の標準のビルドターゲットが Win32(32bit)なので、x86 のフォルダを選択してください。



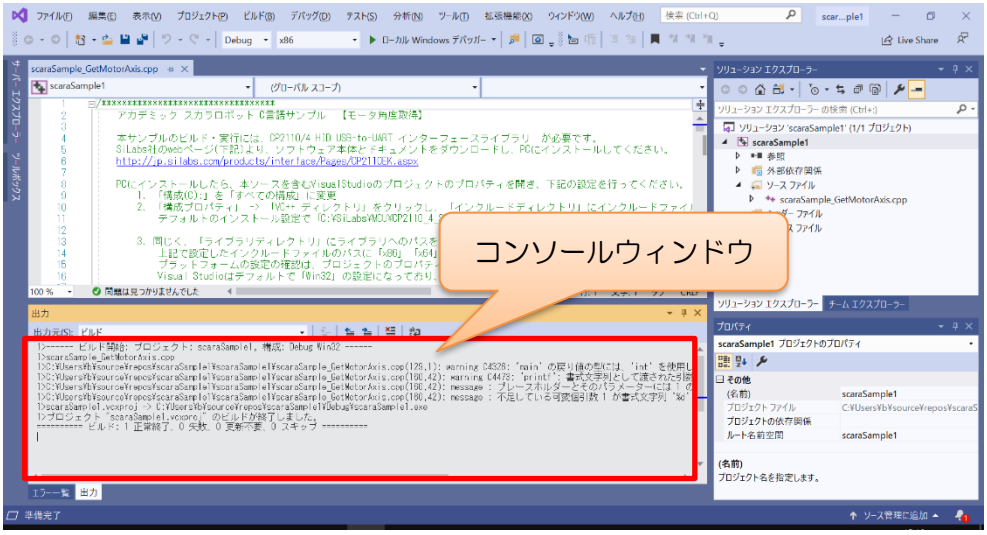
それぞれの設定が完了したら、プロジェクトのプロパティダイアログの「適用」をクリックして、設定を適用してください。

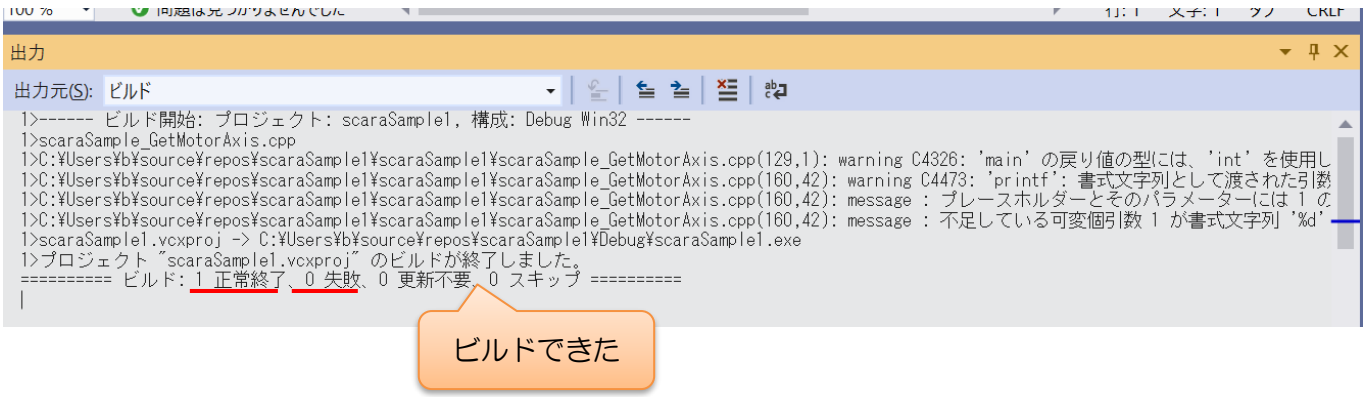
7.　サンプルソースをビルドします。

以上でプロジェクトに関する設定は完了です。ここで一度ソースをビルドしてみましょう。ソリューションエクスプローラーのプロジェクト名の項目を右クリックし、メニューより「ビルド」をクリックしてください。

