CENTUM VP

APCS ユーザカスタムブロック
開発環境

IM 33J15U23-01JA

IM 33J15U23-01JA
2 版

はじめに

APCS ユーザカスタムブロック開発環境機能とは、APCS（Advanced Process Control Station）上で動作するユーザカスタムブロックの演算アルゴリズムを C 言語で作成するために必要な環境を提供するものです。
APCS ユーザカスタムブロック開発環境下で、Microsoft 社の Visual C++ を使用し、ダイナミックリンクライブラリ（DLL）としてユーザカスタムブロックの演算アルゴリズムを作成します。ユーザは、Visual Studio（Visual C++ を含めた開発環境）などの豊富な機能を利用して、演算アルゴリズムをプログラミングすることができます。
なお、本書に関連するドキュメントとして次のものがあります。

- APCS（IM 33J15U10-01JA）
- APCS ユーザカスタムブロック（IM 33J15U20-01JA）
- APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）
- APCS ユーザカスタムブロック ライブラリ（IM 33J15U22-01JA）

2nd Edition : Aug. 2019 (YK)

安全に使用するための注意事項

■ 本製品の保護・安全および改造に関する注意

- 本製品によって制御されるシステムおよび本製品自体を保護し、安全に操作するために、本書に記載されている安全に使用するための注意事項に従ってください。指示事項に反する扱いをされた場合、横河電機株式会社（以下、当社といいます）は安全性の保証をいたしかねます。
- ユーザーズマニュアルで指定していない方法で製品を使用した場合は、本製品で提供される保護機能が損なわれる可能性があります。
- 本製品によって制御されるシステムおよび本製品そのものに保護または安全回路が必要な場合は、本製品外部に別途ご用意ください。
- 本製品と組み合わせて使用する機器の仕様と設定については、必ず、機器の取扱説明書などで確認してください。
- 本製品の部品または消耗品を交換する場合は、当社が指定する部品のみを使用してください。
- 本製品および本製品の電源コードセットなどの付属品を、当社が指定する機器や用途以外に使用しないでください。
- 本製品を改造することは、固くお断りいたします。
- 本製品およびユーザーズマニュアルでは、安全に関する次の記号を使用しています。

「注意」を示します。本製品においては、感電など、人体への危険や機器損傷の恐れがあることを示すとともに、ユーザーズマニュアルを参照する必要があることを示します。また、ユーザーズマニュアルにおいては、人体への危険や機器損傷を避けるための注意事項が記載されている箇所に、本記号を「注意」「警告」の用語と一緒に使用しています。

「注意、高温表面」を示します。このマークの付いた機器は熱くなりますのでご注意ください。接触するとやけどなどの危険があります。

「保護導体端子」を示しています。感電防止のため、本製品を使用する前に、保護導体端子を必ず接地してください。

「機能接地端子」を示しています。「FG」と表示された端子も同じ機能を備えています。保護接地以外を目的とした接地端子です。本製品を使用する前に、機能接地端子を必ず接地してください。

「AC 電源」を示します。

「DC 電源」を示します。

「オン」を示します。電源スイッチなどの状態を示します。

「オフ」を示します。電源スイッチなどの状態を示します。

■ ユーザーズマニュアルに対する注意

- ユーザーズマニュアルは、最終ユーザまでお届けいただき、最終ユーザがお手元に保管して随時参照できるようにしてください。
- ユーザーズマニュアルをよく読んで、内容を理解したのちに本製品を操作してください。
- ユーザーズマニュアルは、本製品に含まれる機能詳細を説明するものであり、お客様の特定目的に適合することを保証するものではありません。
- ユーザーズマニュアルの内容については、将来予告なしに変更することがあります。
- ユーザーズマニュアルの内容について万全を期していますが、もしご不審な点や誤り、記載もれなどお気付きのことがありましたら、当社またはお買い求め先代理店までご連絡ください。乱丁、落丁はお取り替えいたします。

■ 本製品の免責について

- 当社は、保証条項に定める場合を除き、本製品に関していかなる保証も行いません。
- 本製品のご使用または使用不能から生じる間接損害については、当社は一切責任を負いかねますのでご了承ください。

■ ソフトウェア製品について

- 当社は、保証条項に定める場合を除き、本ソフトウェアに関していかなる保証も行いません。
- 本製品の各ソフトウェアに対するライセンスは、ご使用になるコンピュータの台数に応じて適正にご購入ください。
- バックアップ以外の目的で本ソフトウェアを複製することは、当社の知的所有権を侵害する行為であり、固くお断りいたします。
- 本ソフトウェアが収められているソフトウェアメディアは、大切に保管してください。
- 本ソフトウェアをリバースコンパイル、リバースアセンブリ、リバースエンジニアリング、その他の方法により人間が読み取り可能な形にすることは、固くお断りします。
- 当社から事前の書面による承認を得ることなく、本ソフトウェアの全部または一部を譲渡、交換、転貸などによって第三者に使用させることは、固くお断りいたします。

ユーザーズマニュアル中の凡例

■ ユーザーズマニュアル中のシンボルマーク

ユーザーズマニュアルの本文中では、次の各種記号が使用されています。

警告

死亡または重傷を招く可能性がある危険な状況を避けるための注意事項を記載しています。

注意

軽傷または物的損害を招く可能性がある危険な状況を避けるための注意事項を記載しています。

重要

操作や機能を知る上で、注意すべき事柄を記載しています。

補足

説明を補足するための事柄を記載しています。

参照

参照先を示します。
オンラインマニュアルでは、緑色の参照先をクリックすると、該当箇所が表示されます。黒色の参照先は、該当箇所が表示されません。

■ ユーザーズマニュアル中の表記

ユーザーズマニュアル中の表記は、次の内容を示します。

● ユーザーズマニュアル全体を通して共通に使用されている表記

入力文字列

次の書体の文字列は、ユーザが実際の操作において入力する内容を示します。
例：

```
FIC100.SV=50.0
```

▼記号

本製品のエンジニアリングを行うウィンドウの定義項目に関する説明箇所であることを示します。
本製品のエンジニアリングを行うウィンドウのヘルプメニューから「ビルダ定義項目一覧」を選択したときに開くウィンドウを経由して、選択した項目の説明を表示できます。なお、複数の定義項目が併記されている場合には、複数の定義項目に関する説明箇所であることを示します。
例：

