Multi LABELIST Component
テクニックマニュアル

株式会社サトー

2022年11月25日

はじめに

この度は、「Multi LABELIST Component」(以下 MLComponent)をご利用いただき誠にありがとうございます。MLComponent は、弊社製汎用ラベル・タグ発行ソフトウェア「Multi LABELIST V6」(以下 MLV6)の資産を利用し、お客様のアプリケーションにラベル・タグ発行機能を追加するために開発した.NET コンポーネントです。

MLV6 で作成したレイアウトファイルをもとに、自由度の高いラベル／タグ発行システムを構築していただくために MLV6 の一部の機能は省かせていただきましたが、USB、LAN、COM(シリアルポート)、および弊社製プリンタドライバと、出力デバイスを問わない設計が可能です。ステータス監視機能をサポートしており、プリンタがどのような状態になっているか取得することができます。

本マニュアルでは MLComponent をご理解いただくための様々な利用方法をご説明しております。
プロパティ・メソッド毎の詳細な説明は、「MLComponent リファレンスマニュアル」、
ML 製品を初めてご利用する方は「MLComponent 練習マニュアル」ご参照ください。

ご注意

- 本マニュアルの一部または全部を弊社の許可なく複写・複製することは、その形態を問わず禁じます。
- 本マニュアルの内容は、訂正・改善のため予告なく変更することがあります。
- 本マニュアルを運用した結果の影響については責任を負いかねますのでご了承下さい。
- 本マニュアルの内容については万全を期しておりますが、万一ご不審な点やお気づきの点がございましたら、弊社までご連絡ください。

- SATO、Multi LABELIST は、サトーホールディングス株式会社の登録商標または商標です。
- Microsoft、Windows は、米国マイクロソフト社の登録商標です。
- その他記載されている会社名、製品名は各社の登録商標または商標です。

目次

第１章

基本編

1-1 Visual Studio で利用する

■参照の追加　■宣言

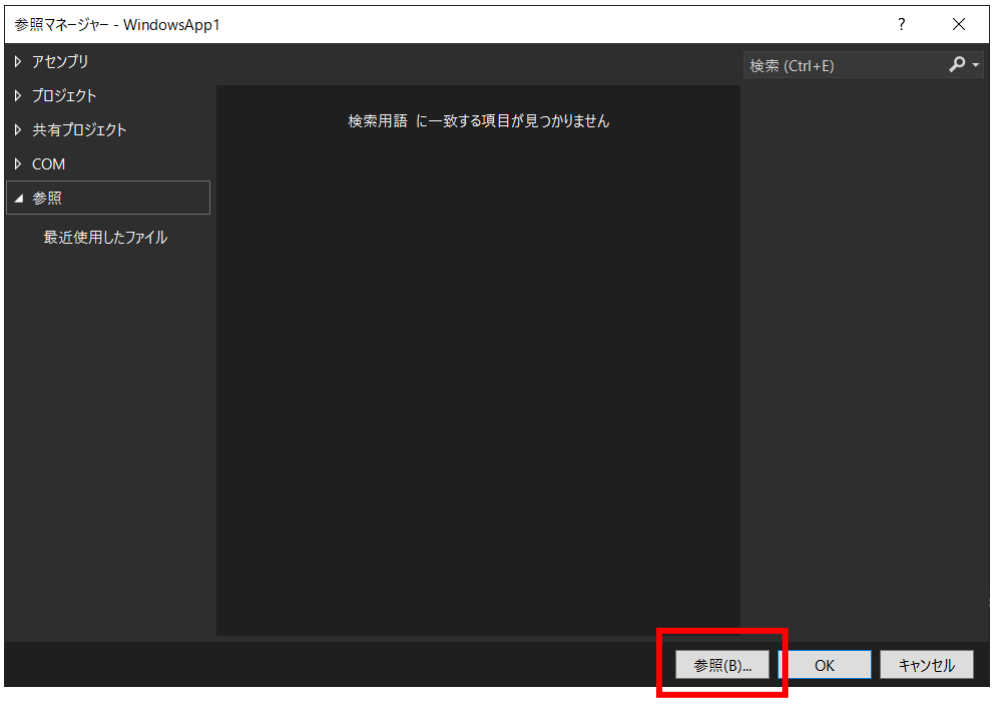
MLComponent を Visual Studio で利用するには、参照の追加を行い、コード上でインスタンスを生成します。

■Visual Studio 2022 での利用手順

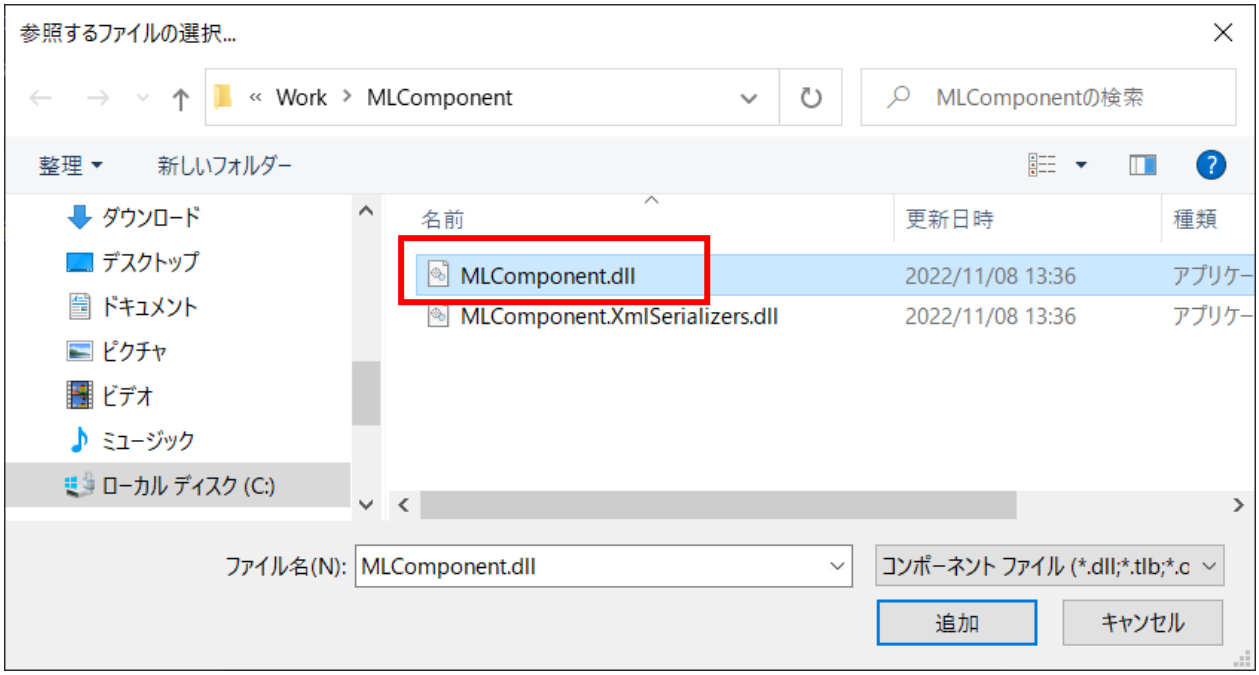
「Windows フォーム アプリケーション（.NET Framework）」でプロジェクトを作成します。
フレームワークは「.NET Framework 4.8」を選択します。

「プロジェクト」＞「参照の追加」を選択します。

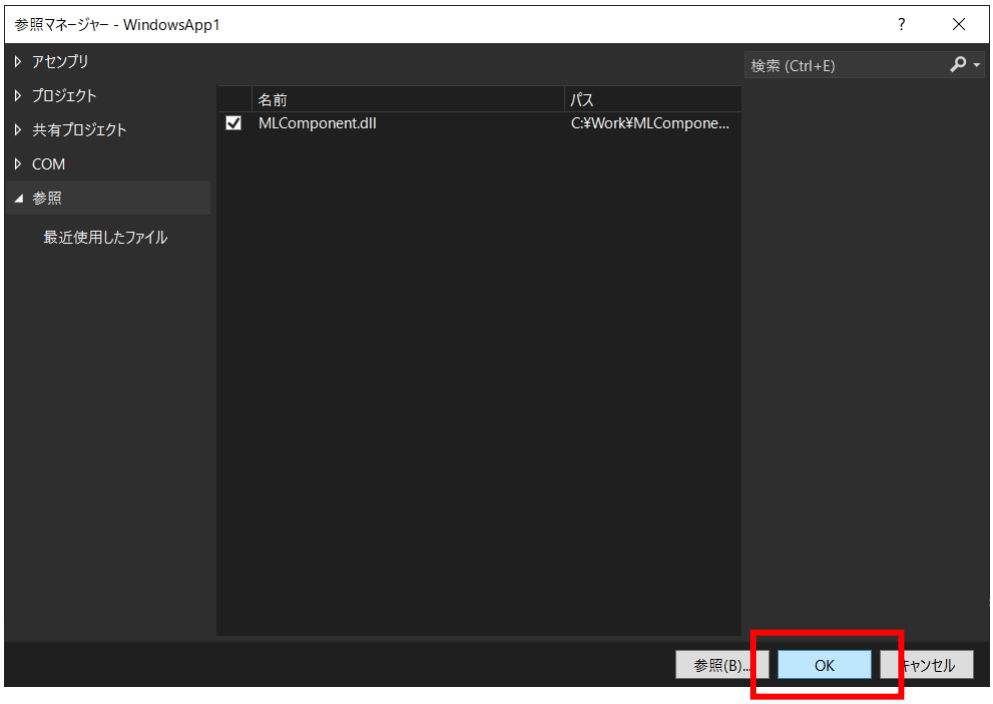
「参照」を押下します。



「MLComponent.dll」を選択して「追加」を押下します。



「参照」に「MLComponent.dll」が追加されたことを確認したら、「OK」を押下します。



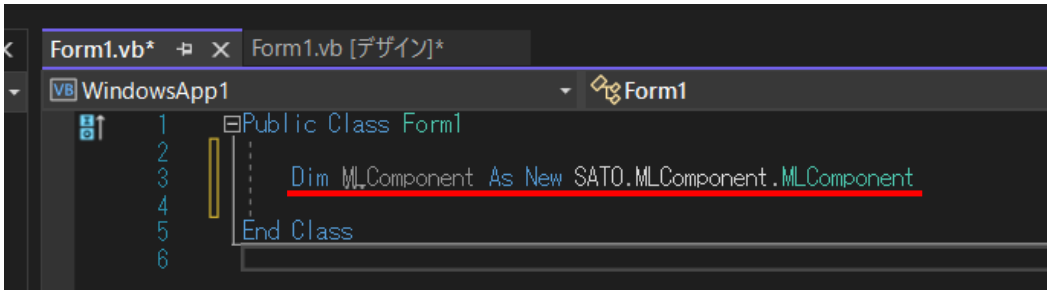
解説：参照設定

ソリューション エクスプローラーで参照設定を開くとアプリケーションで参照しているライブラリが表示されます。MLComponent も参照の追加を行うと、アセンブリとして一覧に追加されていることが確認できます。

■インスタンス生成

フォーム上で右クリックし、「コードの表示」を選択します。

MLComponent のインスタンスを生成します。



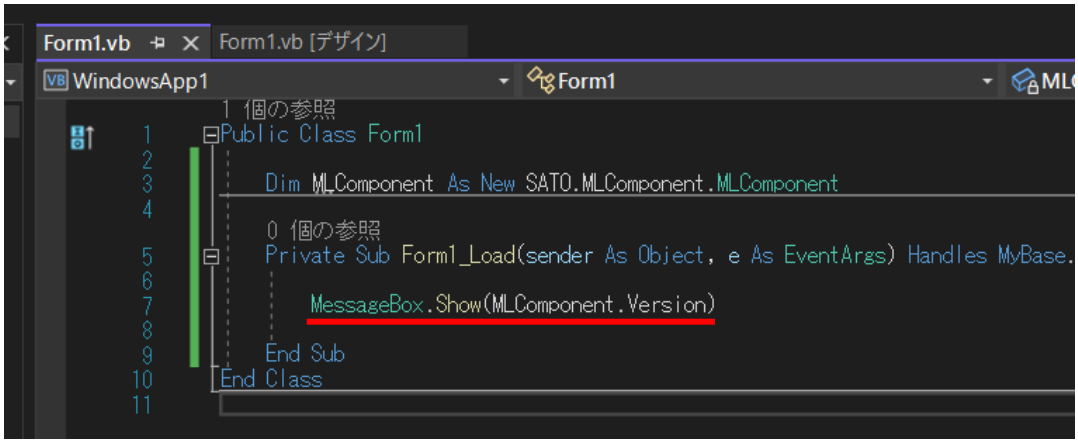
記述例

```
Dim MLComponent As New SATO.MLComponent.MLComponent
```

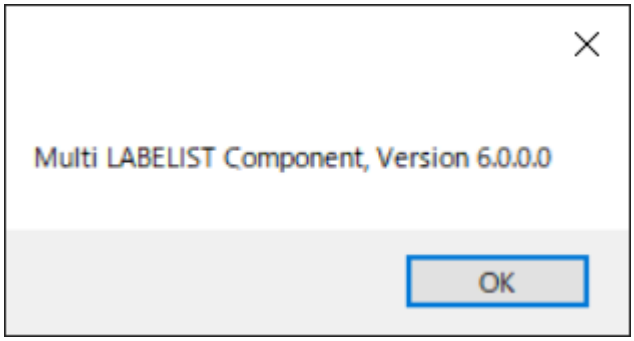
インスタンス名

クラス名

MLComponent のインスタンスが正しく生成できたか確認します。フォームロード時にダイアログでバージョン情報（Version プロパティの値）を表示します。