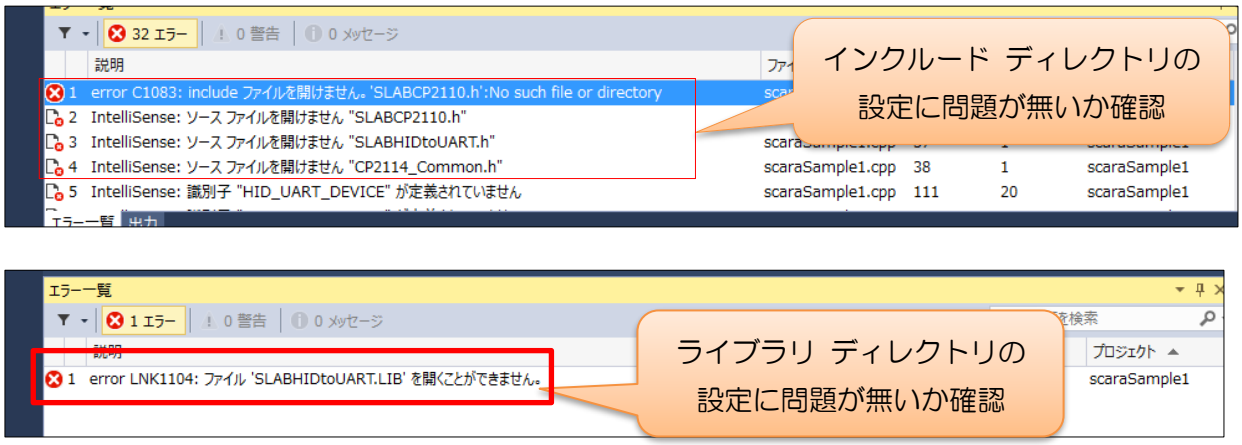


ビルド結果は、画面下のコンソールウィンドウに表示されます。ビルド結果の表示を確認して、問題があれば解説の内容を確認しましょう(Visual Studio のバージョンによっては、図のように警告(Warning)が表示される場合がありますが、実行には差し支えありません)。