▼ タグ名、ステーション名

Δ 記号

ユーザが入力する文字列で、空白文字（スペース）を示します。
例：

.AL Δ PIC010 Δ -SC

{} で囲った文字

ユーザが入力する文字列で、省略可能な文字列を示します。
例：

.PR Δ TAG {Δ . シート名}

● キーまたはボタン操作を示すために使用されている表記

[] で囲った文字

キーまたはボタンの操作説明において［］で囲まれている文字は、キーボードのキー、オペレーションキーボードのキー、ウィンドウに表示されるボタン名、またはウィンドウに表示されるリストボックスの選択項目のいずれかを示します。
例：

機能を切り替えるには［ESC］キーを押します。

● コマンド文やプログラム文などの書式説明の中で使用されている表記

コマンド文やプログラム文などの書式説明の中で使用されている表記は、次の内容を示します。

＜＞で囲った文字

ユーザが一定の規則に沿って任意に指定できる文字列を示します。

例：

```
#define ＜識別子＞ ＜文字列＞
```

…記号

直前のコマンドや引数が繰り返し可能であることを示します。

例：

```
lmax (arg1, arg2, …)
```

[ ] で囲った文字

省略可能な文字列を示します。

例：

```
sysalarm ＜フォーマット文字列＞ [, ＜出力値＞…]
```

| | で囲った文字

ユーザが複数候補から任意に選択できる文字列を示します。

例：

```
opeguide | ＜フォーマット文字列＞[,＜出力値＞…] |
         | OG,＜素子番号＞                       |
```

■ 図の表記

ユーザーズマニュアルに記載されている図は、説明の都合上、部分的に強調、簡略化、または省略されていることがあります。

ウィンドウの図では、機能理解や操作監視に支障を与えない範囲で、実際の表示と部品の表示位置や、大文字小文字など文字の種類が異なっている場合があります。

■ 入力文字

Windows では半角カタカナを使用できますが、本製品のソフトウェアへ入力する文字列には、半角カタカナを使用しないでください。

著作権および商標

■ 著作権

■ 商標

- CENTUM、ProSafe、Vnet/IP、PRM、Exaopc、Exaplog、Exapilot、Exaquantum、Exasmoc、Exarqe、Multivariable Optimizing Control/Robust Quality Estimation、StoryVIEW および FieldMate Validator は、横河電機株式会社の登録商標または商標です。
- 本製品で使用されている会社名、団体名、商品名およびロゴ等は、横河電機株式会社、各社または各団体の登録商標または商標です。

Blank Page

APCS ユーザカスタムブロック
開発環境

IM 33J15U23-01JA 2 版

目　次

Blank Page

1. 概要

APCSユーザカスタムブロック開発環境の概要について説明します。APCSユーザカスタムブロック開発環境機能は、APCS上で動作するユーザカスタムブロックの演算アルゴリズム（以降、ユーザカスタムアルゴリズム）を、C言語で作成するために必要な環境を提供するものです。

1.1 位置づけ

APCSユーザカスタムブロック開発環境の位置づけについて説明します。

参照 本パッケージの動作環境については、APCS 関連の一般仕様書を参照してください。

■ 本機能の位置づけ

APCS ユーザカスタムブロック開発環境は、APCS 上で動作するユーザカスタムブロックのアルゴリズムを、ユーザが C 言語を用いて開発するための環境を提供するものです。

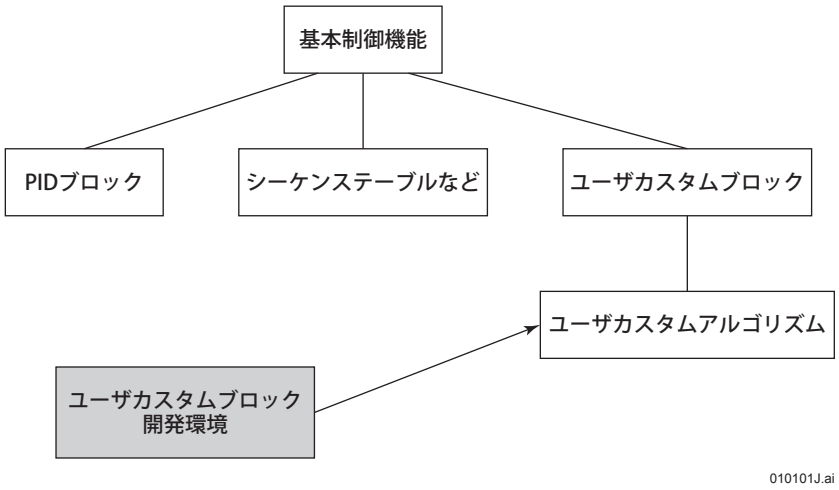


図　本機能の位置づけ

■ 他の機能との関係

APCS ユーザカスタムブロック開発環境下で作成したユーザカスタムアルゴリズムは、システムビューで APCS へ登録することで、APCS 上で動作させることができます。