デバッグを開始して、以下のようにダイアログが表示されれば成功です。



1-2 アプリケーションを配布する

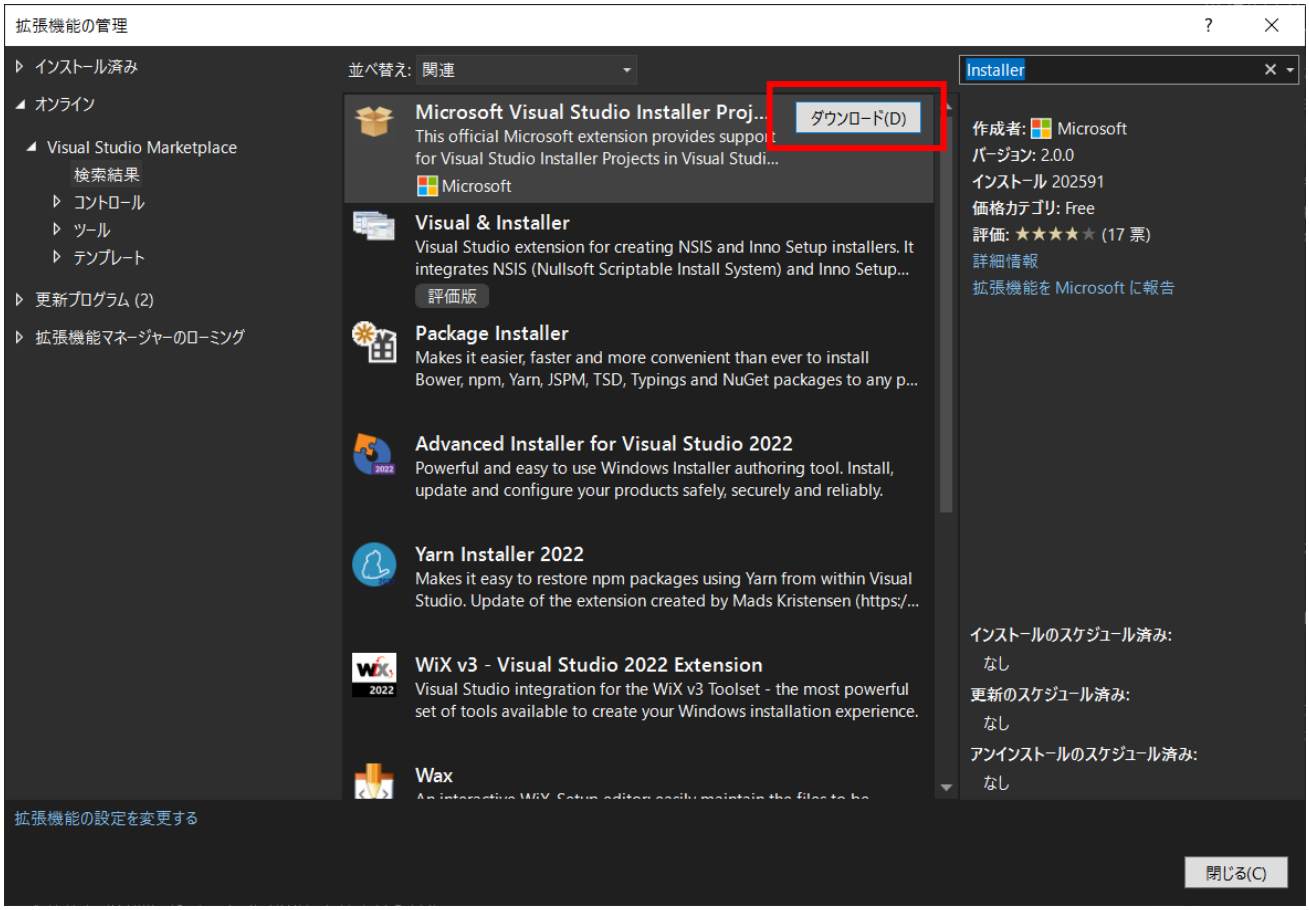
■インストーラ

MLComponent はアプリケーションと共に配布してください。実行ファイル（*.exe）と同じ場所に「MLComponent.dll」と「MLComponent.XmlSerializers.dll」をコピーすることで動作します。ここでは、Visual Studio 2022 のセットアッププロジェクトでインストーラを作成して配布する方法を説明します。

■Visual Studio 2022 でインストーラを作成する

「拡張機能」＞「拡張機能の管理」を選択します。

検索ボックスに「Installer」と入力して、「Microsoft Visual Studio Installer Projects 2022」を検索し、「ダウンロード」を選択します。

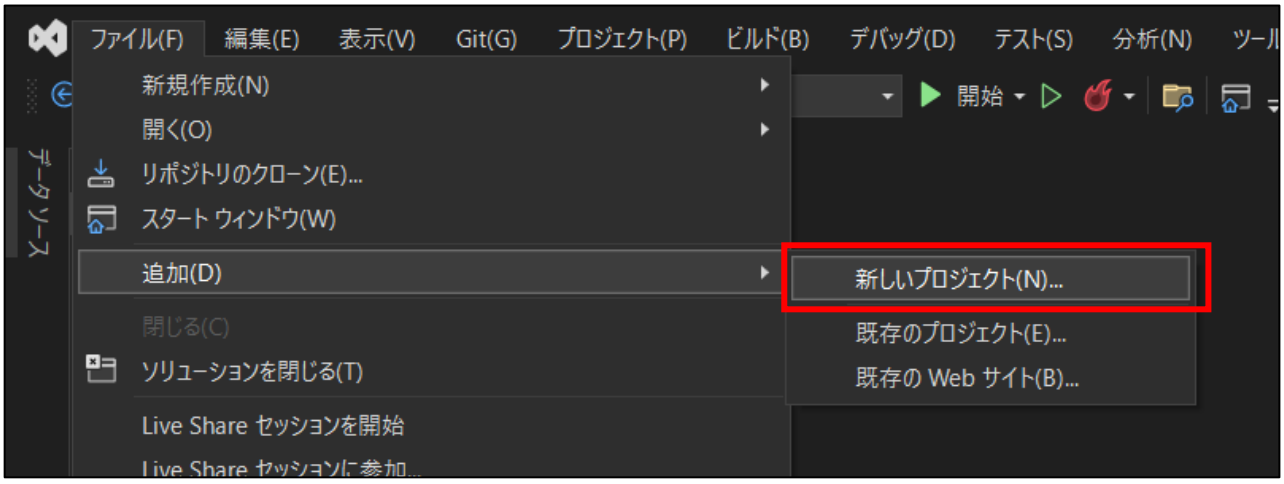


画面を閉じたあと、Visual Studio も終了します。

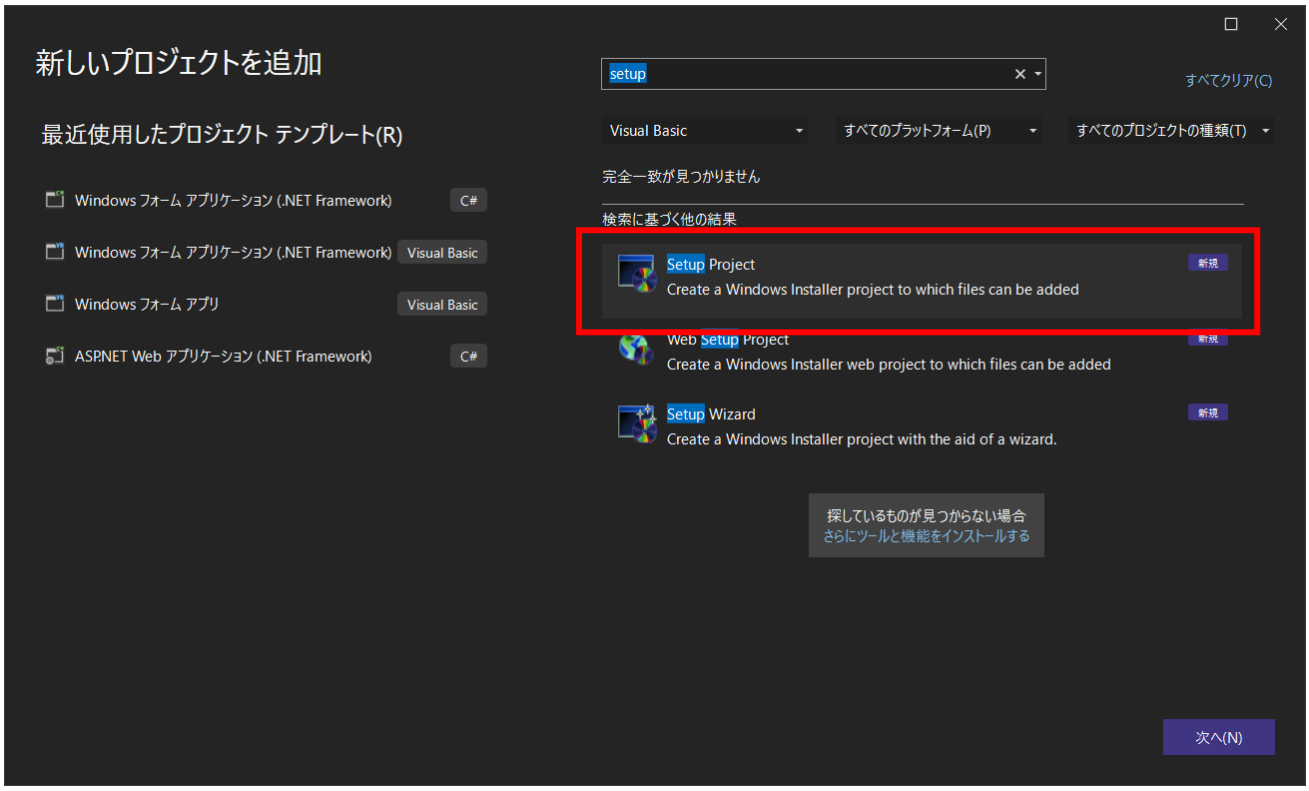
画面の指示に従って拡張機能をインストールします。

インストールが完了したら、再度プロジェクトを開きます。

「ファイル」＞「追加」＞「新しいプロジェクト」を選択します。

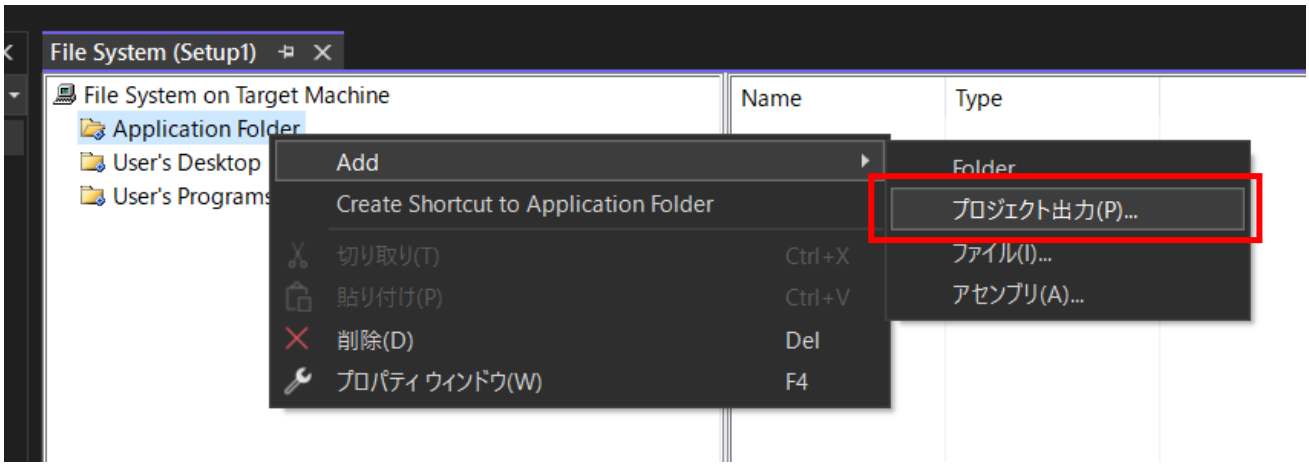


検索ボックスに「setup」と入力、「Setup Project」を選択して「次へ」を選択します。

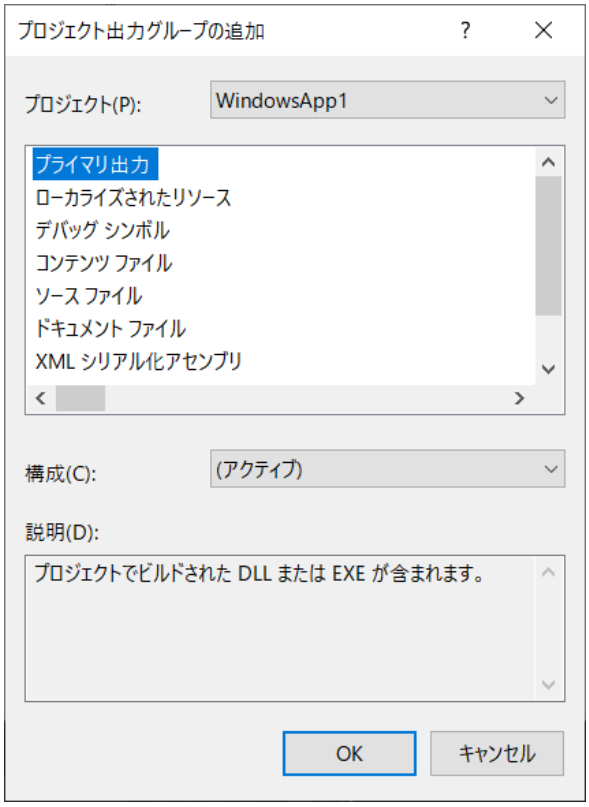


プロジェクト名を入力して「作成」を選択します。
セットアッププロジェクトが作成されます。

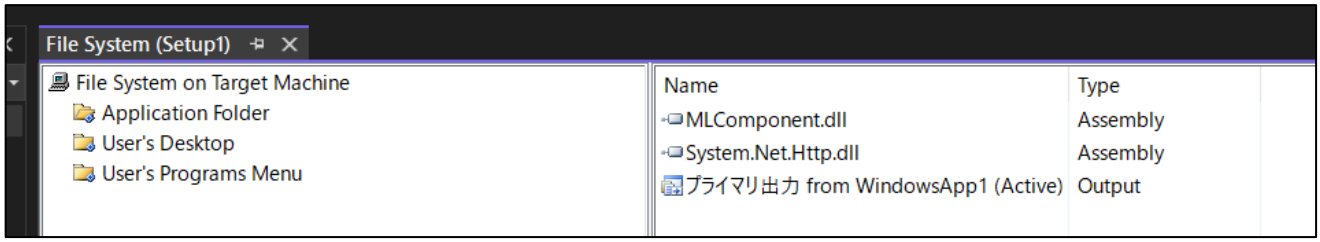
「Application Folder」を右クリックし、「Add」＞「プロジェクト出力」を選択します。



「OK」を選択します。



「MLComponent.dll」が追加されます。



1-3 発行方法を決める

■発行方法

ラベル発行は PC とプリンタをケーブルで接続し、プリンタに発行データを送信して行います。発行方法には大きく分けて「インターフェース出力」と「プリンタドライバ出力」の２つの方法があります。メリットやデメリットから運用方法に合う発行方法を選択してください。