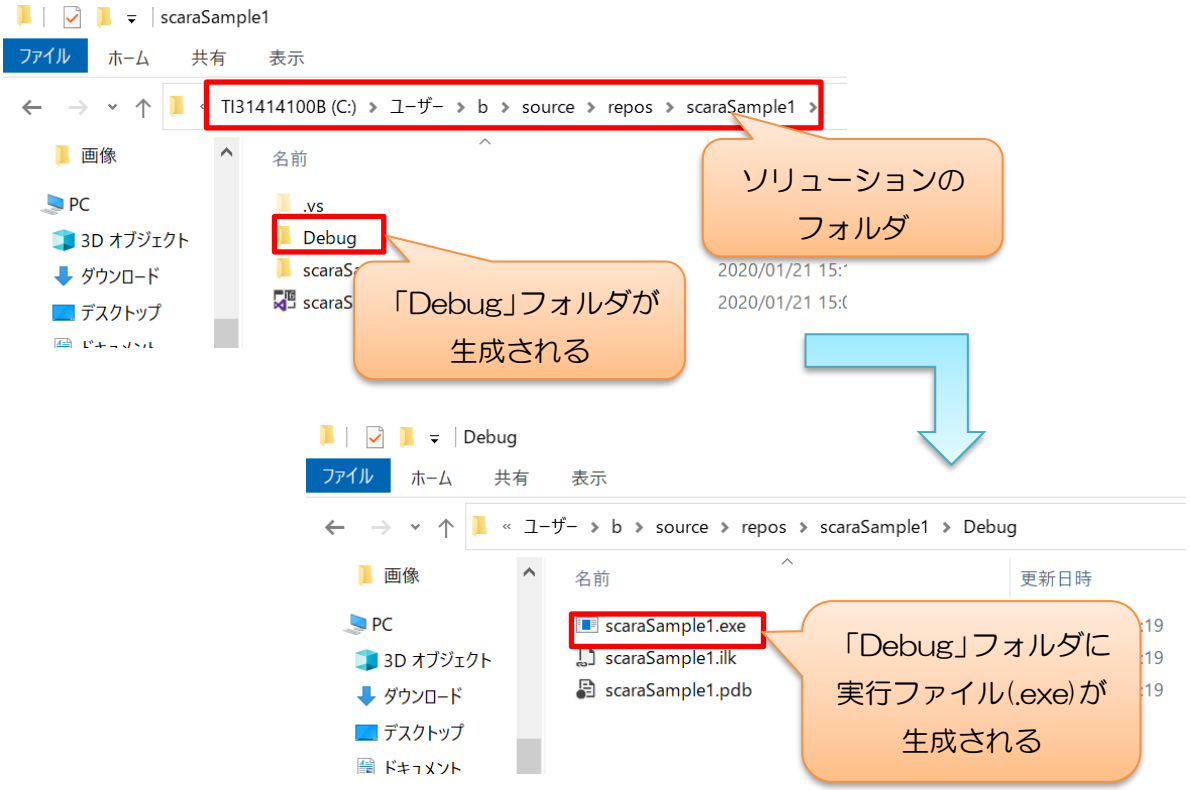
下記は、設定に問題がありビルドに失敗した例です。



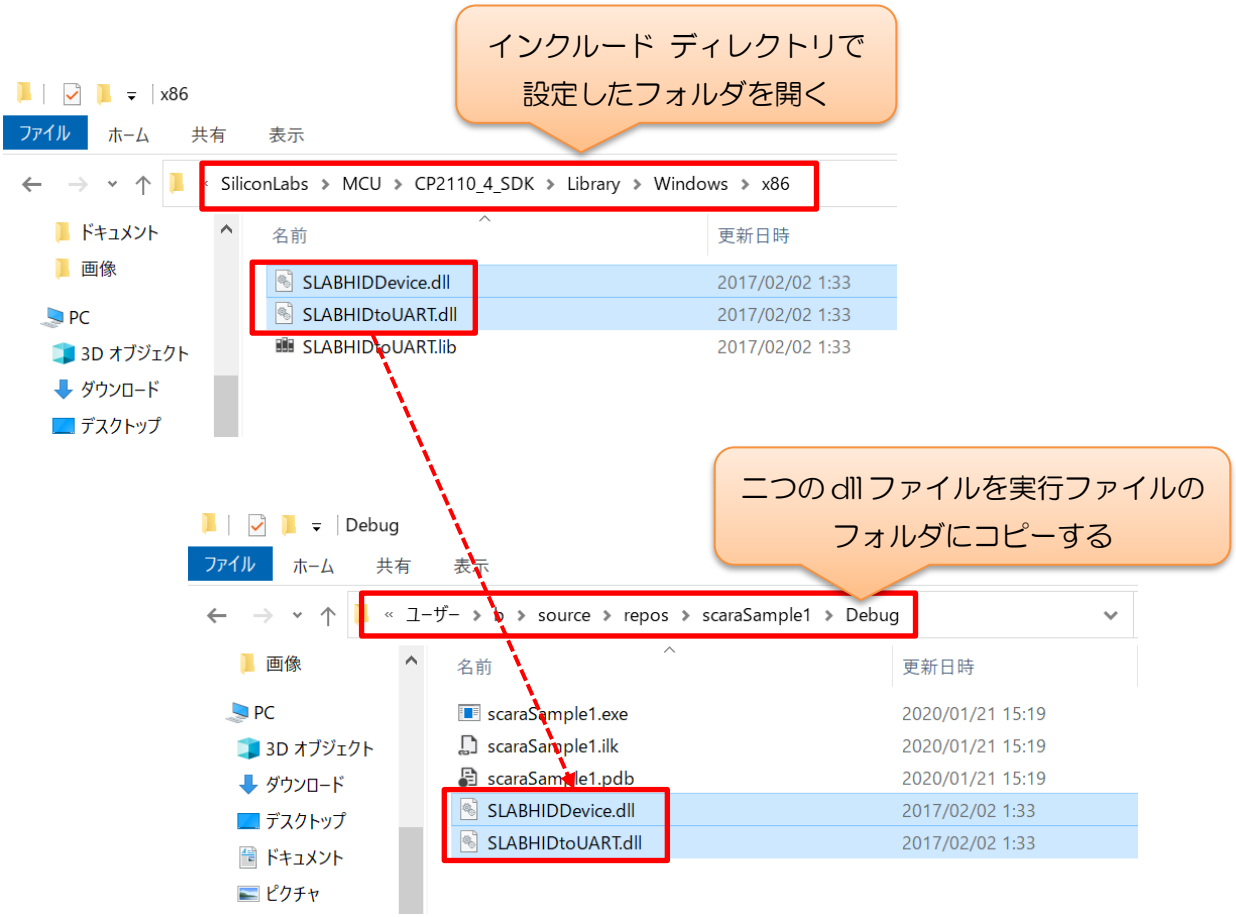
8.　　Dll ファイルを実行ファイルと同じフォルダにコピーして、プログラムを実行します。

プログラムをビルドできたら、ソリューションのフォルダに「Debug」と言う名前のフォルダが作成され、その中に「(プロジェクト名).exe」というファイルが作成されます（これとは別に「(ソリューション名)¥(プロジェクト名)¥Debug」というフォルダができますが、そちらは関係ありません）。