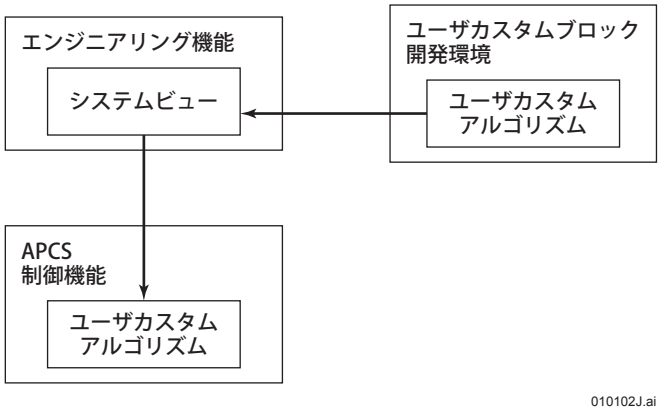


図　他の機能との関係

2. APCSユーザカスタムブロック開発環境での作業概要

本章では、APCSユーザカスタムブロック開発環境で行う操作、設定の概要について説明します。
APCSユーザカスタムブロック開発環境下では、ユーザカスタムアルゴリズムを作成します。
ここでは、ユーザカスタムアルゴリズムをMicrosoft Visual C++のダイナミックリンクライブラリ（以降、DLLと記述します）として作成します。
作成したユーザカスタムアルゴリズムは、エンジニアリング機能に登録した後、APCSへダウンロードして実行します。

参照 操作、設定手順の詳細については、以下を参照してください。
「3. ユーザカスタムアルゴリズム（UCA）の作成手順詳細」

2.1 APCSユーザカスタムブロック開発環境の構成

本節では、APCSユーザカスタムブロック開発環境が、コンピュータ上でどのように構成されているかを説明します。

■ ディレクトリ構成

APCS ユーザカスタムブロック開発環境で提供されるモジュールのディレクトリ構成を以下に示します。

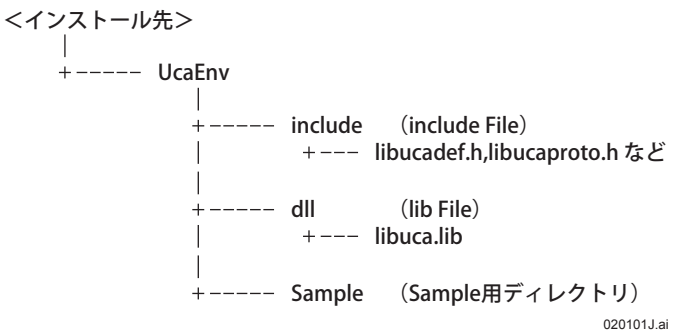


図 APCSユーザカスタムブロック開発環境で提供されるディレクトリ構成

ユーザカスタムアルゴリズム作成者は、APCS ユーザカスタムブロック開発環境で提供されるユーザカスタムアルゴリズム作成用ライブラリ、およびインクルードファイルを使用して、アルゴリズムを作成することができます。

2.2 ユーザカスタムアルゴリズム開発作業

本節では、ユーザカスタムアルゴリズムの作成作業の概要について説明します。

■ 開発作業概要

ユーザカスタムアルゴリズムの開発は、Microsoft Visual C++ 上で APCS ユーザカスタムブロック開発環境で提供されるモジュールを利用して、DLL を作成します。その後、テスト機能による APCS シミュレータでデバッグとテストを行います。テスト機能によるテスト終了後、APCS を使用して最終の動作確認テストを行ってください。

● ユーザカスタムアルゴリズムの作成

Microsoft Visual C++ で以下の作業を行ってください。

- ソリューションの作成
- ディレクトリパスの設定
- プロジェクトの設定
- ソースプログラムの作成／コンパイル／リンク

● ユーザカスタムアルゴリズムのデバッグ

Microsoft Visual C++、APCS シミュレータを使用して、作成したユーザカスタムアルゴリズムのデバッグを行ってください。
この作業の前に制御ドローイングビルダで、作成したユーザカスタムアルゴリズムを使用するユーザカスタムブロックを作成し、デバッグに使用する制御ドローイングを用意してください。

● バーチャルテスト

テスト機能を使用して、バーチャル HIS、APCS シミュレータ、および FCS シミュレータを起動し、総合的な動作確認をしてください。問題が発生した場合、Microsoft Visual C++ のデバッグ機能を用いることでデバッグすることもできます。

● 実機動作確認

APCS の実機を使用して、ユーザカスタムブロックの動作の最終確認を行ってください。

2.2.1 ユーザカスタムアルゴリズムの作成概要

Microsoft Visual C++ を使用してユーザカスタムアルゴリズムとして動作する DLL を C 言語を用いて作成します。
そのためには Microsoft Visual C++ で以下の作業が必要となります。

1. ソリューションの作成
2. ディレクトリパスの設定
3. プロジェクトの設定
4. アルゴリズムの作成
   C 言語によるソースプログラムの作成
5. コンパイル／リンク

■ ソリューションの作成

Microsoft Visual C++ 上でソリューションを作成する場合、以下に示す項目を指定してください。

● ファイル形式

ダイナミックリンクライブラリを作成するので、ファイル形式を DLL（拡張子 .dll）と指定してください。

● プロジェクト名（注：Microsoft Visual C++上でのプロジェクト名）

プロジェクト名は、ユーザカスタムアルゴリズム名になるため、ユーザカスタムアルゴリズム名を指定してください。

● ダイナミックリンクライブラリ名

作成するダイナミックリンクライブラリ名は、以下の形式としてください。
ユーザカスタムアルゴリズム名 .dll

● ユーザカスタムアルゴリズム名

16 文字以内の半角英数字、および「_」で指定してください。（ただし、先頭文字に数字は使用できません。）
大文字／小文字の区別ありませんが、ユニークな名前になるようにしてください。
下記のダイナミックリンクライブラリ名は、使用しないでください。