インターフェース出力の使い方は「プリンタに接続する」を、
プリンタドライバ出力の使い方は「プリンタドライバを利用する」をご確認ください。

■発行方法

	プリンタドライバ出力	インターフェース出力
利用可能な インターフェース	LAN（無線 LAN） USB RS-232C	LAN（無線 LAN） USB RS-232C Bluetooth
開発レベル	初級	中級～上級
プリンタの制御	不要 （プリンタドライバにおまかせ）	必要
メリット	通信制御はプリンタドライバが行うので、アプリケーションの開発が簡単。	プリンタの状態を確認できる。 プリンタドライバのインストール不要。
デメリット	各 PC にプリンタドライバのインストールが必要。 アプリからプリンタの状態は確認できない。	プリンタとの通信制御の実装が必要。通信に関わる技術やプリンタの仕様を理解した上で、プリンタのエラーや通信障害など様々な状況応じた処理を作り込む必要がある。

1-4 プリンタに接続する

■接続・切断　■USB　■LAN　■COM ■Bluetooth

ラベル発行するにはまずプリンタに接続します。また、発行後は必ず最後に切断を行います。切断を忘れると、再度発行するときにプリンタと接続できなかったり、他の PC から発行ができなかったりといったトラブルになります。

プリンタドライバを利用して簡単に発行したい場合は、「プリンタドライバを利用する」をご確認ください。

■接続・切断

・サンプルコード

```
'MLComponentのインスタンス生成
Dim MLComponent As New SATO.MLComponent.MLComponent

'処理結果
Dim Result As Integer

'通信設定（LAN接続）
MLComponent.Setting = "LAN:192.168.1.100,1024"

'プリンタと接続
Result = MLComponent.OpenPort(1)
If Result <> 0 Then
        MessageBox.Show("OpenPortError No." & Result.ToString)
End If

'プリンタと切断
MLComponent.ClosePort()
```

■USB で接続する

接続するプリンタが 1 台の場合は「USB:」と指定してください。最初に見つかったサトー製 USB プリンタに自動的に接続します。PC に 2 台以上 USB プリンタを接続する場合は、「USB:」の後にプリンタの USB シリアル ID を指定してください。USB シリアル ID の確認方法は「リファレンスマニュアル」をご確認ください。

サンプルコード

```
'PCに接続されたプリンタに接続
MLComponent.Setting = "USB:"

'PCに接続されたシリアルNo.0000T123に接続"
MLComponent.Setting = "USB:0000T123"
```

■LAN で接続する

LAN ではプリンタの**[IP アドレス]**と**[ポート番号]**を指定します。
ポート番号には通常「1024」もしくは「9100」を指定してください。

サンプルコード

```
'IPアドレス192.168.1.10、ポート番号1024を使用してプリンタに接続
MLComponent.Setting = "LAN:192.168.1.10,1024"

'IPアドレス192.168.1.10、ポート番号9100を使用してプリンタに接続
MLComponent.Setting = "LAN:192.168.1.10,9100"
```

■COM（シリアルポート）で接続する

COM ではプリンタの**[COM ポート番号]**と**[ボーレート]**、**[パリティビット]**、**[データビット]**、**[ストップビット]**を指定します。プリンタの設定値に合わせてください。

サンプルコード

```
'COM19に接続されたプリンタに接続
MLComponent.Setting = "COM19:115200,n,8,1"
```

■Bluetooth で接続する

Bluetooth ではプリンタの**[BD アドレス]**を指定します。
接続可能なデバイスは、ペアリング済みのプリンタまたは Bluetooth Ver.3.0 対応で認証レベルが認証無しに設定されているプリンタです。Bluetooth のペアリングは AuthenticateBluetoothDevice メソッドまたは Windows 標準の設定画面で行ってください。

サンプルコード

```
'BDアドレス000b5db4aebbのプリンタに接続
MLComponent.Setting = "BT:000b5db4aebb"
```

1-5 プリンタの状態を確認する

■通信プロトコル　■状態確認

プリンタドライバを利用せずインターフェース出力で発行する場合は、必ず発行前と発行後にプリンタの状態を確認してください。確認を行わないと、プリンタでエラーが発生していても印字データを送信してしまい、ラベルが発行されずにデータが消失してしまうトラブルになる可能性があります。

■通信プロトコル

プリンタと通信するためには**[通信プロトコル]**を設定します。プリンタの設定値と合わせてください。

通信プロトコル	インターフェース			
	USB	LAN	Bluetooth	COM
ステータス３	×	○	○	○
ステータス４	○	○	○	○

○：利用可能　×：利用不可

■状態確認

・サンプルコード

```
'MLComponentのインスタンス生成
Dim MLComponent As New SATO.MLComponent.MLComponent

'処理結果
Dim Result As Integer

'ステータス文字列
Dim Status As String = ""

'通信設定（LAN接続）
MLComponent.Setting = "LAN:192.168.1.100,1024"

'通信プロトコル設定
MLComponent.Protocol = SATO.MLComponent.Protocols.Status3

'プリンタと接続
Result = MLComponent.OpenPort(1)
If Result <> 0 Then
        MessageBox.Show("OpenPortError No." & Result.ToString)
        Exit Sub
End If

'プリンタの状態確認
Result = MLComponent.GetStatus(Status)
If Result = 0 Then
        MessageBox.Show("PrinterStatus = " & Status.Substring(2, 1))
Else
```

```
            MessageBox.Show("OpenPortError No." & Result.ToString)
End If

 'プリンタと切断
MLComponent.ClosePort()
```

プリンタの状態は、取得したステータス文字列の３桁目で判断できます。
ステータスの詳細と送信可否の判断は「リファレンスマニュアル」のステータス仕様をご確認ください。

例として、「A」を取得した場合は、プリンタはオンライン状態でエラーも発生していないため、問題なく発行できます。「I」を取得した場合は、プリンタがオンライン状態でラベルを発行中ですが、受信バッファの容量が少ないため、数秒待機してから再確認を行うことが望ましいです。「g」を取得した場合は、プリンタヘッドが断線して正常に発行できないため、発行を行わずプリンタ交換を促す注意メッセージを表示するといった対応を取ることができます。

1-6 プリンタドライバを利用する

■接続・切断

細かくプリンタの状態を監視したり、１ 枚１枚チェックしながら発行する必要がなく、簡単にラベル発行したい場合は、プリンタドライバの利用がおすすめです。プリンタドライバに対してデータを出力するだけで、面倒な通信制御は必要ありません。
細かな制御が必要な場合は、「プリンタに接続する」をご確認ください。

■接続・切断

プリンタドライバは Setting プロパティで「DRV:」で指定します。オプションとして**[プリンタドライバ名]**を指定します。ラベル発行するには、まずプリンタドライバへ接続（出力開始）します。発行完了後は必ず切断（出力完了）を行います。

・サンプルコード

```
'MLComponentのインスタンス生成
Dim MLComponent As New SATO.MLComponent.MLComponent

'処理結果
Dim Result As Integer

'通信設定（プリンタドライバ）
MLComponent.Setting = "DRV:SATO CL4NX-J 305dpi"

'プリンタドライバへ出力開始
Result = MLComponent.OpenPort(1)
If Result <> 0 Then
        MessageBox.Show("OpenPortError No." & Result.ToString)
End If

'プリンタドライバへ出力完了
MLComponent.ClosePort()
```

プリンタドライバは、出力が開始される（接続）とジョブという発行するための箱を作り、発行データを 1 データずつ蓄積していきます。データ出力中にプリンタへ送信するタイミングは、OS により異なります。すぐにプリンタにデータを送信するには、Output メソッド直後に ClosePort を実行してください。

1-7 ラベル・タグを発行する

■ラベル発行

ラベル発行は Multi LABELIST V6 で作成したレイアウトファイルと入力データを組み合わせて行います。[発行枚数]は必ず必要な入力データで、[発行枚数]が入力されていない場合は発行時にエラーとなります。

■ラベル発行

発行する前に、**[レイアウトファイルのパス名]**と**[入力データ]**を指定します。入力データは様々な指定方法が利用可能です。詳しくは「データを入力する」をご確認ください。

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥label.mllayx"

'入力データ指定(入力変数が[発行枚数]のみの場合)
MLComponent.PrnData = "10"

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
End If
```

ワンポイントテクニック

同一データのラベルを複数枚発行する場合、「発行枚数=1」を複数送信するのではなく、
「発行枚数=N」（例：10 枚の場合は発行枚数=10）を 1 回送信する方が高速にラベル発行できます。

1-8 データを一括で入力する

■入力順位　■データ形式　■複数データ

指定したデータ形式に従って、一括でデータを入力します。プリンタドライバを利用する場合は、1つのデータだけでなく、複数のデータを一括して入力することもできます。
順番を意識せず変数名で入力したい場合は「データを変数名で指定して入力する」をご確認ください。

■入力順位

まずデータを入力する順番を、MLデザインの入力定義で**[入力順位]**を表示させて確認します。

・サンプルコード

```
'入力データ指定
'商品名、商品コード、シリアルNo.、発行枚数を順番に入力
MLComponent.PrnData = "ラベルプリンタ" & vbTab & "490310999999" & vbTab & "RX00007802" & vbTab & "3"
```

■データ形式

デフォルトではタブ区切り（TSV形式）でデータを指定しますが、カンマ区切りのCSV形式やスペース区切りのPRN形式でデータを指定することもできます。

・サンプルコード（CSV 形式）

```
 'データ形式指定(CSV形式)
MLComponent.PrnDataType = SATO.MLComponent.PrnDataTypes.Csv

'入力データ指定
MLComponent.PrnData = "ラベルプリンタ,490310999999,RX00007802,3"

'括り文字""（ダブルクォーテーション）を使って改行コードも入力可能
MLComponent.PrnData = """このプリンタは、" & vbCrLf & "高速発行が可能な4インチ堅牢型プリンタです。"",1"
```

・サンプルコード（PRN 形式）

```
'データ形式指定(PRN形式)
MLComponent.PrnDataType = SATO.MLComponent.PrnDataTypes.Prn

'入力データ指定
MLComponent.PrnData = "ラベルプリンタ 490310999999 RX00007802 3"
```

■複数のデータを入力する

プリンタドライバを利用する場合は、複数のデータを一括で入力する方法が利用できます。

・サンプルコード

```
'前提条件としてプリンタドライバを利用すること

'文字列型の配列を作成
Dim inputData(0 To 2) As String
inputData(0) = "ラベルプリンタA,490310999999,RX00007802,1"
inputData(1) = "ラベルプリンタB,490310123456,MS00000619,1"
inputData(2) = "ラベルプリンタC,490310000005,FX00000550,1"

'入力データの複数指定
MLComponent.SetPrnDataArray(inputData)
```

1-9 データを変数名で指定して入力する

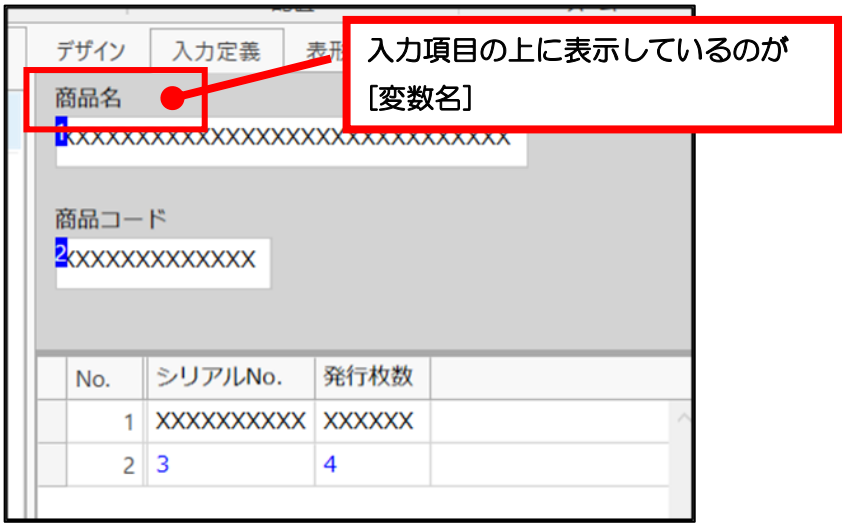
■変数名

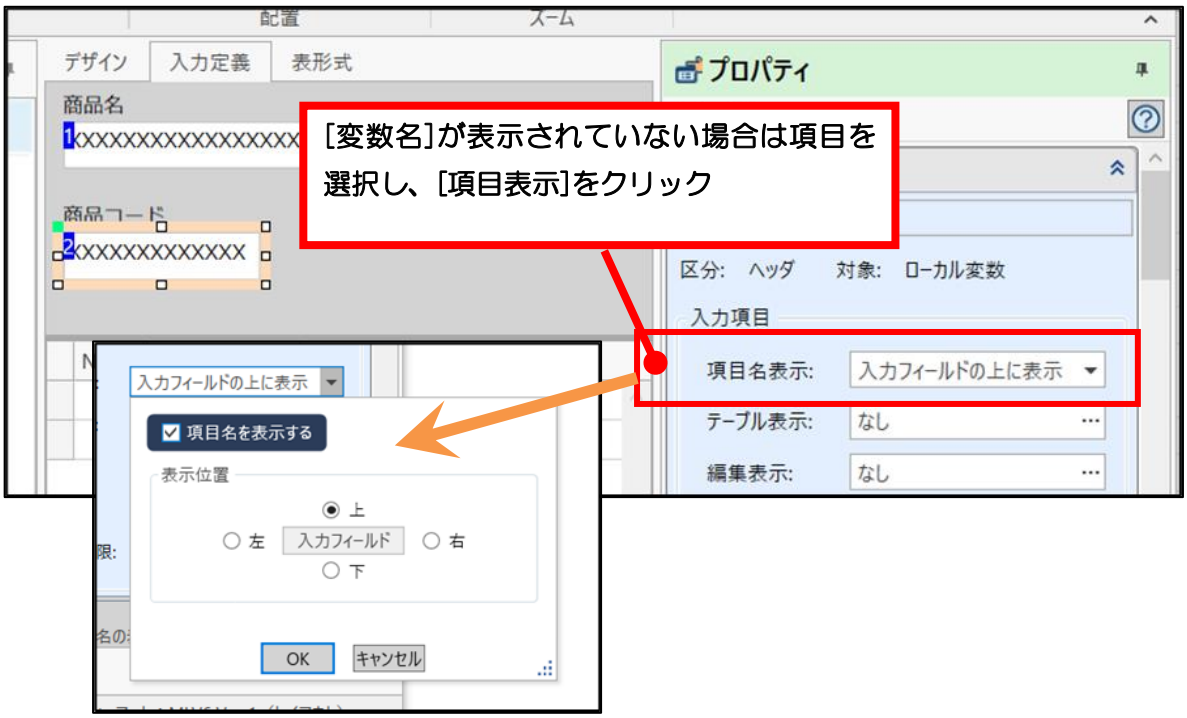
変数名を指定してデータを入力します。変数名が共通しているレイアウトファイルを複数使う場合や入力順番が変更される可能性がある場合など、レイアウトファイルの入力順位を意識せずにデータを入力できます。

変数名を意識せず入力順位で簡単に入力したい場合は「データを一括で入力する」をご確認ください。

■変数名

まずデータ指定に必要な**[変数名]**を、MLデザインの入力定義で確認します。





・サンプルコード