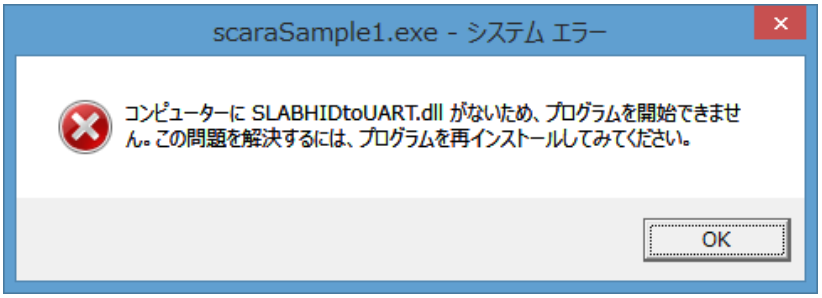
実行ファイルをダブルクリックするとプログラムが起動しますが、プログラムを動作させるためには、通信ライブラリの dll ファイルを、実行ファイルと同じフォルダにコピーする必要があります。



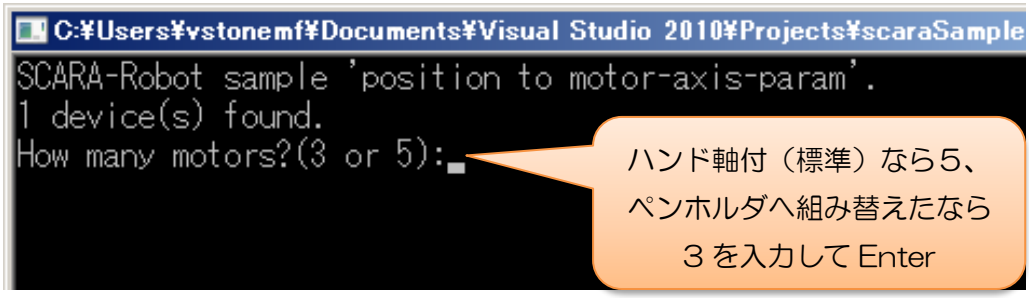
先ほどインクルードディレクトリの設定で指定したフォルダを開き、「SLABHIDDevice.dll」「SLABHIDtoUART.dll」の二つのファイルを、前述の実行ファイルが作成されたフォルダにコピーしてください。



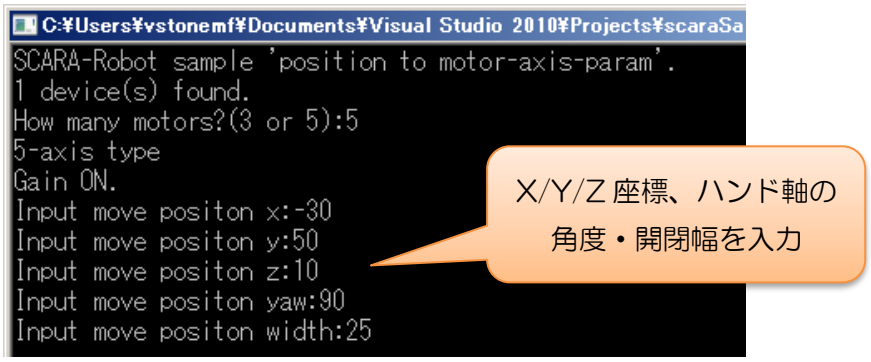
Dll ファイルをコピーしたら、ロボット本体を PC に接続して、実行ファイルをダブルクリックして実行してください。実行したときに下記のエラーメッセージが表示される場合、正しく dll ファイルがコピーできているかご確認ください。



実行すると、最初に使用する軸数を聞かれるので、ロボット本体の現在の軸数に合わせて、キーボードより’３‘か’５’を入力して Enter を押してください。軸数の確認が行われず、すぐにプログラムが終了する場合は、ロボット本体が PC に正しく接続されているかご確認ください。



軸数を入力すると、ロボット本体のモータが現在の位置で ON になります。モータが ON にならない場合、ロボット本体に電源を接続しているかご確認ください。続いて、座標とモータ角度の入力確認が行われます。３ 軸の場合は X/Y/Z 座標(mm)、５ 軸の場合はそれに加えてハンドの回転軸角度(360° 法)と開閉軸の幅(mm)を聞かれるので、それぞれ数値を入力してください(座標系については、後述の「座標系の説明」をご参照ください)。