LIBB***	：Libb ではじまる名称
YUC***	：YUC ではじまる名称
Z***	：Z ではじまる名称
_S*** から _Z***	：_S から _Z ではじまる名称

■ ディレクトリパスの設定

ユーザアルゴリズム開発環境で提供されるファイルを参照するため、そのディレクトリパスをソリューションに設定してください。

■ プロジェクトの設定

コンパイルオプション、リンクオプションなどを設定してください。

■ アルゴリズムの作成

C 言語でアルゴリズムを作成してください。

参照 使用できる関数の詳細については、以下を参照してください。
APCS ユーザカスタムブロック（IM 33J15U20-01JA）

重要 CRT の使用を推奨されていない関数を用いた場合、コンパイル時に警告が表示されます。警告を取り除くには、推奨されている代わりの関数に置き換える必要があります。
警告の表示はプリプロセッサの定義を行うことで無効にすることもできます。_CRT_NONSTDC_NO_DEPRECATE を定義すると POSIX が使用されなくなったことを、_CRT_SECURE_NO_WARNINGS を定義するとセキュリティが弱いことを知らせる警告を無効にすることができます。_s 付きの関数はセキュリティ強化版です。警告を無効にしても、その警告の原因になった問題はそのまま存在します。警告は有効にしたままにし、新しい _s 付きの関数を使用することを推奨します。

2.2.2 APCSシミュレータによるデバッグ操作概要

作成したユーザカスタムアルゴリズムを Microsoft Visual C++、およびテスト機能を使用してデバッグすることができます。

■ デバッグ操作手順概要

1. 制御ドローイングビルダで、デバッグしたいユーザカスタムアルゴリズムを使用するユーザカスタムブロックを作成し、デバッグ用の制御ドローイングを作成します。
2. テスト機能でバーチャル HIS および、APCS シミュレータを起動します。
   FCS シミュレータが必要な場合は、FCS シミュレータも起動します。
3. 一旦、テスト機能で動作している APCS シミュレータを停止します。
4. Microsoft Visual C++ のデバッグ機能で、APCS シミュレータをデバッグ起動します。
5. ユーザカスタムアルゴリズムのソースコード上にブレークポイントを設定し、Microsoft Visual C++ のデバッグ機能でデバッグを行います。

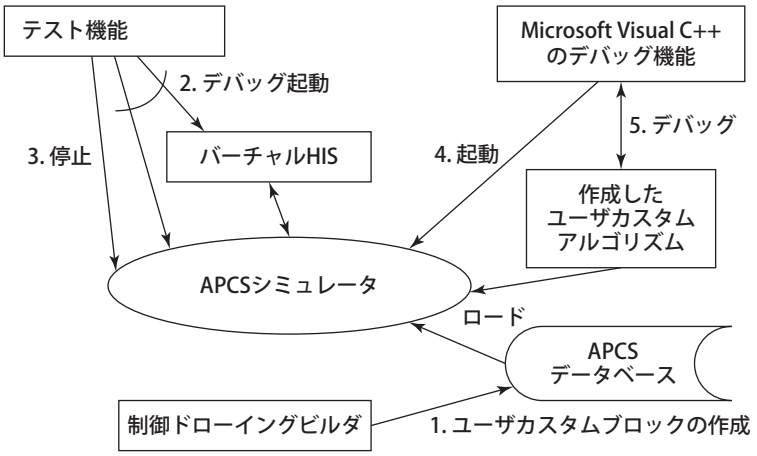


図　操作手順と各機能関連図

重要

- APCS シミュレータが本機能でデバッグ起動された場合、APCS シミュレータの下記の機能は動作しません。
  - ユーザカスタムアルゴリズムのタイムアウト監視機能（デバッグのため、実行を停止しているとタイムアウトが発生するため）
  - 例外発生時の処理（例外が発生した場合、問題個所がすぐわかるように Microsoft Visual C++ 上で異常個所を表示させるため）
- デバッグ途中でオンラインメンテナンスは行わないでください。デバッグ中は、APCS シミュレータが停止状態となっているため、オンラインメンテナンスの処理を行うことができません。

参照 デバッグ操作の詳細については、以下を参照してください。
「3.3 ユーザカスタムアルゴリズムのデバッグ」

2.2.3 バーチャルテスト機能による動作テスト概要

前項でデバッグしたユーザカスタムアルゴリズムをシステムビューで APCS へ登録後、テスト機能を用いてユーザカスタムアルゴリズムの総合的な動作確認を行ってください。
ここでは動作確認が主体です。その他、Microsoft Visual C++ のアタッチデバッグ機能を利用してデバッグを行うこともできます。

■ 動作テスト手順概要

1. システムビューでデバッグの完了したユーザカスタムアルゴリズムを、APCS へ登録します。
2. テスト機能でバーチャル HIS および、APCS シミュレータを起動します。
   FCS シミュレータが必要な場合は、FCS シミュレータも起動します。
3. バーチャル HIS 機能およびテスト機能を利用して、動作テストを行ってください。
4. ここで問題が発生し、この状態でデバッグしたい場合は、Microsoft Visual C++ のプロセスアタッチデバッグ機能を利用してデバッグを行ってください。