```
'「商品名」を入力
MLComponent.SetPrnDataField("商品名", "ラベルプリンタ")

'「商品コード」を入力
MLComponent.SetPrnDataField("商品コード", "4903109999999")

'「シリアルNo.」を入力
MLComponent.SetPrnDataField("シリアルNo.", "RX00007802")

'「発行枚数」を入力
MLComponent.SetPrnDataField("発行枚数", "3")
```

1-10 バージョンを確認する

■プロパティ　■ファイルバージョン

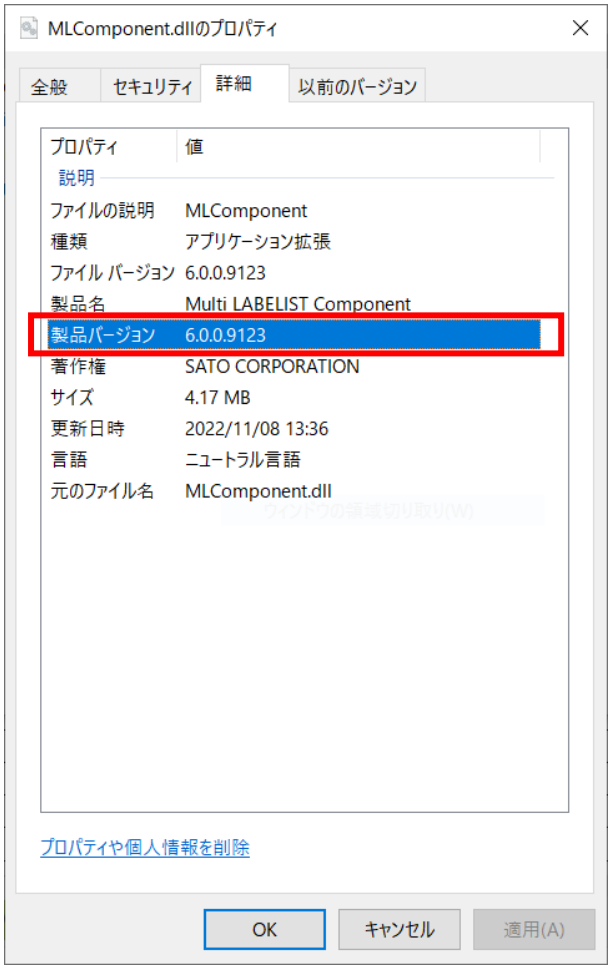
■Version プロパティで取得する

Version プロパティで取得できる情報からバージョン番号を確認できます。

Multi LABELIST Component, Version 6.x.x.x

■ファイルのプロパティで確認する

MLComponent の本体である「MLComponent.dll」のファイルプロパティで確認できます。



1-11 バージョンアップを行う

■開発環境　■実行環境

■開発環境の MLComponent を更新する

「1-1. Visual Studio で利用する」で新しいバージョンの「MLComponent.dll」を参照の追加で選択することで更新されます。

■実行環境の MLComponent を更新する

実行アプリケーションと共に配布した「MLComponent.dll」を新しいバージョンのファイルに入れ替えるだけで更新可能です。開発環境でのリビルド・リコンパイルは必要ありません。

1-12 Microsoft Excel/Access(VBA)で利用する

■参照の追加　■宣言

■インストール

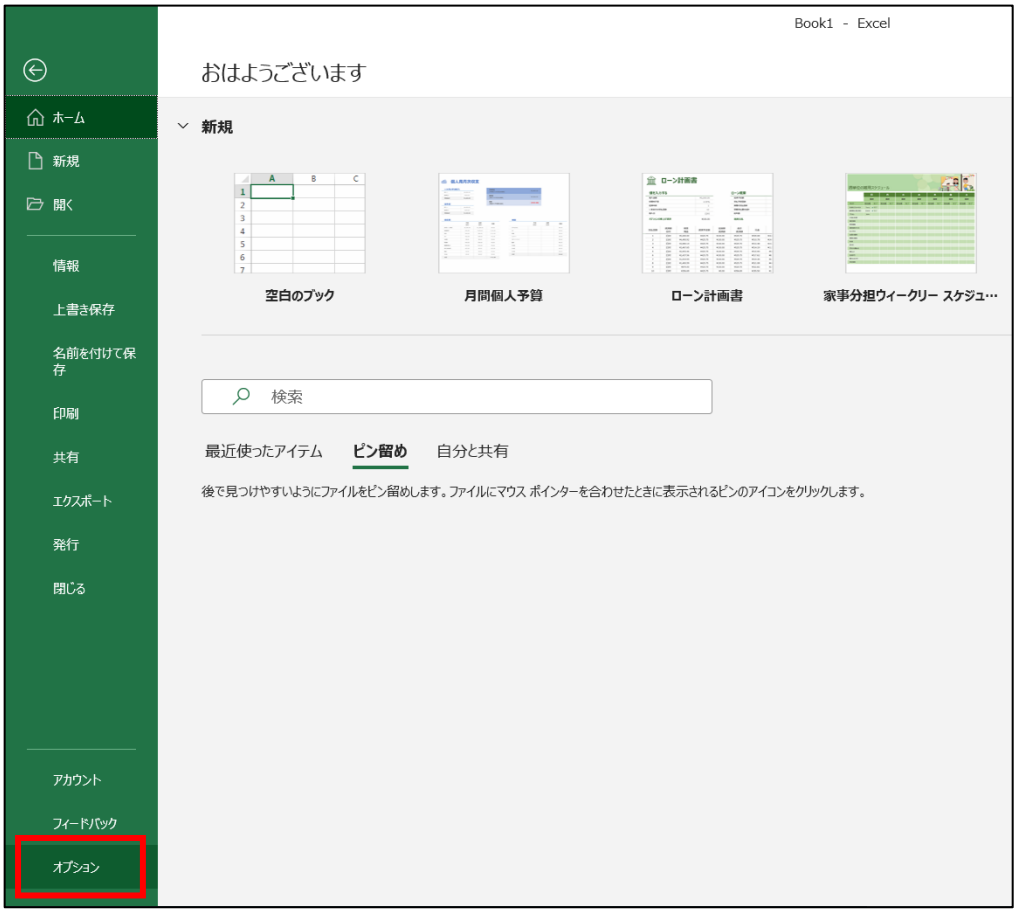
Microsoft Excel/Access(VBA)で利用するには、インストーラを利用して MLComponent を発行環境にインストールしてください。インストーラは以下のページからダウンロードしてください。

https://www.sato.co.jp/support/printertool/tool/multi-labelist-component/

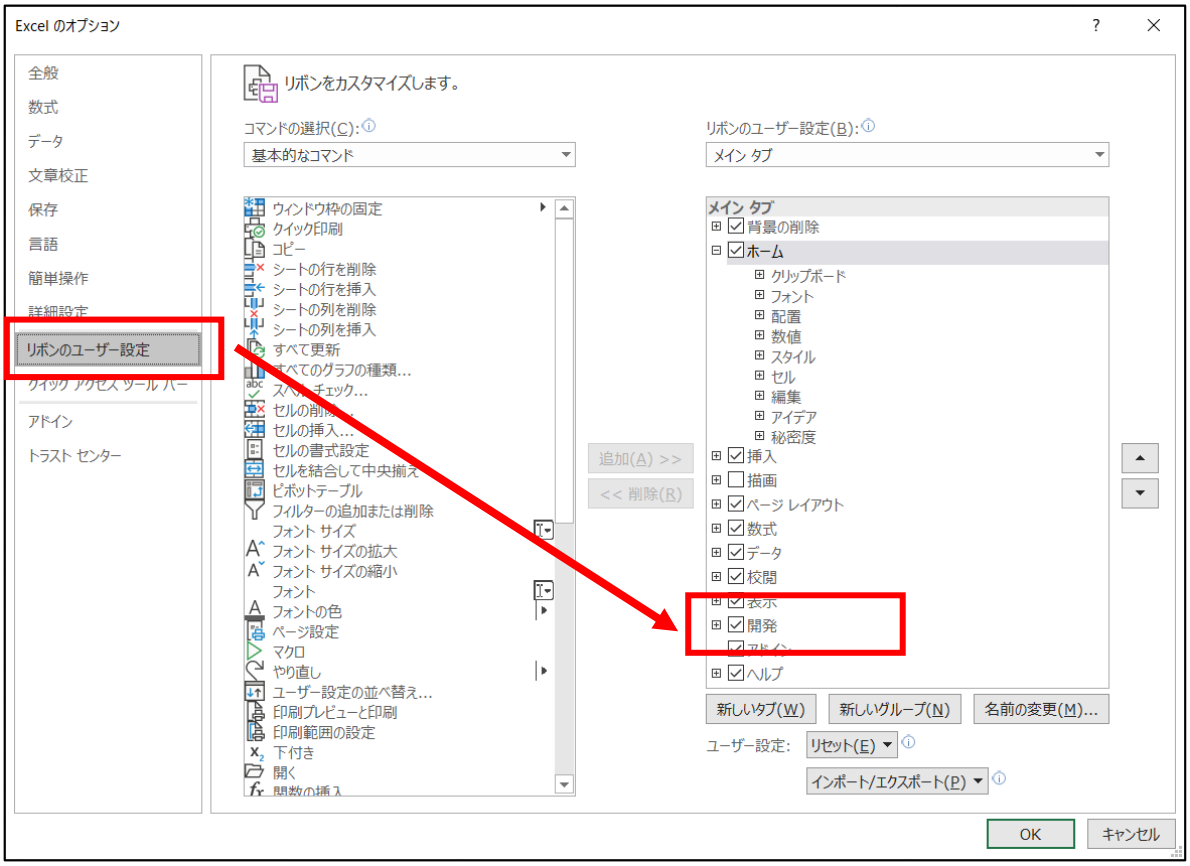
■参照の追加

Excel の場合は、参照の追加を行うためにリボンに「開発」タブを表示させます。

「ファイル」メニューから「オプション」を選択します。

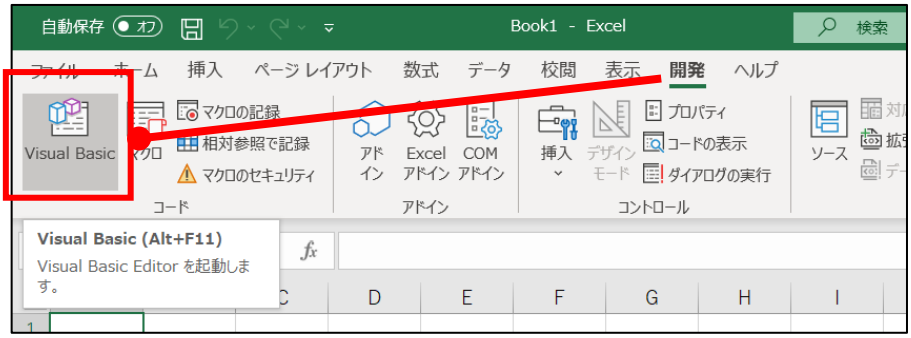


「リボンのユーザー設定」を選び、「開発」にチェックを付けて「OK」を選択します。

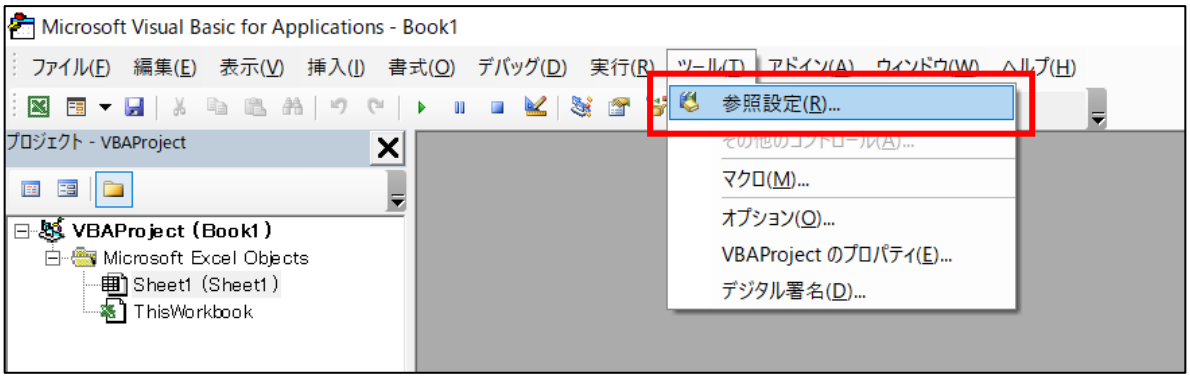


「開発」タブが表示されたら、参照設定を行います。

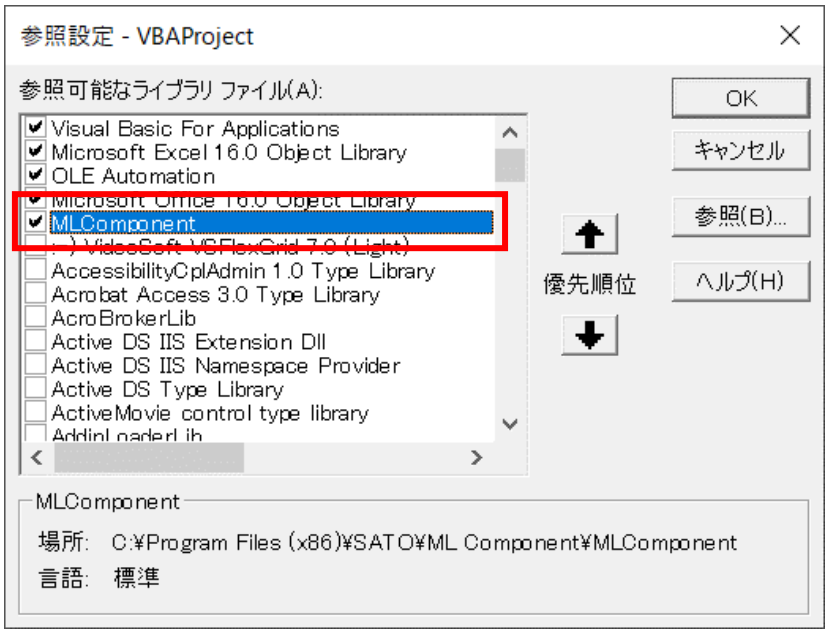
「開発」タブ>「Visual Basic」を選択します。(Access は「データベースツール」タブ>「Visual Basic」)



「ツール」>「参照設定」を選択します。

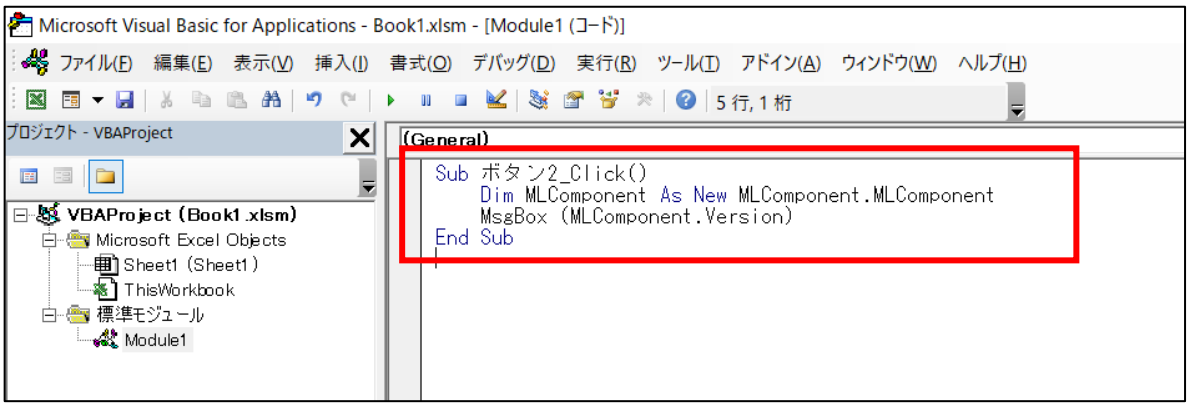


「MLComponent」にチェックを付けて「OK」を選択します。

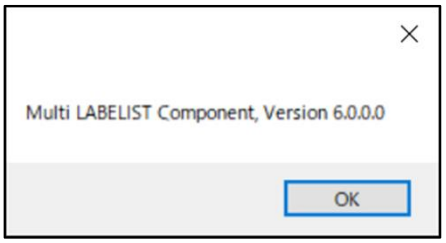


■インスタンス生成

コード上にボタンをクリックすると MLComponent のバージョンをダイアログで表示する処理を記述します。



実行すると、以下のダイアログが表示されます。



第２章

応用編

2-1 プリンタの濃度・速度を変更する

■濃度　■速度

プリンタの濃度と速度は、通常プリンタ本体で調整しますが、複数種類のラベルを付け替えて発行する運用で、共通の設定で対応できない場合に、レイアウト毎にMLComponentで調整してご利用ください。

濃度と速度共に、3つの指定方法があります。

①プリンタ本体の濃度・速度を使用する
②MLComponentで指定した値を使用する
③レイアウトファイルに設定された値を使用する

■濃度を変更する

・サンプルコード