数値を入力すると、ロボット本体が入力した数値に合わせたポーズに変形します。動かすときは、モータロックが発生しないように、十分ご注意ください。

他のサンプルソースをビルドする場合も、同様の手順でプロジェクトの作成及び設定を行ってください。

サンプルソースについて

　この項目では、配布している各サンプルソースの概要、及び主な関数やマクロなど、プログラミングに関する必要情報について説明します。プログラムを改造する際には、本項目の説明をご参照ください。

サンプルソース一覧

　ダウンロードページで配布しているサンプルソースの種類は下記の通りです。

・モータ角度取得サンプル

　ロボットからのモータ角度の読み取り、及びモータ角度から直交座標への変換処理のサンプルです。

・直交座標変換サンプル

　直交座標からモータ角度への変換処理、及びモータの制御に関するサンプルです。

・物体運搬サンプル

　付属のスポンジキューブを運搬し、モーション制御を学習するためのサンプルです。

・図形描画サンプル

　三角形・四角形を描画し、PtoP 制御と直線補間を学習するためのサンプルです。

・円弧描画サンプル

　二点の座標・円の半径・回転方向を指定して円弧を描画し、CP 制御を学習するためのサンプルです。

主な関数

サンプルソースに含まれる主な関数について説明します。なお、サンプルソースによっては、一部必要ない関数を省略しているものがあります。

■CP2110/4 HID USB-to-UART インターフェースライブラリに関連する関数

これらの関数は、通信ライブラリに含まれる関数、または関係の深い処理です。一部の引数や戻り値で使用されている宣言子はライブラリ内で定義されています。これらの詳細は、通信ライブラリの説明資料をご参照ください。

HID_UART_STATUS HidUart_GetNumDevices(DWORD* numDevices, WORD vid, WORD pid)

概要： 現在 PC に接続されているロボットの数を取得する。関数が成功すると、ポインタで与えた変数に現在のロボットの接続数が代入される。

引数： DWORD* numDevices … ロボットの接続数を取得する変数へのポインタ
WORD vid , WORD pid … ロボットの VenderID、ProductID。
それぞれマクロの VID , PID を代入する。

戻り値：成功の場合 HID_UART_SUCCESS、失敗の場合それ以外を返す。HID_UART_STATUS の各数値については、ライブラリの説明資料を参照。

HID_UART_STATUS HidUart_Open(HID_UART_DEVICE* device, DWORD deviceNum, WORD vid, WORD pid)

概要： ロボットとの通信ハンドルを取得し、通信を開始する。関数が成功すると、ポインタで与えた変数に、通信に必要なハンドルの値が代入される。

引数： HID_UART_DEVICE* device … 通信ハンドルを取得する変数へのポインタ
DWORD deviceNum … 接続するロボットの番号。複数のロボットを同時に制御する場合、この数値によって接続先を選択する。
WORD vid, WORD pid … ロボットの VenderID、ProductID。
それぞれマクロの VID , PID を代入する。

戻り値：成功の場合 HID_UART_SUCCESS、失敗の場合それ以外を返す。HID_UART_STATUS の各数値については、ライブラリの説明資料を参照。

HID_UART_STATUS HidUart_Close(HID_UART_DEVICE device)

概要： 取得したロボットの通信ハンドルを閉じてロボットとの通信を終了する。

引数： HID_UART_DEVICE device … 取得した通信ハンドル

戻り値：成功の場合 HID_UART_SUCCESS、失敗の場合それ以外を返す。HID_UART_STATUS の各数値については、ライブラリの説明資料を参照。

HID_UART_STATUS SetTXOpenDrain(HID_UART_DEVICE dev)

概要： ロボットの通信ポートの設定で、TX を Open-Drain へ変更する。ロボットと通信する場合は、必ずこの設定変更を行う必要がある（設定はロボット本体に記録されるため、一度変更していれば再度呼び出しの必要はない）

引数： HID_UART_DEVICE dev … 取得した通信ハンドル

戻り値：成功の場合 HID_UART_SUCCESS、失敗の場合それ以外を返す。HID_UART_STATUS の各数値については、ライブラリの説明資料を参照。

■座標変換に関連する関数

これらの関数は、モータ角度と直交座標の変換を行う処理です。座標系については「座標系の説明」の項目をご参照ください。

void pos_to_rad(double x, double y, double z,double yaw, double w,short *sPos,int sign,int num)

概要： X/Y 座標と回転軸の角度からアームの角度へ変換する。

引数： double x , y , z … 変換元の X/Y/Z 座標(mm 単位)
double yaw … 変換元のハンド回転軸の角度(360° 法)
double w … 変換元のハンド開閉軸の幅(mm 単位)
short *sPos … 変換後のモータ角度を代入する配列変数へのポインタ
int sign … アームの折れ曲がる向き。
正の値だと時計回り、負の値だと反時計回りに折れ曲がる
int num … アームの軸数。４以上の場合、ハンドの２軸の計算を行う