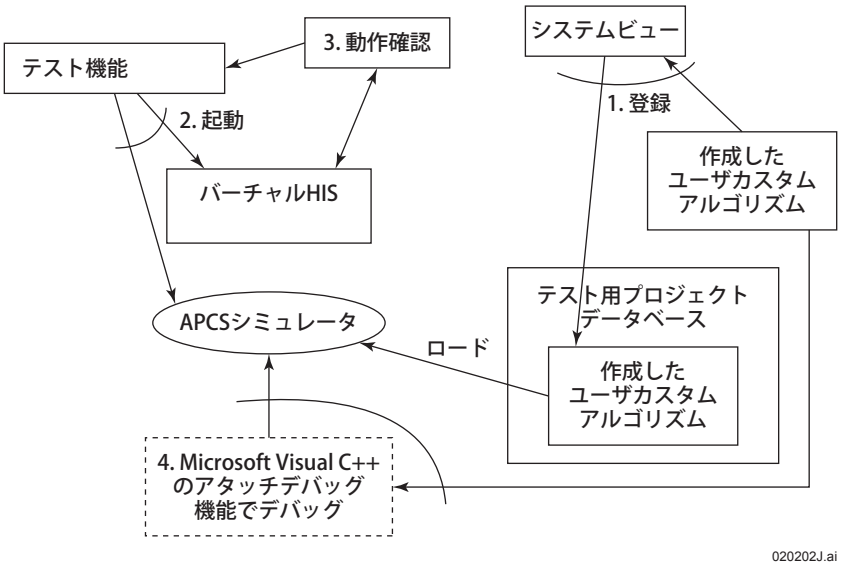


図　動作テスト時の操作手順と各機の関連図

重要 Microsoft Visual C++ のプロセスアタッチデバッグ機能を行った場合や、オンラインメンテナンス実行時に、タイムアウト監視および、例外処理は行われません。

参照 動作テストの詳細については、以下を参照してください。
「3.5 テスト機能での動作確認」

2.2.4 実機APCS上での動作テスト

前項での動作確認後、APCS の実機上でも動作確認を必ず行ってください。

3. ユーザカスタムアルゴリズム（UCA）の作成手順詳細

本章では、ユーザカスタムアルゴリズムの作成とデバッグの操作方法の詳細について説明します。

3.1 ユーザカスタムアルゴリズム作成時の注意事項

ユーザカスタムアルゴリズムは、Microsoft Visual C++を使用して作成します。MicrosoftVisual C++ではデバッグ時に使用するデバッグ版と製品版として使用するリリース版を作成することができますが、ユーザカスタムアルゴリズムの作成においては、リリース版のみを作成します（ユーザがデバッグする場合にも、製品版として利用するリリース版を用いて行います）。

また、Microsoft Visual C++で用意されているリソースファイルを使用し、バージョン情報の設定を行います。ここで設定されたバージョン情報は、ユーザカスタムアルゴリズムのバージョン情報となり、このアルゴリズムを使用しているユーザカスタムブロックのデータアイテム「バージョン情報」になります。

3.1.1 Visual Studioの初期設定

ユーザカスタムアルゴリズムはVisual Studioのうち、Microsoft Visual C++を使用して作成します。

■ 既定の環境設定

● はじめてVisual Studioを起動する場合

Visual Studio をはじめて起動すると、次の画面が表示されます。

030101J.ai

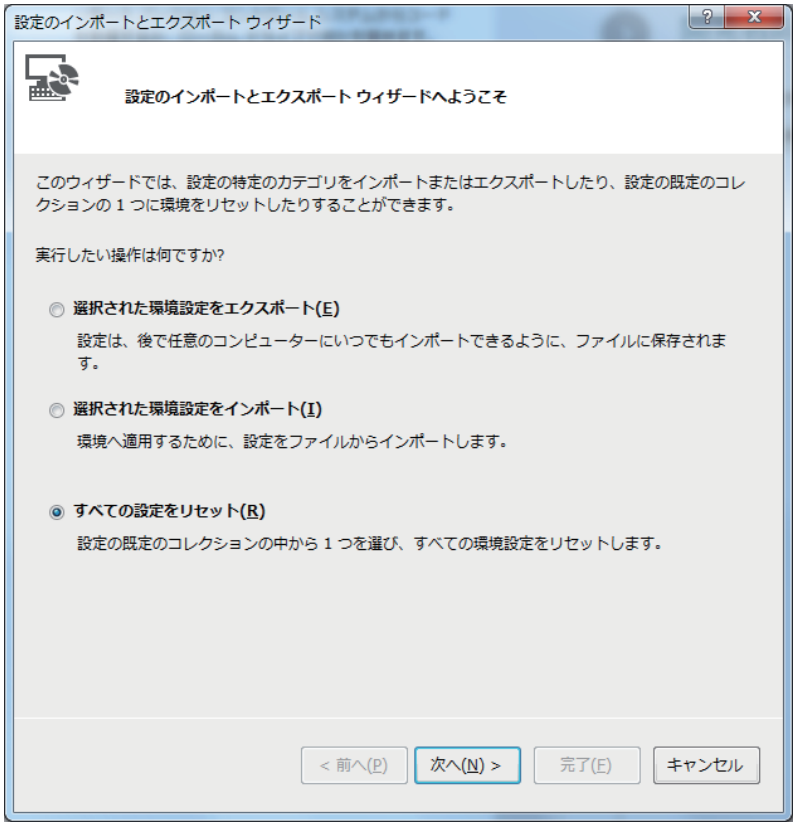
ユーザカスタムアルゴリズム開発をするときは、［開発設定］ドロップダウンリストから「Visual C++」を選択し、「Visual Studio の開始」ボタンをクリックしてください。

● Visual Studioを既に起動したことがある場合

既定の環境設定で「Visual C++」以外を選択している場合は、次の手順で「Visual C++」にリセットしてください。また、既定の環境が不明な場合も次の手順でリセットを行ってください。