```
'プリンタ本体の設定を使用
MLComponent.Darkness = ""

'MLComponentで指定した値を使用
MLComponent.Darkness = 3

'レイアウト設定を使用
MLComponent.Darkness = "S"
```

■速度を変更する

・サンプルコード

```
'プリンタ本体の設定を使用
MLComponent.Speed = ""

'MLComponentで指定した値を使用
MLComponent.Speed = 3

'レイアウト設定を使用
MLComponent.Speed = "S"
```

2-2 印字位置を調整する

■印字位置

ラベルに印字される位置を全体的に微調整します。位置が大きく異なる場合は、MLデザインで用紙サイズやオブジェクトの位置を変更してください。

２つの指定方法があります。

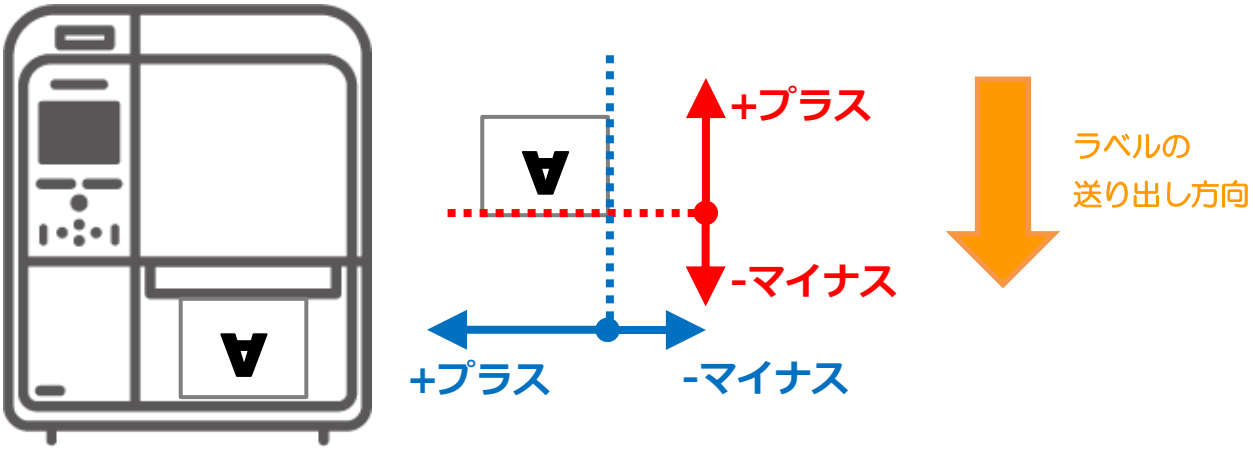
①MLComponentで指定した値を使用する

②レイアウトファイルに設定された値を使用する

■印字位置を調整する

印字位置は、ラベルの送り出し方向が縦位置のマイナスで、ラベルの内側が横位置のプラスです。

値はmm単位で小数点第2位まで調整可能です。



・サンプルコード

```
'MLComponentで指定した値で調整
MLComponent.Offset = "1.5,3.25"

'レイアウト設定を使用
MLComponent.Offset = "S,S"
```

2-3 消費税を設定する

■税編集　■消費税

MLV6の変数設定で[編集パラメータ]の[税編集]を使用している場合に、消費税をTaxRateプロパティで設定する必要があります。消費税の設定がないと発行時にOutputメソッドでエラー413（税編集時にエラーが発生しました。）が返送されます。

■税編集

変数設定で[編集パラメータ]の[税編集]が[なし]の場合、[税編集]が[あり]で[税率]が[固定値を使用する]または[変数値を使用する]の場合は、TaxRate プロパティを設定する必要はありません。

■消費税

・サンプルコード

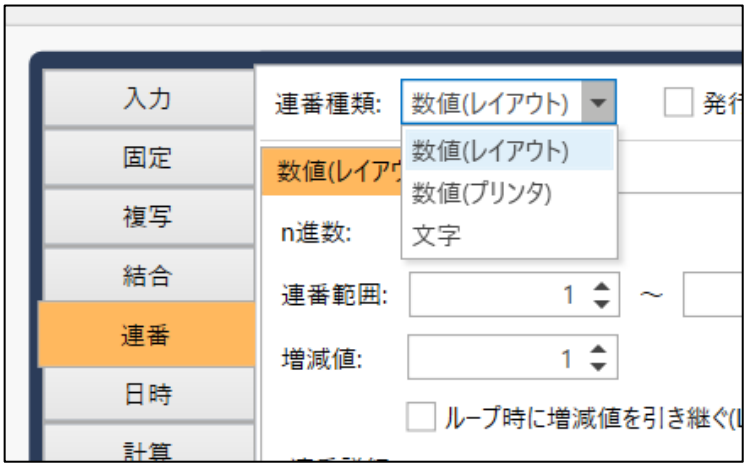
```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥price.mllayx"

'入力データ指定（価格"100"を入力する場合）
MLComponent.PrnData =  "緑茶" & vbTab & "100" & vbTab & "2"

'消費税指定
MLComponent.TaxRate = "10"

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
End If
```

2-4 連番を印字する

■連番　■初期値

商品のシリアルNo.やラベルの識別で利用できる連番を印字できます。

連番は、MLV6の連番変数を使います。通常はレイアウト情報を使う「数値（レイアウト）」を利用します。連番値の保存が必要なくプリンタフォントで印字する場合は「数値（プリンタ）」、特殊なパターンや文字で連番させる場合は「文字」も選択可能です。



■連番を印字する

連番を印字するために、MLComponentで特別な操作は必要ありません。

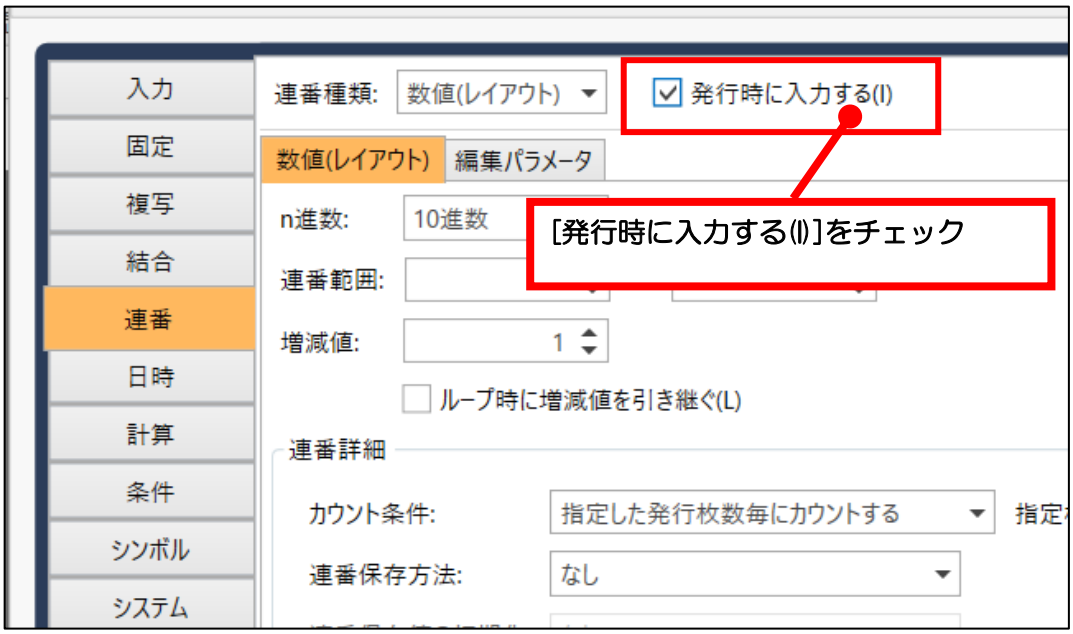
・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'連番変数を利用したレイアウトファイルを指定
MLComponent.LayoutFile = "C:¥sato¥count.mllayx"

'入力データ指定(入力変数が[発行枚数]のみの場合)
MLComponent.PrnData = "10"

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
End If
```

■連番の初期値を入力する

連番の開始値や終了値をアプリケーション側で管理する場合は、発行する度に連番の初期値をデータとして入力できるようにします。

連番連数の設定画面で[発行時に入力する(I)]にチェックを入れてください。



・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'連番変数を利用したレイアウトファイルを指定
MLComponent.LayoutFile = "C:¥sato¥count.mllayx"

'入力データ指定（初期値"310"から3枚発行する場合）
MLComponent.PrnData =  "小麦粉" & vbTab & "4903101234567" & vbTab & "310" & vbTab & "3"

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
End If
```

2-5 ヘッダ・テール札を発行する

■ヘッダ・テール札

レイアウトファイルに設定されたヘッダ札、テール札を発行します。テッダ・テール札でよく利用するシステム変数の[総発行枚数]、[レイアウト名]も MLComponent から設定可能です。

■ヘッダ・テール札を発行する

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥price.mllayx"

'入力データ指定(ヘッダ札・テール札用のデータも含めて入力)
MLComponent.PrnData = _
        "ジャケット" & vbTab & "19000" & vbTab & "490310041310" & vbTab & "東京本店" &
vbTab & "3"

'消費税指定（税編集利用時）
MLComponent.TaxRate = "10"

'システム変数[総発行枚数]指定
MLComponent.TotalQtyCaption = "3"

'システム変数[レイアウト名]指定
MLComponent.LayoutNameCaption = "プライスタグ"

'ヘッダ札発行
Result = MLComponent.OutputHeader
If Result <> 0 Then
        MessageBox.Show("OutputHeaderError No." & Result.ToString)
        Exit Sub
End If

'ボディ札発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
        Exit Sub
End If

'テール札発行
Result = MLComponent.OutputTail()
If Result <> 0 Then
        MessageBox.Show("OutputTailError No." & Result.ToString)
        Exit Sub
End If
```

■ヘッダ・テール札をレイアウトの設定に従って発行する

プリンタドライバ出力を利用している場合は、複数データを一括で発行できる機能と組合せて、ヘッダ・テール札をレイアウトの設定に従って自動的に発行できます。

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥price.mllayx"

'文字列型の配列を作成(ヘッダ札・テール札用のデータも含めて作成)
Dim inputData(0 To 2) As String
inputData(0) = "ジャケットA" & vbTab & "19000" & vbTab & "490310041310" & vbTab & "東京本店" & vbTab & "2"
inputData(1) = "ジャケットB" & vbTab & "15000" & vbTab & "490310841310" & vbTab & "東京本店" & vbTab & "2"
inputData(2) = "ジャケットC" & vbTab & "9500" & vbTab & "490310413108" & vbTab & "東京本店" & vbTab & "2"

'入力データの複数指定
MLComponent.SetPrnDataArray(inputData)

'消費税指定（税編集利用時）
MLComponent.TaxRate = "10"

'システム変数[総発行枚数]指定
MLComponent.TotalQtyCaption = "6"

'システム変数[レイアウト名]指定
MLComponent.LayoutNameCaption = "プライスタグ"