戻り値：なし

void rad_to_pos(double *x,double *y, double *z ,double *yaw, double *w ,short *sPos,int num)

概要： モータ角度を変換して X/Y 座標と回転軸角度を求める。

引数： double *x , *y, *z … 変換後の X/Y/Z 座標を代入する変数へのポインタ
double *yaw … 変換後のハンド回転軸角度を代入する変数へのポインタ
double *w … 変換後のハンド開閉軸幅を代入する変数へのポインタ
short *sPos … 変換元のモータ角度を格納した配列変数へのポインタ
int num … アームの軸数。４以上の場合、ハンドの２軸の計算を行う

戻り値：なし

■モータの制御に関連する関数

これらの関数は、モータとの通信を行う処理です。モータの通信使用については「モータに関する資料」で紹介しているサーボモータ「RS304MD」の資料をご参照ください。また、引数に配列変数を使用するものは、他の引数で与えるモータの数やバッファのサイズ等と差異が無いようにしてください。これらのサイズに違いがあると、不正なメモリアクセスによるエラーが発生します。

int RSTorqueOnOff(HID_UART_DEVICE dev, short sMode ,BYTE id,int num)

概要： モータの ON/OFF を切り替える。ID が連続した複数のモータを一度に制御可能。

引数： HID_UART_DEVICE dev … ロボットの通信ハンドル
short sMode … モータ ON/OFF の指定。0 の場合 off、1 の場合 ON になる
BYTE id … コマンドを送信するモータの ID。複数のモータを切り替える場合は先頭のモータの ID を代入する。
int num … コマンドを送信するモータの数。

戻り値：正しく通信できた場合は TRUE、通信エラーが発生した場合は FALSE

int RSGetAngle(HID_UART_DEVICE dev ,BYTE id,short *getParam)

概要： モータから現在の角度を取得する関数。モータの角度は 1 個ずつしか取得できない。

引数： HID_UART_DEVICE dev … ロボットの通信ハンドル
BYTE id … 角度を取得するモータの ID
short *getParam … 取得した角度値（0.1° 単位）

戻り値：正しく通信できた場合は TRUE、通信エラーが発生した場合は FALSE

int RSMove(HID_UART_DEVICE dev , short *sPoss, unsigned short sTime ,BYTE id,int num)

概要： 時間と角度を指定してモータを動かす関数。ID が連続した複数のモータを一度に制御可能。

引数： HID_UART_DEVICE dev … ロボットの通信ハンドル
short *sPoss … 送信するモータ角度を記録した配列変数へのポインタ。
unsigned short sTime … モータの遷移時間（10msec 単位）
BYTE id … コマンドを送信するモータの ID。複数のモータを切り替える場合は先頭のモータの ID を代入する。
int num … コマンドを送信するモータの数。

戻り値：正しく通信できた場合は TRUE、通信エラーが発生した場合は FALSE

int RSWriteMem(HID_UART_DEVICE dev , BYTE address , BYTE size , BYTE id , BYTE *data , int num)

概要： モータのメモリマップに数値を書き込む関数。アドレスが連続したメモリブロックや ID が連続した複数のモータを一度に制御可能。

引数： HID_UART_DEVICE dev … ロボットの通信ハンドル

BYTE address … データ書き込みを行うメモリマップのアドレス。2byte 以上書き込む場合は、先頭のアドレスを指定する

BYTE size … 書き込むデータのサイズ。複数のモータに書き込む場合は、モータ 1 個当たりのデータサイズを指定する

BYTE id … 書き込みを行うモータの ID。複数のモータに書き込む場合は、先頭のモータの ID を指定する

BYTE *data … 書き込むデータ内容を記録したメモリブロックへのポインタ。データ部は、必ず BYTE id * int num のサイズであること

Int num … 書き込みを行うモータの個数。

戻り値：正しく通信できた場合は TRUE、通信エラーが発生した場合は FALSE

int ReadLocalEcho(HID_UART_DEVICE dev ,unsigned char *sendbuf,DWORD data_len)

概要： ロボットへコマンドを送信したときに発生するローカルエコーを読み込む。main 関数内では直接使用していないが、ロボットとの通信を行う各関数内では、通信処理後にこの関数を用いて正しくコマンドを送信できているか確認している。