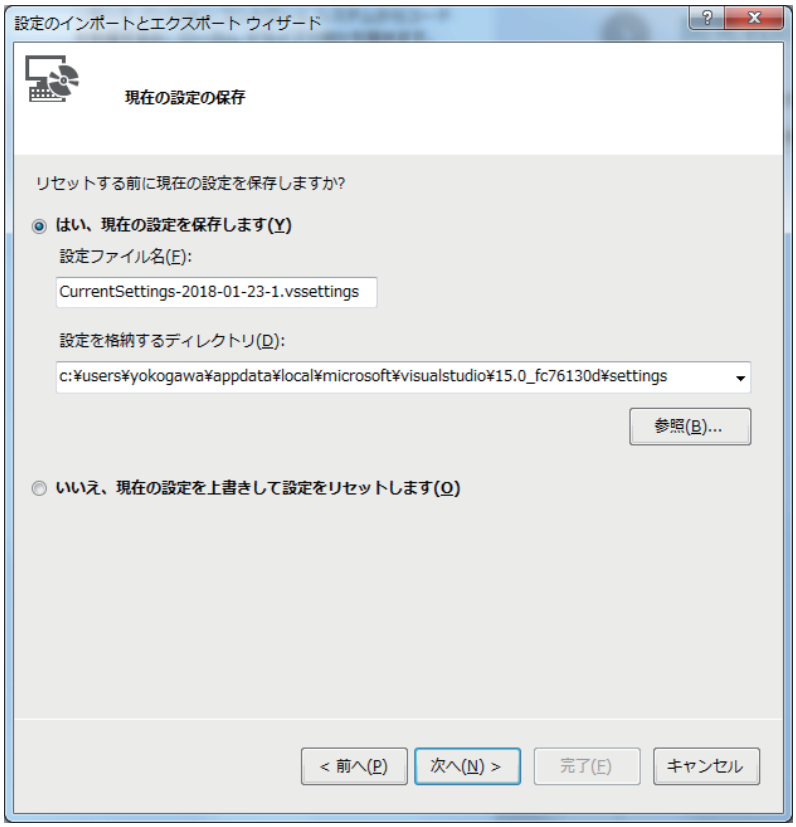
1. Visual Studio を起動してください。
2. ［ツール］メニューの［設定のインポートとエクスポート］を選択してください。
   「設定のインポートとエクスポートウィザード」が起動します。