'ヘッダ・テール札の自動発行を指定
MLComponent.HeaderTailSetting = True

'タグ発行
Result = MLComponent.Output()
```

```
If Result ◇ 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
        Exit Sub
End If
```

2-6 多面取りラベルを使う

■多面取り

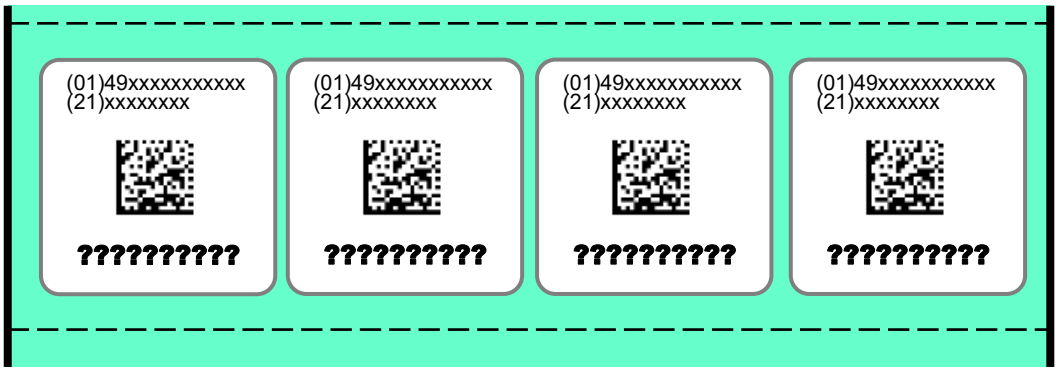
1 枚の台紙（1 シート）に複数のラベルが配置された多面取りラベルを使用します。出力方法によって、入力できる発行枚数が異なりますのでご注意ください。

インターフェース出力の場合、1 シート分の発行枚数のみ指定可能です。例として、1 シート 4 面取りのラベルに発行枚数「6」を入力して発行すると Output メソッドでエラー801 が返送されます。

プリンタドライバ出力の場合、発行枚数に制限はありません。複数データの指定と合わせれば、多くのラベルを一回の指示で発行できます。

■多面取りを 1 シート分入力して発行する

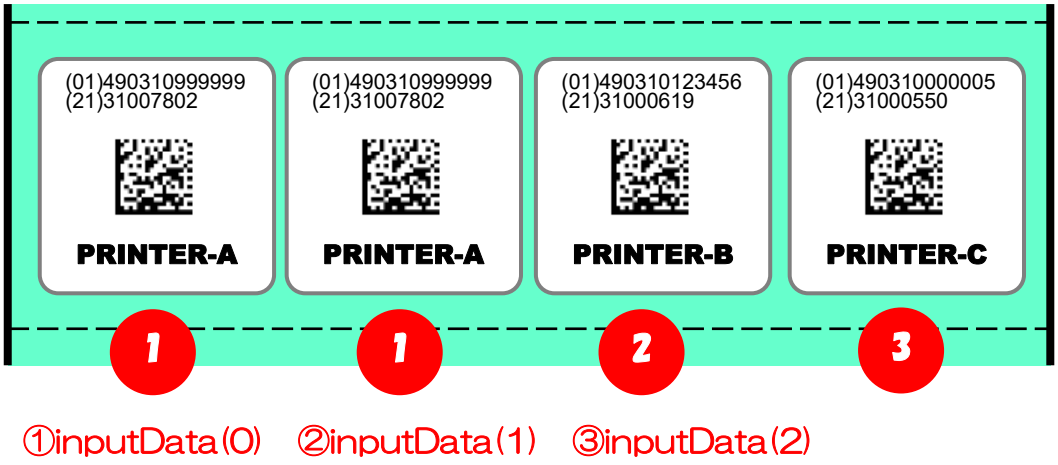
例)1 シート 4 面取りラベルの場合



・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥sheet.mllayx"

'文字列型の配列を作成(発行枚数の合計4)
Dim inputData(0 To 2) As String
inputData(0) = "PRINTER-A" & vbTab & "490310999999" & vbTab & "31007802" & vbTab & "2"
inputData(1) = "PRINTER-B" & vbTab & "490310123456" & vbTab & "31000619" & vbTab & "1"
inputData(2) = "PRINTER-C" & vbTab & "490310000005" & vbTab & "31000550" & vbTab & "1"

'入力データの複数指定
MLComponent.SetPrnDataArray(inputData)
```

```
'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
        Exit Sub
End If
```

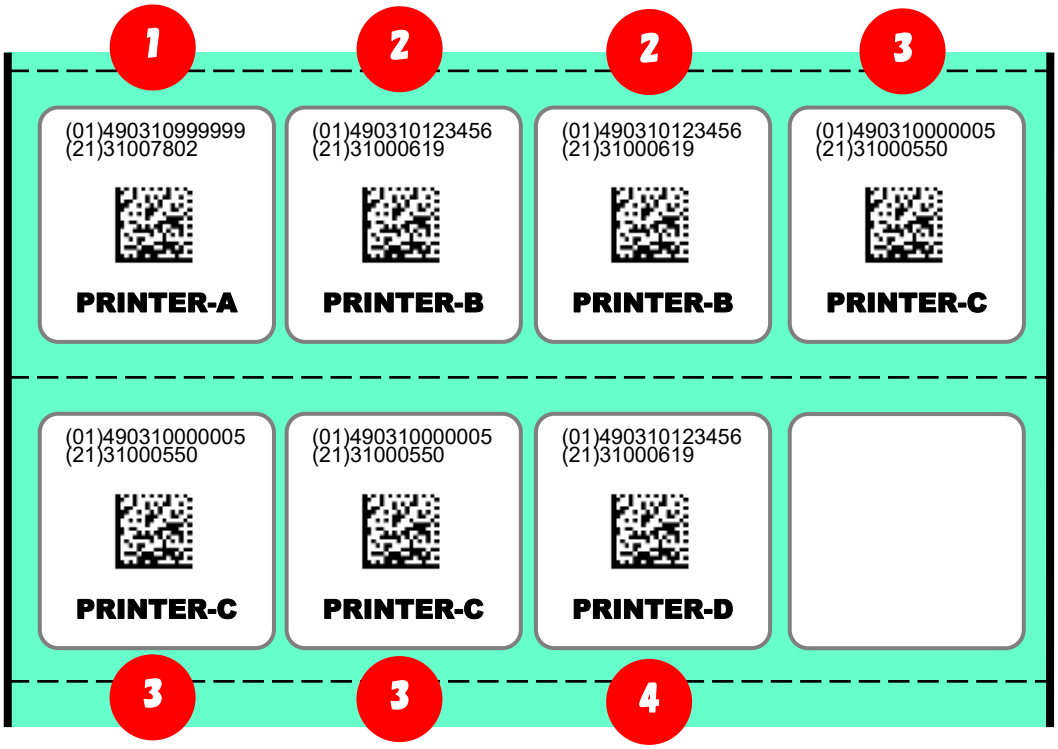


①inputData(0)　②inputData(1)　③inputData(2)

■多面取りを複数シート分入力して発行する（プリンタドライバ出力のみ）

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'レイアウトファイル指定
MLComponent.LayoutFile = "C:¥sato¥sheet.mllayx"

'文字列型の配列を作成(発行枚数の合計6)
Dim inputData(0 To 3) As String
inputData(0) = "PRINTER-A" & vbTab & "490310999999" & vbTab & "31007802" & vbTab &
"1"
inputData(1) = "PRINTER-B" & vbTab & "490310123456" & vbTab & "31000619" & vbTab &
"2"
inputData(2) = "PRINTER-C" & vbTab & "490310000005" & vbTab & "31000550" & vbTab &
"3"
inputData(3) = "PRINTER-D" & vbTab & "490310041310" & vbTab & "31000100" & vbTab &
"1"

'入力データの複数指定
MLComponent.SetPrnDataArray(inputData)

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
        Exit Sub
End If
```



①inputData(0)　②inputData(1)　③inputData(2)　④inputData(3)

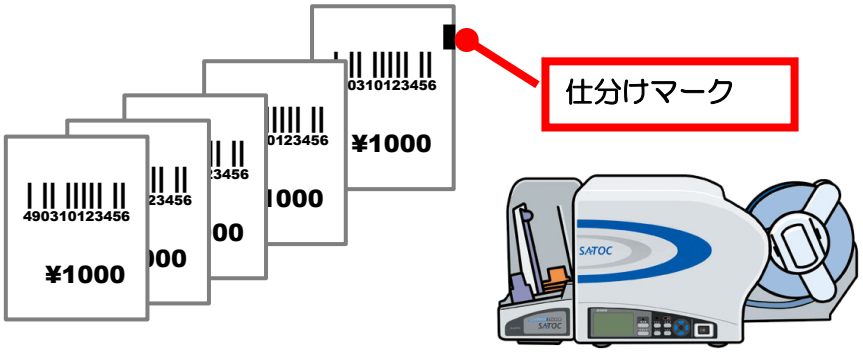
2-7 仕分けマークを印字する

■仕分けマーク

複数種類のタグを発行する際に、種類の切り替わりを分かりやすくする仕分けマークを印字します。利用できるプリンタ機種は「リファレンスマニュアル」のサポートプリンタをご確認ください。

■仕分けマークを印字する

先頭タグの側面に印字されます。スタッカをセットするとより効果的な運用が可能です。



・サンプルコード

```
'処理結果
Dim Result As Integer

'仕分けマークを使用
MLComponent.SortMark = True

'ラベル発行
Result = MLComponent.Output()
If Result <> 0 Then
        MessageBox.Show("OutputError No." & Result.ToString)
End If
```

2-8 タグ・ラベルをカットする

■カット

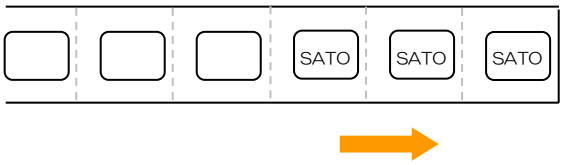
タグ・ラベルを好きなタイミングでカットすることが出来ます。カット方法は様々ありますが、お客様の運用に合わせて最適な方法をご選択ください。利用できるプリンタ機種は「リファレンスマニュアル」のサポートプリンタをご確認ください。

■カットを行う

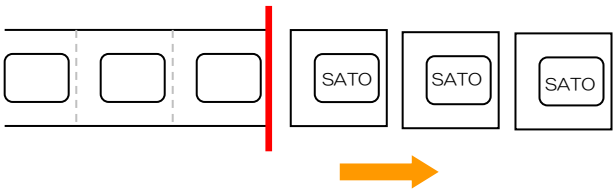
カット方法	設定方法	
	MultiCut プロパティ	EjectCut プロパティ
カットしない（デフォルト値）	0	False
プリンタの動作モードに従う	-1	必要なし
指定した枚数でカットする	枚数指定	必要なし
発行指示毎にカットする	0	True
レイアウトの設定に従う	-2	必要なし

・プリンタの動作モードに従う

例）発行枚数が３枚の場合



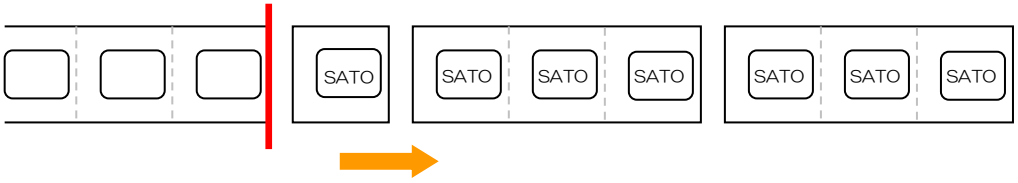
プリンタが[連続発行]の場合は、
カットしません。



プリンタが[カッタ]の場合は、
１枚毎にカットします。

・指定した枚数でカットする

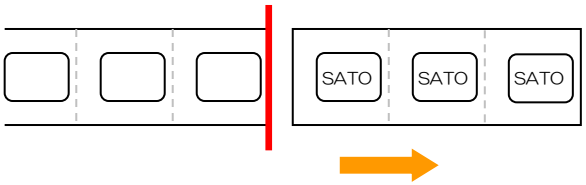
例）発行される枚数が７枚、指定枚数が３の場合



指定枚数毎にカットし、端数は末尾でカットします。

・発行指示毎にカットする

例）発行枚数が３枚の場合

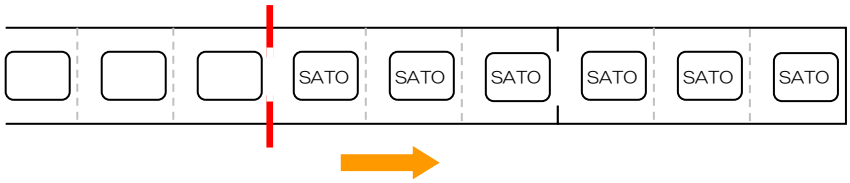


１回の発行指示で、すべてのラベルが発行された後にカットします。

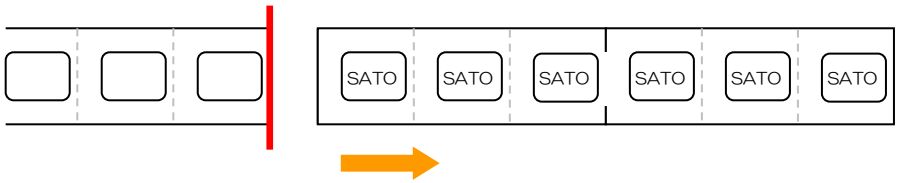
・パーシャルカットで最後のラベルを全カットする（パーシャルカット対応機種のみ）

パーシャルカットモードに設定し、最後のラベルを全カットする場合は、EjectCut プロパティを「True」に設定してください。

例）MultiCut のみの場合（パーシャルカット）



例）EjectCut を利用した場合（パーシャルカット、全カット）



・サンプルコード

```
'カットしない
MLComponent.MultiCut = 0
MLComponent.EjectCut = False

'プリンタの動作モードに従う
MLComponent.MultiCut = -1

'5枚毎にカットする
MLComponent.MultiCut = 5

'発行指示毎にカットする
MLComponent.MultiCut = 0
MLComponent.EjectCut = True

'レイアウトの設定に従う
MLComponent.MultiCut = -2
```

2-9 ラベル発行を中止する

■発行キャンセル

プリンタに送信したすべてのデータをクリアしてラベル発行を中止します。

■発行キャンセル

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'処理結果
Dim Result As Integer

'発行中止
Result = MLComponent.SendCancel()
If Result <> 0 Then
        MessageBox.Show("SendCancelError No." & Result.ToString)
        Exit Sub
End If
```

2-10 プリンタコマンド（SBPL）を送信する

■コマンド送信　■コマンド受信

プリンタにプリンタコマンド（SBPL：Sato Barcode Printer Language）を送信します。SBPL の仕様を十分にご理解いただいてからご利用ください。間違った SBPL を送信すると、プリンタでコマンドエラーが発生したり、ラベル発行が中断したり、思わぬトラブルが発生する可能性があります。SBPL の詳細は当社営業担当までご相談ください。

■コマンド送信

SBPL は、文字列型もしくはバイト配列型で送信できます。
文字列型は SendStringData メソッドを、バイト配列型は SendRawData メソッドを使用します。
SendRawData メソッドは NULL（16 進文字コード：00）など文字列で表現できないバイナリデータを送信する際に利用します。

■コマンド受信

SBPL には、プリンタのバージョン情報や動作設定などデータが返送されるコマンドがあります。
返送されたデータは、文字列、バイト配列、16 進文字コードで取得できます。データに NULL（16 進文字コード：00）など文字列で表現できないバイナリデータがある場合は、バイト配列か 16 進文字コードで取得してください。

例）返送データが「13.00.03.00」の場合
文字列　　：″13.00.03.00″
バイト配列　　：{31h , 33h , 2Eh , 30h , 30h , 30h , 2Eh , 30h , 33h , 2Eh , 30h , 30h}
16 進文字コード ：″31332E30302E30332E3030″