引数： HID_UART_DEVICE dev … ロボットの通信ハンドル

unsigned char *sendbuf … 送信コマンドを記録した配列変数へのポインタ

DWORD data_len … 送信コマンドの長さ

戻り値：元の送信コマンドと全く同じ文字列を取得できれば TRUE、文字数や内容に違いがあれば FALSE

主なマクロ

各サンプルプログラムで共通する主なマクロについて紹介します。

■通信に関する設定

VID　(0x10C4)

PID　(0xEA80)

ロボット本体の Vender ID、Product ID です

■ロボット本体の寸法に関する設定

AXISLEN_A　(80.0)

AXISLEN_B　(80.0)

アーム軸の第一関節、第二関節の軸間距離（mm）

FIELD_H　(210.0)

FIELD_W　(230.0)

ステージの縦幅・横幅(mm)

X_OFS　(FIELD_W/2 - 53.0)

ステージ中央からアームの根元（ID1 モータの出力軸）までの距離(mm)

■上下軸の距離・モータ角度の変換に関する設定

RAD_TO_HEIGHT(r)　(((double)r/(HEIGHT_RATE*10.0))*HEIGHT_RANGE)

モータ角度から上下軸の距離(mm)に変換するマクロ関数。

引数は、r=モータ角度値(0.1 度単位)。戻り値は距離(mm)

HEIGHT_TO_RAD(h)　(short) (h/HEIGHT_RANGE*HEIGHT_RATE*10.0)

距離からモータ角度に変換するマクロ関数。

引数は、h=距離(mm)。戻り値はモータ角度(0.1° 単位)

■ハンド開閉軸の距離・モータ角度の変換に関する設定

RAD_TO_WIDTH(r,p)

(((double)-r/(WIDTH_RATE*10.0))*WIDTH_RANGE + GROW_W*(p+1))

モータ角度からハンド軸の開閉幅に変換するマクロ関数。

引数は、r=モータ角度値(0.1 度単位)、p=爪の取り付け位置（0～3)。戻り値は距離(mm)。

WIDTH_TO_RAD(w,p)
　　(short)(-(w-GROW_W*(p+1))/WIDTH_RANGE*WIDTH_RATE*10.0)
　ハンド軸の開閉幅からモータ角度に変換するマクロ関数。
　引数は、w=開閉幅(mm)、p=爪の取り付け位置（0～3)。戻り値はモータ角度(0.1°単位)

ARM_RAD_RANGE　　(1350)
HAND_WIDTH_RANGE (350)
モータの可動範囲の設定（0.1 度単位)。アームの２軸（ID1,ID2）及びハンド開閉軸（ID5）の設定

■座標系に関する設定
BASE_ANGLE（180.0)
BASE_OFFSET_X　　(+0.0)
BASE_OFFSET_Y　　(+0.0)
　本体の向き（360°法）と原点からの移動量（mm）

CROW_POS　　(1)
　現在のハンド軸の爪のねじ穴番号。0～3 がねじ穴①～④に相当する

列挙子
　RS304D の内部パラメータ（メモリマップ）のアドレスについて定義した列挙子です。サンプルでは、モータ出力に関連するパラメータのみを定義しています。その他のアドレスを使用する場合は、別途モータの資料を参考に追記してください。

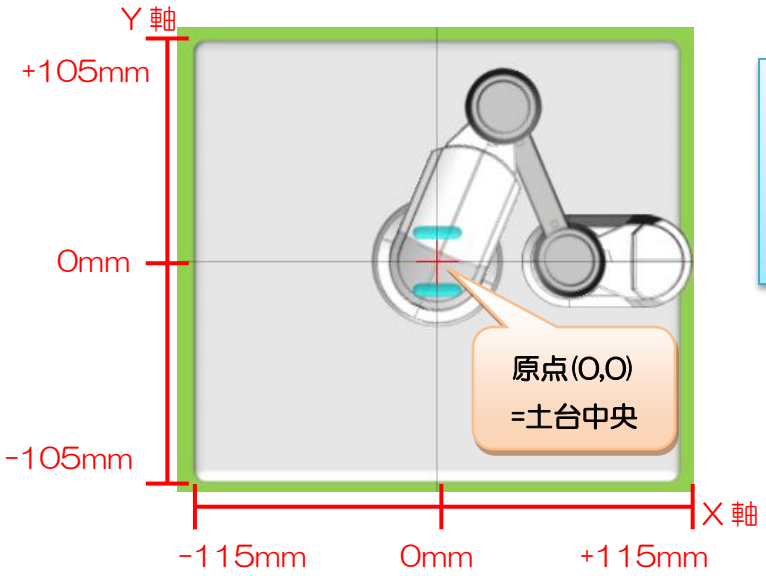
```
typedef enum{
        CW_Compliance_Margin        = 0x18, //コンプライアンスマージン（時計回り）
        CCW_Compliance_Margin       = 0x19, //コンプライアンスマージン（反時計回り）
        CW_Compliance_Slope         = 0x1a, //コンプライアンススロープ（時計回り）
        CCW_Compliance_Slope        = 0x1b, //コンプライアンススロープ（反時計回り）
        Punch_L                     = 0x1c, //パンチ（下位 byte）
        Punch_H                     = 0x1d, //パンチ（上位 byte）

        ROMSIZE                     = 30    //ROM 領域のサイズ

} RS304ROM;
```