3. ［すべての設定をリセット］を選択し、［次へ］をクリックしてください。



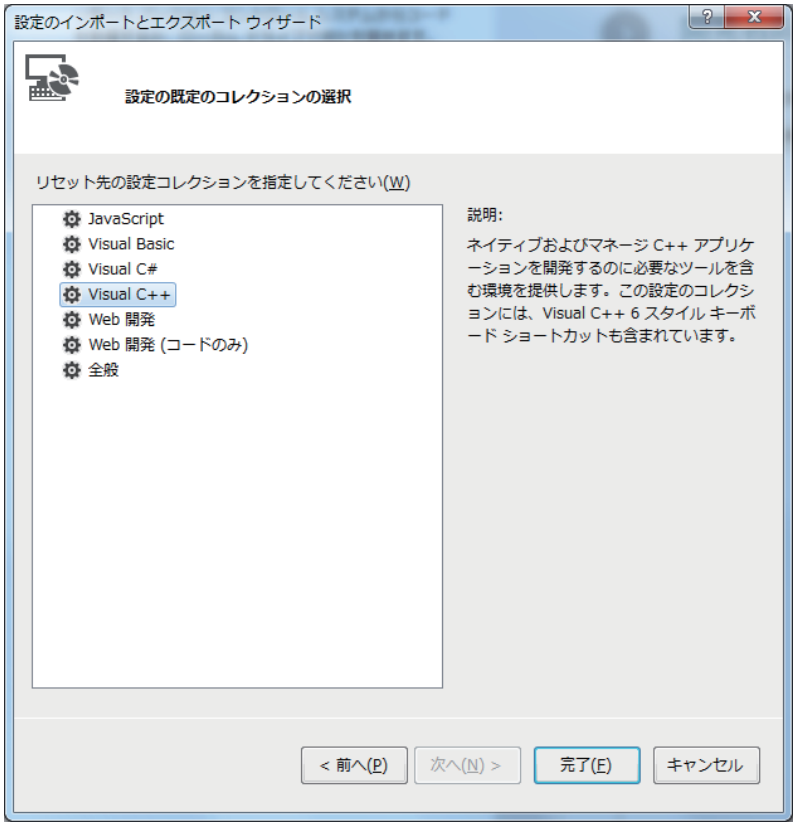
030102J.ai

4. ［はい、現在の設定を保存します］を選択し、［次へ］ボタンをクリックしてください。



030103J.ai

5. ［リセット先の設定コレクションを指定してください］で、[Visual C++] を選択し、[完了］をクリックしてください。



030104J.ai

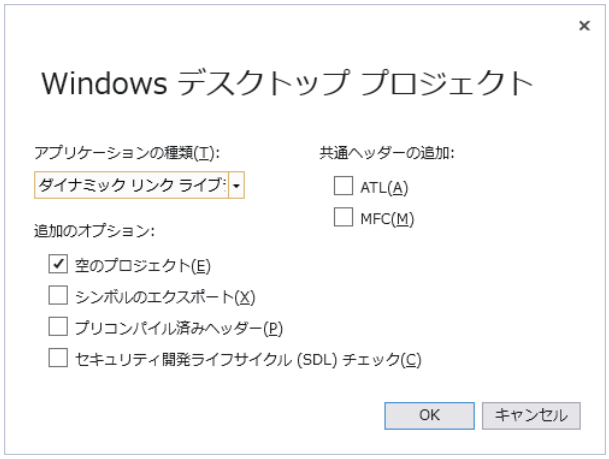
6. リセットの完了が表示されたら、［閉じる］ボタンをクリックしてください。

3.1.2 ソリューションの作成

Visual Studioを起動して、ソリューションを作成します。

■ ソリューション作成手順

1. Visual Studio を起動してください。
2. ［ファイル］メニューから［新規作成］ー［プロジェクト］を選択してください。
「新しいプロジェクト」ダイアログが表示されます。
3. ［インストール済み］ー［Visual C++］ー［Windows デスクトップ］を選択し、右ペインの［Windows デスクトップウィザード］を選択（反転表示）してください。
4. ［名前］ボックスに、ユーザカスタムアルゴリズム名を入力してください。
プロジェクト名には 16 文字以内の英数字またはアンダーバー “_” を使用できますが、先頭文字に数字を使用することはできません。
5. ［場所］ボックスに、プロジェクトを作成するフォルダの場所を入力してください。
6. ［ソリューションのディレクトリを作成］チェックボックスをクリアし、［OK］ボタンをクリックしてください。
「Windows デスクトッププロジェクト」ダイアログが表示されます。



030105J.ai

7. 設定を次のように行ってください。
- ［アプリケーションの種類］ドロップダウンリストから［ダイナミック リンク ライブラリ（.dll）］を選択してください。
- ［追加のオプション］の［空のプロジェクト］チェックボックスを選択してください。
- ［追加のオプション］の［プリコンパイル済みヘッダー］チェックボックスをクリアしてください。
- ［追加のオプション］の［セキュリティ開発ライフサイクル（SDL）チェック］チェックボックスをクリアしてください。

8. ［OK］ボタンをクリックしてください。

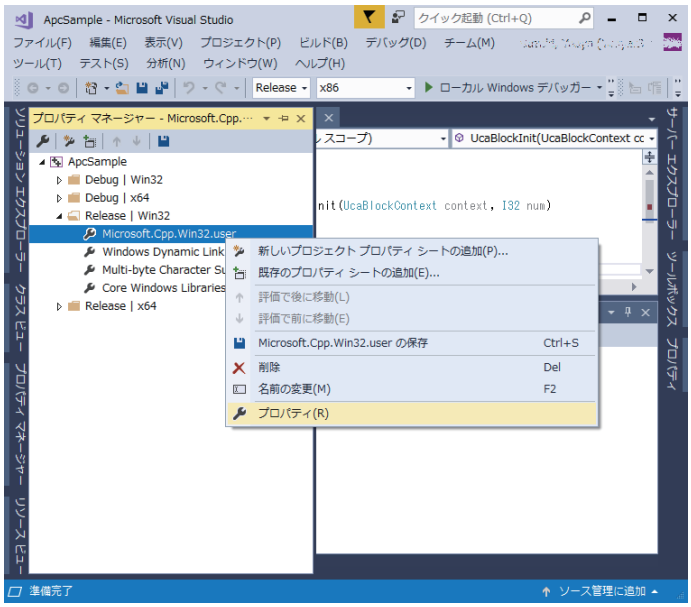
以上でソリューションの作成は終了です。

3.1.3 ディレクトリパスの設定

Microsoft Visual C++ 上でユーザカスタムアルゴリズムを開発するには、インクルードファイルとライブラリファイルのディレクトリパスを設定します。
ここで設定したディレクトリパスは、ソリューションごとに管理されます。したがって、あるソリューションで設定したディレクトリパスは、別のソリューションを起動した場合でも変更されません。

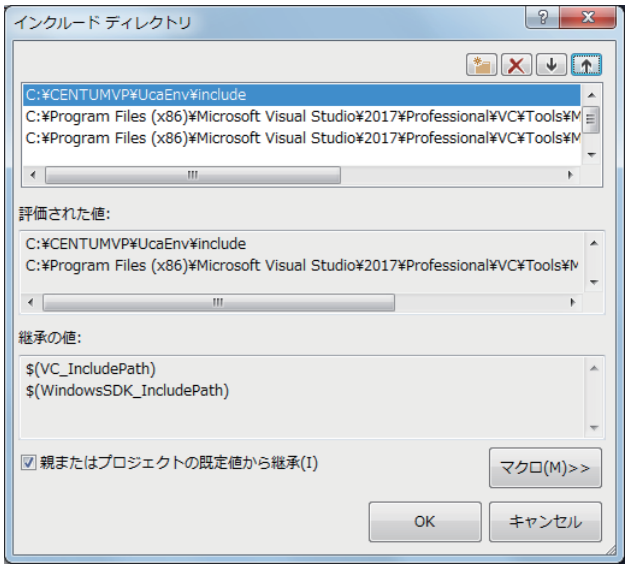
■ ディレクトリパスの設定手順

1. ［表示］メニューから［プロパティ マネージャー］を選択してください。
2. ［プロパティ マネージャー］の［Release | Win32］のユーザカスタムアルゴリズム開発で使用しているユーザプロパティシートを右クリックし、［プロパティ］を選択してください。
「プロパティページ」ダイアログが表示されます。



030106J.ai

3. 左ペインの［共通プロパティ］－［VC++ ディレクトリ］を選択してください。
4. 右ペインの［全般］－［インクルードディレクトリ］を選択し、［編集］をクリックしてください。
   「インクルードディレクトリ」ダイアログが表示されます。