■SBPL を文字列で送信する

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'返送データ
Dim Result As String

'SBPLを文字列型で作成(DC2+PG：プリンタステータス情報取得コマンド)
Dim printerCommand As String
printerCommand = Chr(&H12) & ″PG″

'SBPL送信(受信終了キャラクタ：ETX、文字列で受信)
Try
        Result = MLComponent.SendStringData(0, printerCommand, 0, Chr(&H3))
Catch ex As SATO.MLComponent.MLComponentException
        MessageBox.Show(″SendStringDataError No.″ & ex.Number.ToString)
        Exit Sub
End Try
MessageBox.Show(″SendStringData Result=″ & Result)
```

■SBPL をバイト配列で送信する

・サンプルコード

```
'前提条件としてOpenPortが成功していること

'返送データ
Dim Result As String

'SBPLをバイト配列で作成(DC2+PG：プリンタステータス情報取得コマンド)
Dim printerCommand(0 To 2) As Byte
printerCommand(0) = &H12
printerCommand(1) = &H50
printerCommand(2) = &H47

'SBPL送信(受信終了キャラクタ：ETX、16進文字コードで受信)
Try
        Result = MLComponent.SendRawData(2, printerCommand, 0, Chr(&H3))
Catch ex As SATO.MLComponent.MLComponentException
        MessageBox.Show("SendRawDataError No." & ex.Number.ToString)
        Exit Sub
End Try
MessageBox.Show("SendRawData Result=" & Result)
```

2-11 動作設定ファイルを利用する

■動作設定ファイル

MLComponentと同じフォルダに動作設定ファイルを配置することで、プロパティでは設定できない拡張設定が利用できます。設定値（XMLタグ）がない場合は、初期値が利用されます。

■ファイル名

MLComponentSettings.xml

■格納先

MLComponent.dllと同一フォルダ

■文字エンコーディング

Unicode（UTF-8）

■書式（記述例）