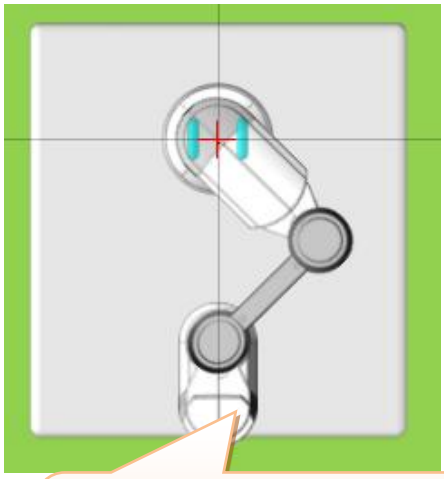
座標系の説明

サンプルソースでは、アームの座標系を全て下記の通りに統一しています。

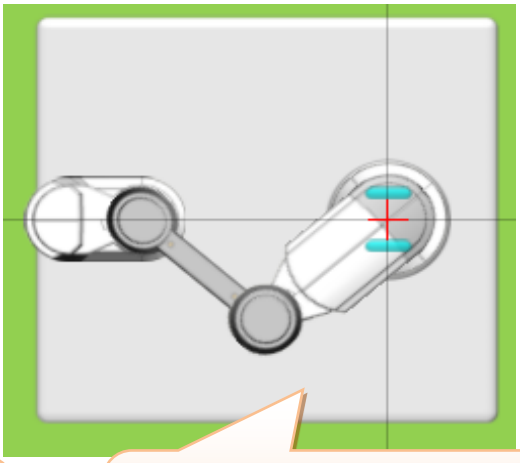


【デフォルトの設定】
BASE_ANGLE=180
BASE_OFFSET_X=+0
BASE_OFFSET_Y=+0

マクロの設定で説明した BASE_ANGLE、BASE_OFFSET_X/Y の各マクロを書き換えることで、座標系を変更できます。ただし、座標系を変更すると、対応するアームの角度や可動範囲が変わるため、過去に作成したプログラムが意図通りに動作しなくなる場合がありますのでご注意ください。



BASE_ANGLE=90
BASE_OFFSET_X=+0
BASE_OFFSET_Y=-50



BASE_ANGLE=0
BASE_OFFSET_X=-60
BASE_OFFSET_Y=+0

モータに関する資料

ロボット本体を制御するための通信コマンドは、ロボット本体に組み込まれているシリアルサーボモータ「RS304MD」（双葉電子工業社製）の仕様に従います。このサーボモータに関する通信コマンド・メモリマップ等の資料は、双葉電子工業の web ページで公開しております。モータのメモリマップのダンプなど、サンプルソースに含まれていない操作を行う場合は、この資料をご参照ください。

■双葉電子工業「RS304MD」製品ページ URL

https://futaba.co.jp/robot/command_type_servos/rs304md.html

ご質問について

開発環境の導入やサンプルソースの具体的な処理・改造などに関するご質問につきましては、原則として承っておりません。これらに関する情報は、資料中で紹介している各種資料や web 上に公開している情報を参考に、各自ご確認ください。

■オプションパーツ、関連商品のご購入は・・・

No.1 の品揃え！　各種オプションパーツ、ロボット関連製品のご購入はコチラ

http://www.vstone.co.jp/robotshop/

商品に関するお問い合わせ

商品の技術的なご質問は、問題・症状・ご使用の環境などを記載の上メールにてお問い合わせください。

E-mail: **infodesk@vstone.co.jp**　　受付時間　：10:00～17:00（土日祝日は除く）

ヴイストン株式会社

〒555−0012　大阪市西淀川区御幣島 2-15-28

TEL：06-4808-8701　FAX：06-4808-8702

Vstone®

www.vstone.co.jp