```
<?xml version="1.0"?>
<MLComponentSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <IsLog>false</IsLog>
  <LogFolder>C:¥Sato¥Logs</LogFolder>
  <IsSheetCountError>false</IsSheetCountError>
  <TaxRate>8.0,8.0</TaxRate>
  <AlternativeFont>false</AlternativeFont>
  <DesignDefaultWindowsFontName>MS ゴシック</DesignDefaultWindowsFontName>
  <DesignDefaultWindowsFontSize>9</DesignDefaultWindowsFontSize>
  <IsFileForwardCompatibleError>true</IsFileForwardCompatibleError>
  <IsEnableGcCollectForced>false</IsEnableGcCollectForced>
  <GcCollectForcedThreshold>0</GcCollectForcedThreshold>
  <IsWindowsFontTraceLog>false</IsWindowsFontTraceLog>
  <WindowsFontDrawingNumberOfTimes>0</WindowsFontDrawingNumberOfTimes>
  <WindowsFontTraceLogFolder></WindowsFontTraceLogFolder>
</MLComponentSettings>
```

動作設定ファイルのサンプルはソフトウェア・ダウンロード・サイトにて公開しております。

https://www.sato.co.jp/support/printertool/tool/multi-labelist-component/

IsLog（ログ出力の有効・無効）	
ログファイルを出力するかを設定します。	
true	ログファイルを出力します。
false(初期値)	ログファイルを出力しません。

LogFolder（ログ出力先）
ログファイルの出力先フォルダをフルパスで指定します。 IsLogが有効な場合に使用されます。

IsSheetCountError（発行枚数エラーの有効・無効）	
発行枚数が指定されていない場合に、エラーとするかを設定します。MLOCX互換設定のため、通常は設定を変更せず、印字データで発行枚数を指定してください。	
true(初期値)	PrnData プロパティまたはPrnDataArrayメソッドで発行枚数が指定されていない場合に、Output メソッドがエラーNo.802となります。
false	発行枚数が未設定でも発行エラーとせず、発行枚数ゼロでプリンタコマンドを送信します。カット動作が機能しないなど問題が発生する可能性があります。通常は使用しないでください。

TaxRate（税率の設定）
税編集で使用する税率を設定します。設定値はTaxRateプロパティをご参照ください。 緊急対応用として用意している設定値です。通常はTaxRateプロパティで指定してください。

AlternativeFont（代替フォント使用の有効・無効）	
レイアウトで使用しているWindowsフォントがない場合に、代替フォントを使用するか設定します。	
true	代替フォントが利用されます。 デザイン時に指定したフォントと異なるフォントで印字されるため、文字の形状やサイズ、自動改行位置が異なります。通常は発行環境に該当フォントをインストールするか、フォントを変更するなどの対応を行ってください。
false(初期値)	Outputメソッドがエラーno.600となります。

DesignDefaultWindowsFontName（デフォルトフォントのフォント設定）
代替フォントに使用されるデフォルトフォントのフォントをWindowsフォント名で設定します。 本設定がない場合は、OSのデフォルトフォントが利用されます。

DesignDefaultWindowsFontSize（デフォルトフォントのフォントサイズ設定）
代替フォントに使用されるデフォルトフォントのフォントサイズをポイント数で設定します。 本設定がない場合は、OSのデフォルトフォントサイズが利用されます。

IsFileForwardCompatibleError（ファイルバージョンチェックの有効・無効）	
レイアウトのファイルバージョンがMLComponentより新しい場合に発行エラーとするか設定します。	
true(初期値)	発行エラー（No.61～66）となります。
false	エラーが発生しません。 レイアウトによっては、正常に印字されない危険性があります。ファイルバージョンチェックを無効にする場合は、レイアウトの新規追加・編集に十分注意し、変更後は運用前に必ず正常に印字されるかテストを行ってください。

IsEnableGcCollectForced（メモリ自動解放の有効・無効）
アプリケーションの使用メモリが閾値に達した場合にガベージコレクションによるメモリ解放(GC.Correct)

を実行します。閾値はGcCollectForcedThresholdで指定します。	
true	自動解放を行います。
false(初期値)	自動解放を行いません。

GcCollectForcedThreshold（メモリ自動解放の閾値設定）	
メモリ解放を行う閾値（アプリケーションのメモリ使用量）を設定します。	
0(初期値)	発行する度にメモリ解放を実行します。 発行速度が遅延するため、アプリケーションのメモリ使用量によって、適切な値に調整してください。
1～2048(MB)	MLComponentを組み込んだアプリケーションが指定したメモリ使用量（MB単位）を越えた場合に、メモリ解放を実行します。

■ログファイルを出力する

動作設定ファイルでログ出力を有効にすることで、通信・発行・プリンタ制御のメソッドを実行する度に、ログを出力することができます。

ログファイルは、MLComponentがロードされたタイミングで「MLComponent_*.log」（*GUID）のファイル名で生成され、メソッドが呼び出されるタイミングでログが出力されます。MLComponentがアンロードされると編集中のファイルを閉じます。

ログファイルの出力時に、フォルダ有無、権限、ディスク残量等のチェックなどは行いません。また、出力したログファイルの自動削除も行いませんので、手動またはお客様のアプリケーションで削除してください。

・書式

20XX/11/25 10:00:00:123<tab>OpenPort<tab>0<tab>COM1:9600,n,8,1…
① ② ③ ④

①メソッドの実行日時
YYYY/MM/DD HH:MM:SS:MMM

②メソッド名

通信	OpenPort、ClosePort
発行	Output、OutputHeader、OutputTail、SendStringData、SendRawData
プリンタ制御	GetStatus、Cut、SendCancel

③メソッドの戻り値
メソッドの実行結果を表す戻り値（0：正常終了、0以外はエラー）
SendStringData、SendRawDataは例外エラーの番号

④付加情報

メソッド毎に出力されるプロパティの値やステータスがタブ区切りで付加されます。

メソッド	内容
OpenPort	Setting、Protocol、Timeout
ClosePort	なし
Output	LayoutFile、Darkness、Speed、Offset、MultiCut、SortMark、HeaderTailSetting、Formoverlay、TaxRate、PrnDataType、PrnData（PrnDataArray）
OutputHeader	
OutputTail	
SendStringData	なし
SendRawData	なし
GetStatus	プリンタから受信したステータス文字列
Cut	なし
SendCancel	なし

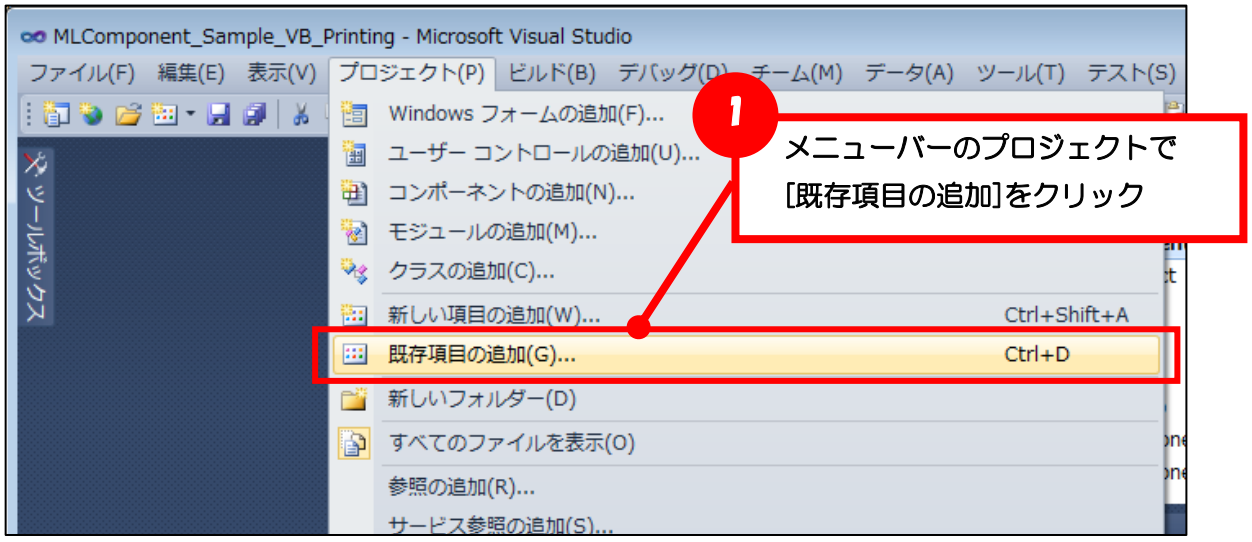
2-12 レイアウトの初回読込速度を改善する

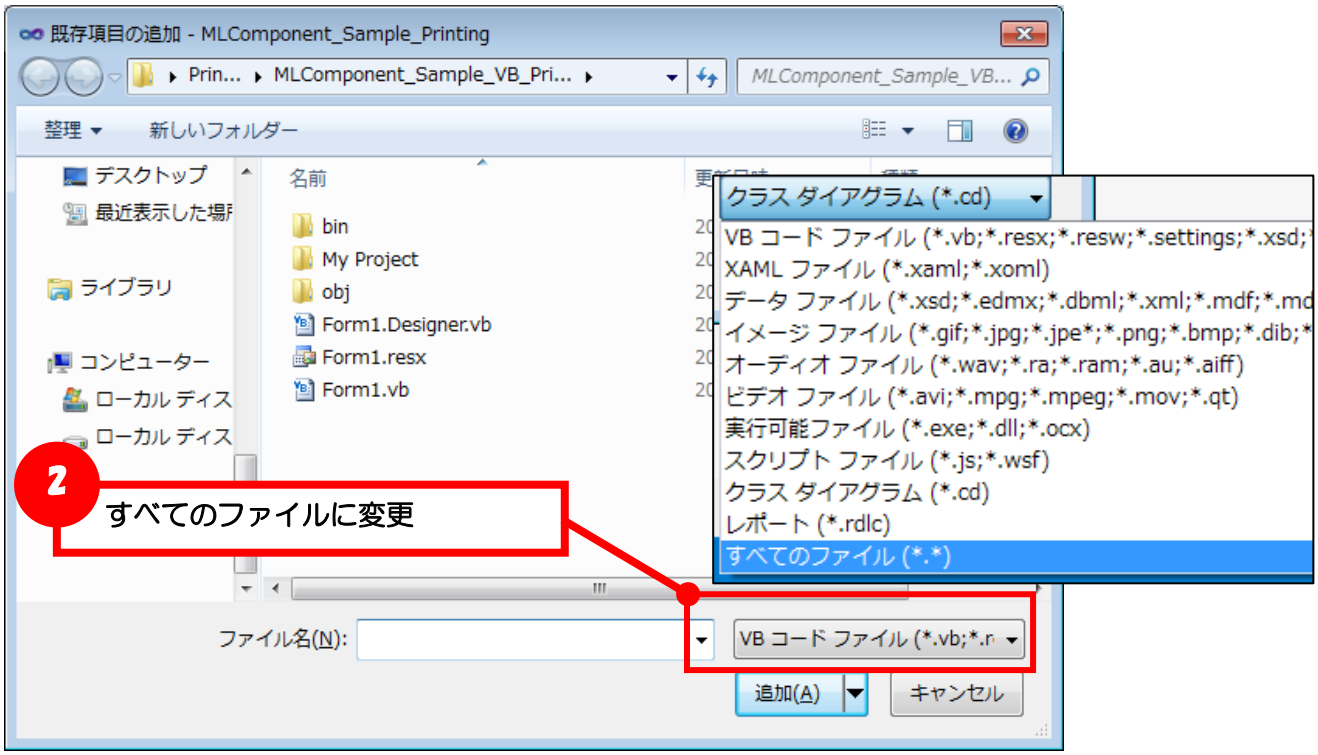
■MLComponent.XmlSerializers.dll

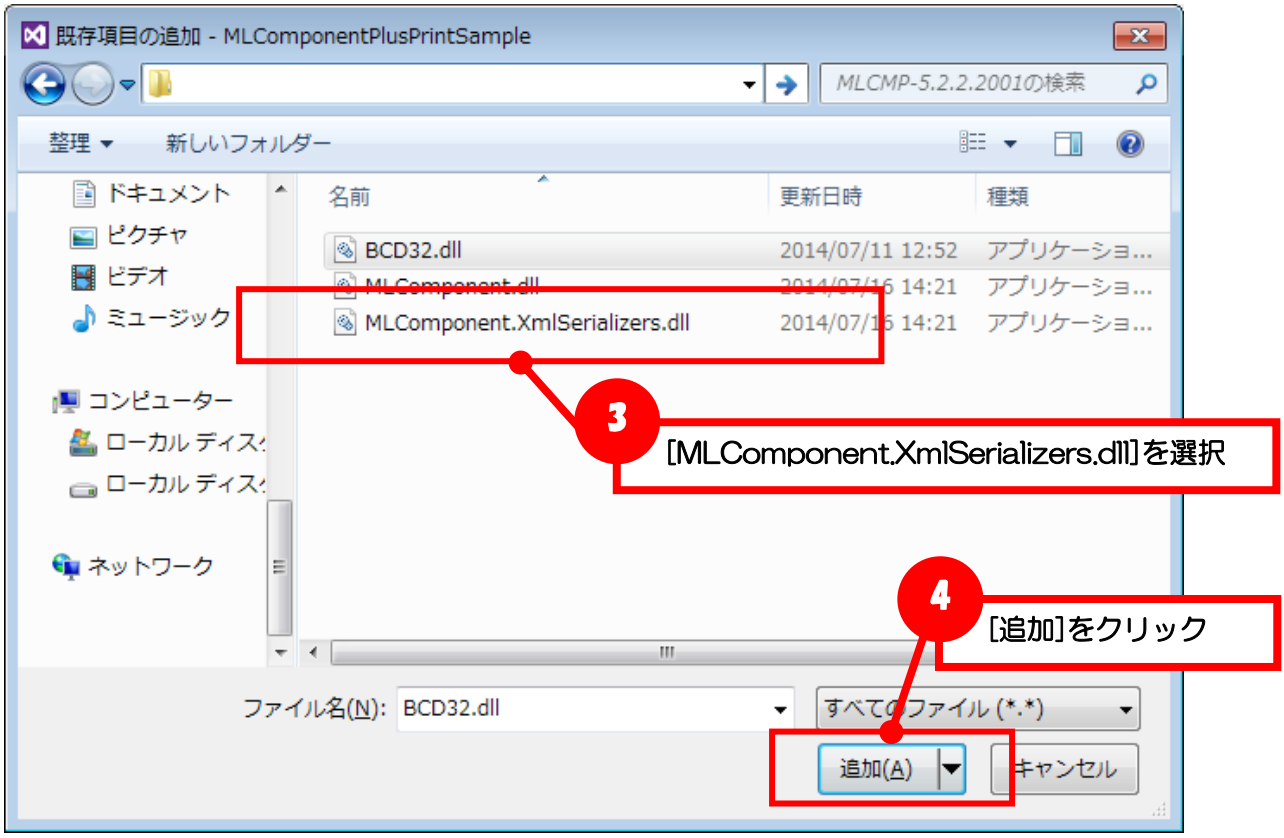
■MLComponent.XmlSerializers.dll を配置する

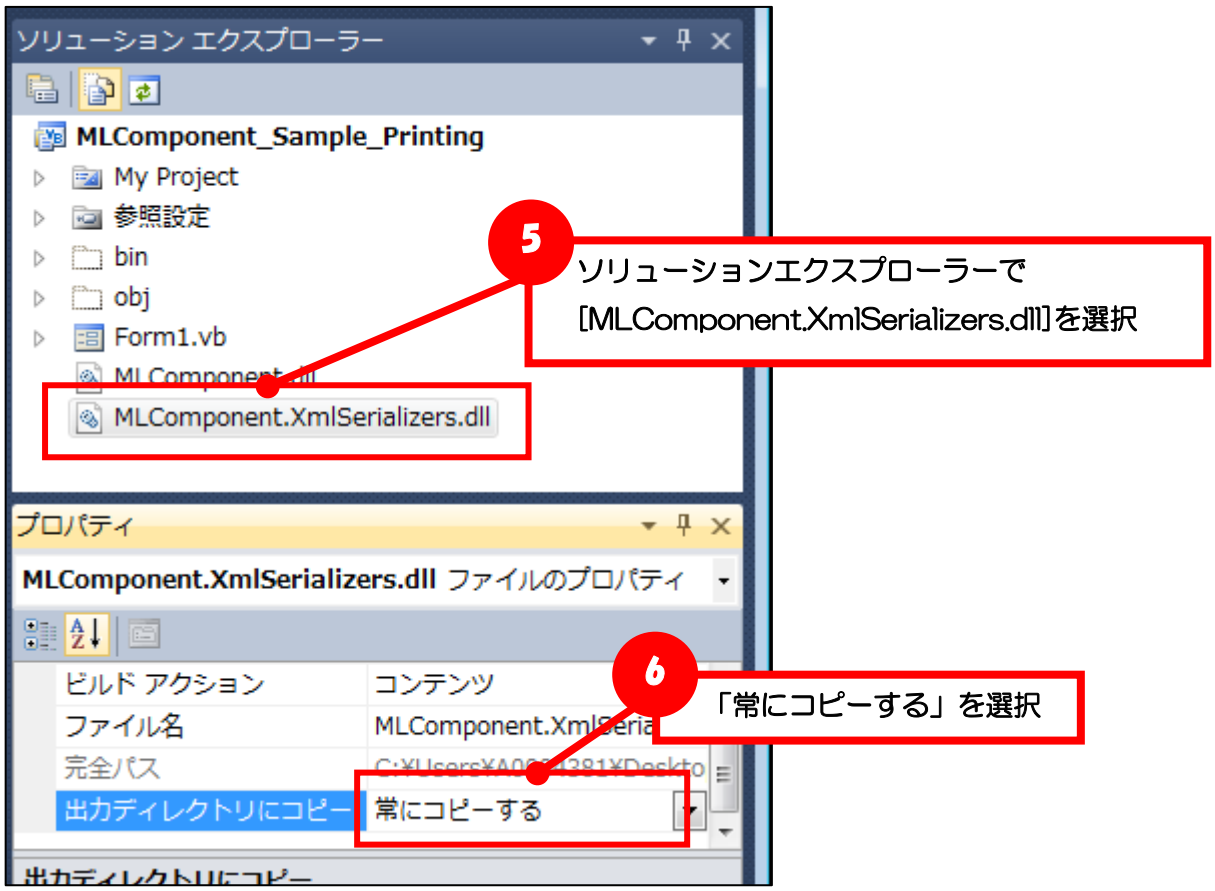
MLComponent で使用している.NET Framework のライブラリがロードされるため、アプリケーション起動後に、初めてレイアウト情報を読み込む時（Output メソッドや GetPriter メソッドなど）に処理が遅延することがあります。

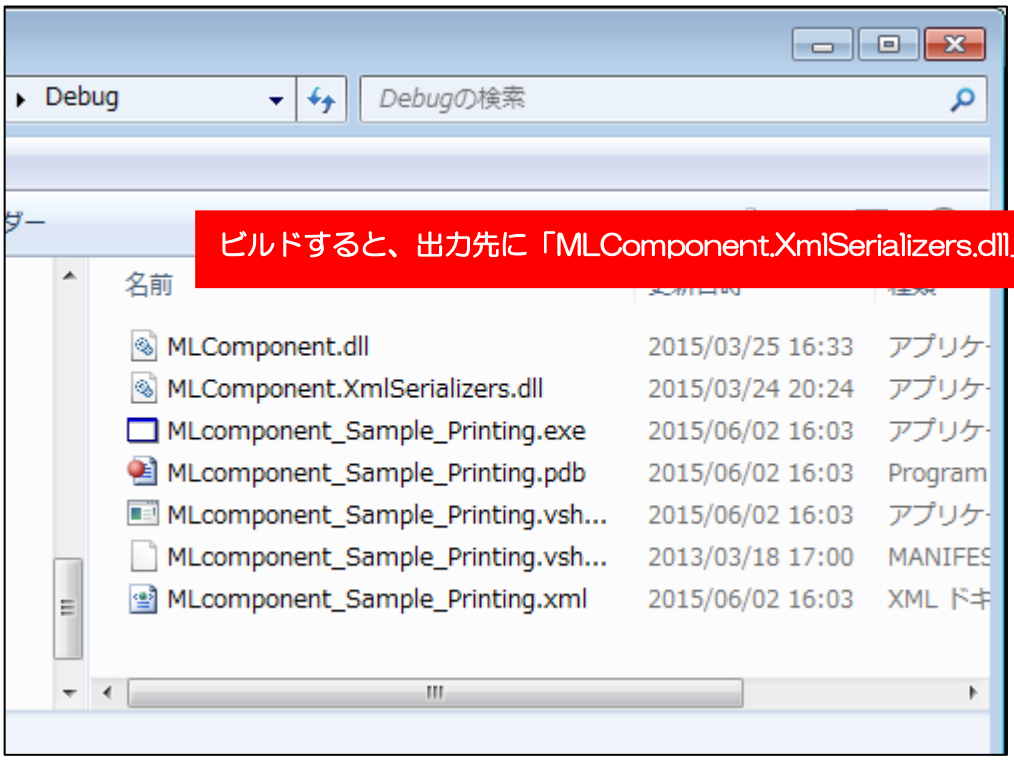
ライブラリのロード時間を改善するために、XML シリアライザ「MLComponent.XmlSerializers.dll」を MLComponent.dll と同じフォルダに配置します。例として Visual Studio でビルドする際に、MLComponent.XmlSerializers.dll を自動的に配置する方法を説明します。






既存項目の追加 - MLComponentPlusPrintSample
MLCMP-5.2.2.2001の検索
整理
新しいフォルダー
ドキュメント
ピクチャ
ビデオ
ミュージック
コンピューター
ローカル ディス
ローカル ディス
ネットワーク
名前
更新日時
種類
BCD32.dll
2014/07/11 12:52
アプリケーショ...
MLComponent.dll
2014/07/16 14:21
アプリケーショ...
MLComponent.XmlSerializers.dll
2014/07/16 14:21
アプリケーショ...
3
[MLComponent.XmlSerializers.dll]を選択
4
[追加]をクリック
ファイル名(N): BCD32.dll
すべてのファイル (*.*)
追加(A)
キャンセル


ソリューション エクスプローラー
MLComponent_Sample_Printing
My Project
参照設定
bin
obj
Form1.vb
MLComponent.dll
MLComponent.XmlSerializers.dll
5
ソリューションエクスプローラーで
[MLComponent.XmlSerializers.dll]を選択
プロパティ
MLComponent.XmlSerializers.dll ファイルのプロパティ
ビルド アクション
コンテンツ
ファイル名
MLComponent.XmlSeria
完全パス
出力ディレクトリにコピー
常にコピーする
6
「常にコピーする」を選択



ビルドすると、出力先に「MLComponent.XmlSerializers.dll」が配置されます。

2-13 発行速度を改善する

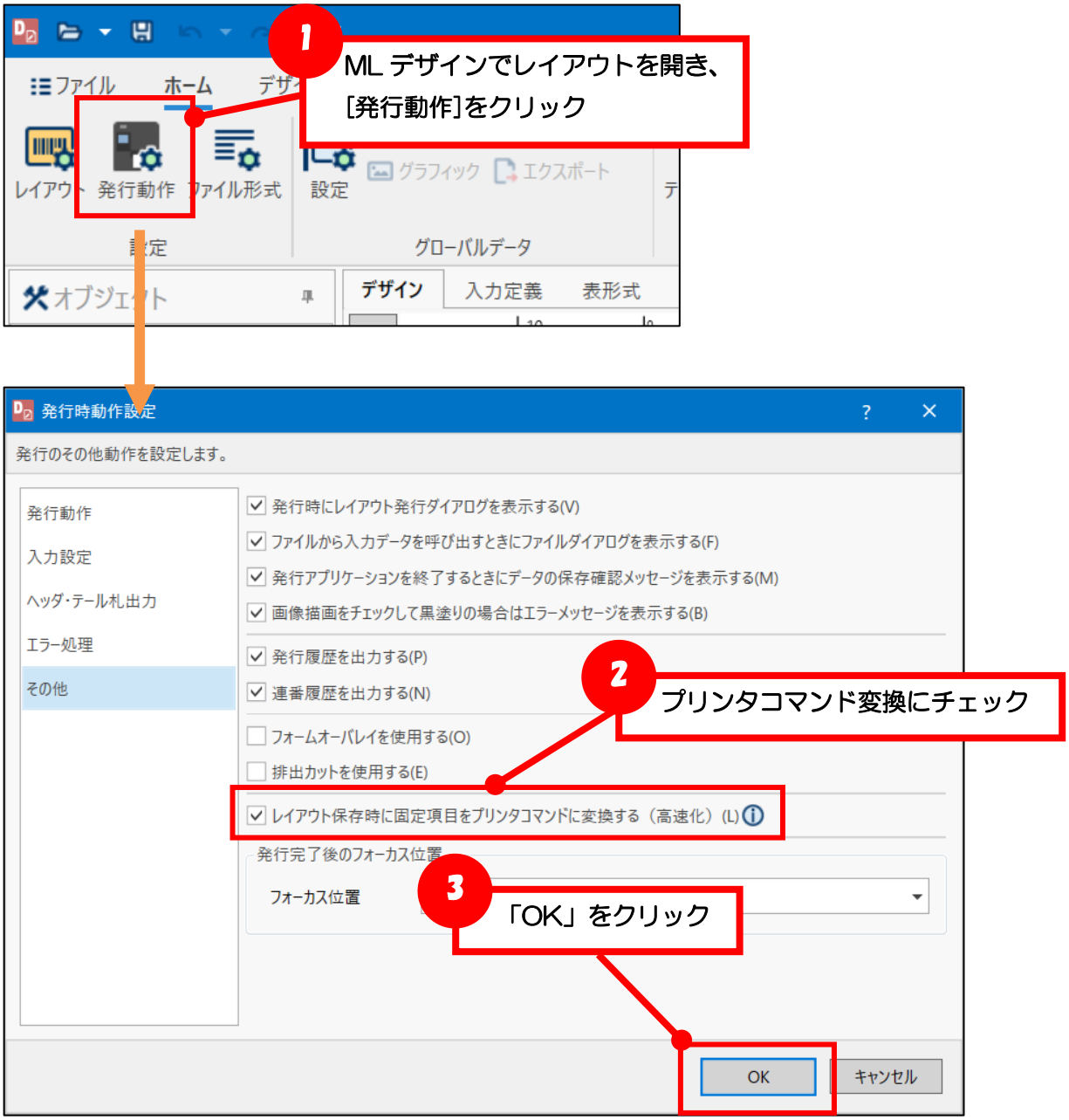
■固定オブジェクトのプリンタコマンド生成

■固定オブジェクトのプリンタコマンド生成を事前に行う

MLV5 Ver.5.7.5.0 以降、MLV6 Ver.6.0.0.0 以降で利用可能です。

貼り付け文字や罫線など、データによって内容が変わらない固定オブジェクトをレイアウト保存時にプリンタコマンドに変換することで、ラベル発行時の速度を改善させることが可能です。

固定オブジェクトの数が多い場合やWindowsフォントの貼り付け文字で自動改行が多く利用されている場合に高い効果が期待できます。



MLComponent 側の設定はありません。

Ver.5.7.5.0 より古いバージョンの場合は、Ver.5.7.5.0 以降にバージョンアップしてください。