030107J.ai

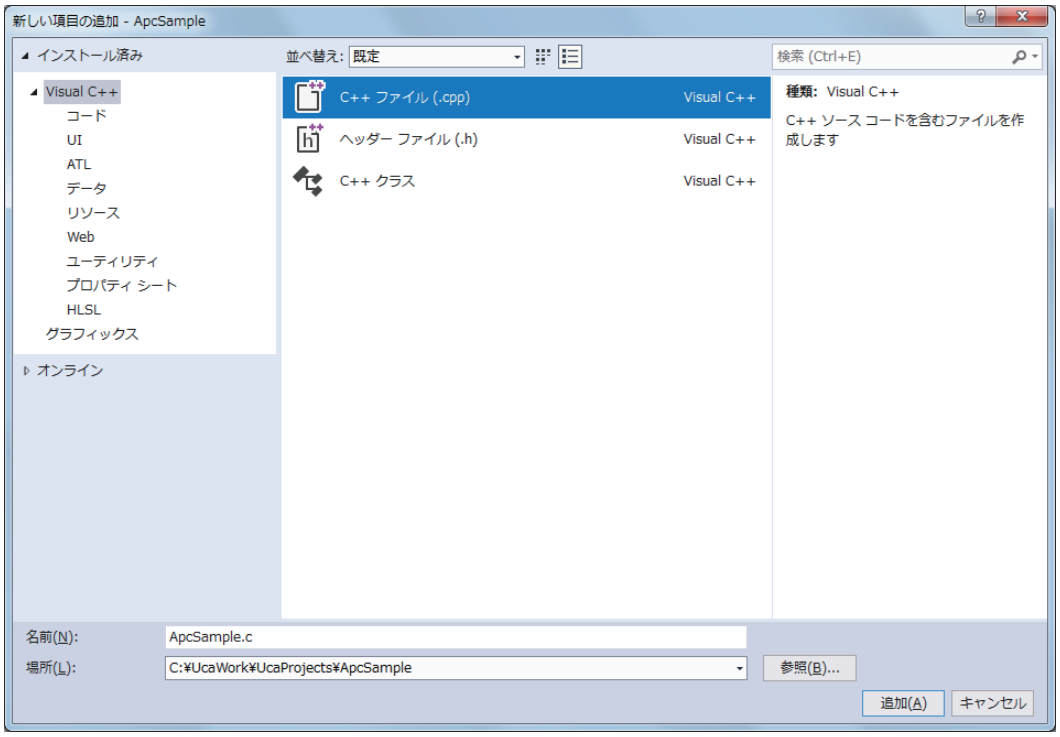
5. 一番上に［＜インストール先＞¥UcaEnv¥include］を設定してください。
6. ［OK］ボタンをクリックしてください。
7. 「プロパティページ」ダイアログの左ペインの、［共通プロパティ］－［VC++ ディレクトリ］を選択してください。
8. 右ペインの［全般］－［ライブラリディレクトリ］を選択し、［編集］をクリックしてください。
   「ライブラリディレクトリ」ダイアログが表示されます。
9. 一番上に［＜インストール先＞¥UcaEnv¥dll］を設定してください。
10. ［OK］ボタンをクリックしてください。
11. 「プロパティページ」ダイアログの［OK］ボタンをクリックしてください。

3.1.4 プロジェクトの設定

ここでは、プロジェクトの設定方法について説明します。
プロジェクトの設定をする前にソースファイルをプロジェクトに追加してください。
ソースファイルが存在しないと、以降の「C/C++」に関する設定ができません。

■ ソースファイルの追加手順

1. ［プロジェクト］メニューの［新しい項目の追加］を選択してください。
「新しい項目の追加」ダイアログが表示されます。

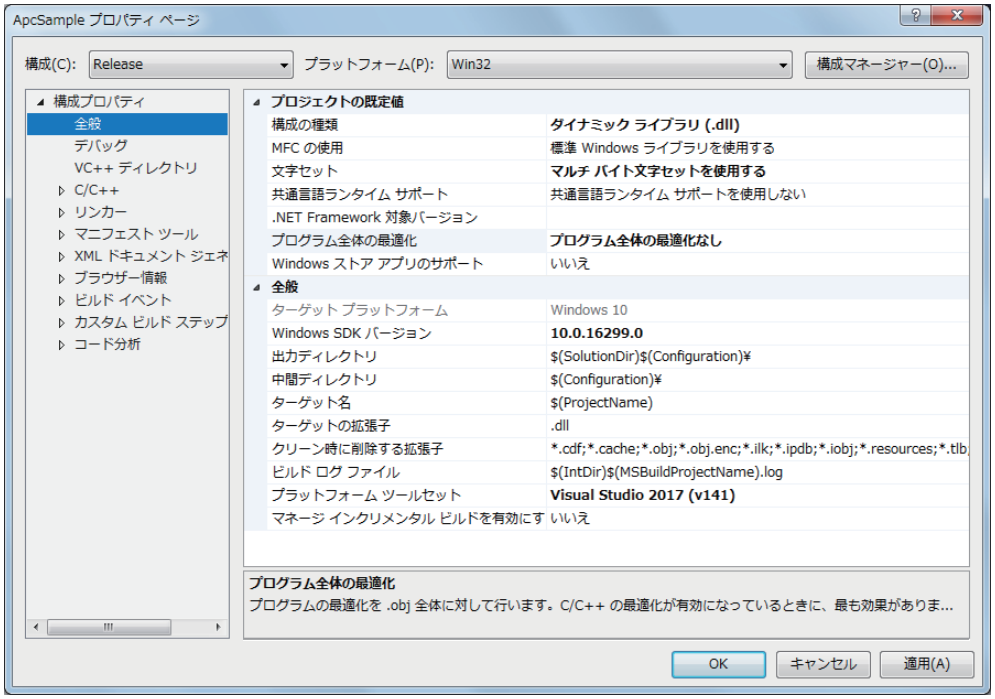


030108J.ai

2. 左ペインの［インストール済み］ー［Visual C++］を選択したあとで、右ペインの［C++ファイル］を選択してください。
3. ［名前］ボックスにソースコードのファイル名を入力し、拡張子を .c としてください。
4. ［追加］ボタンをクリックしてください。

■ プロジェクトの設定手順

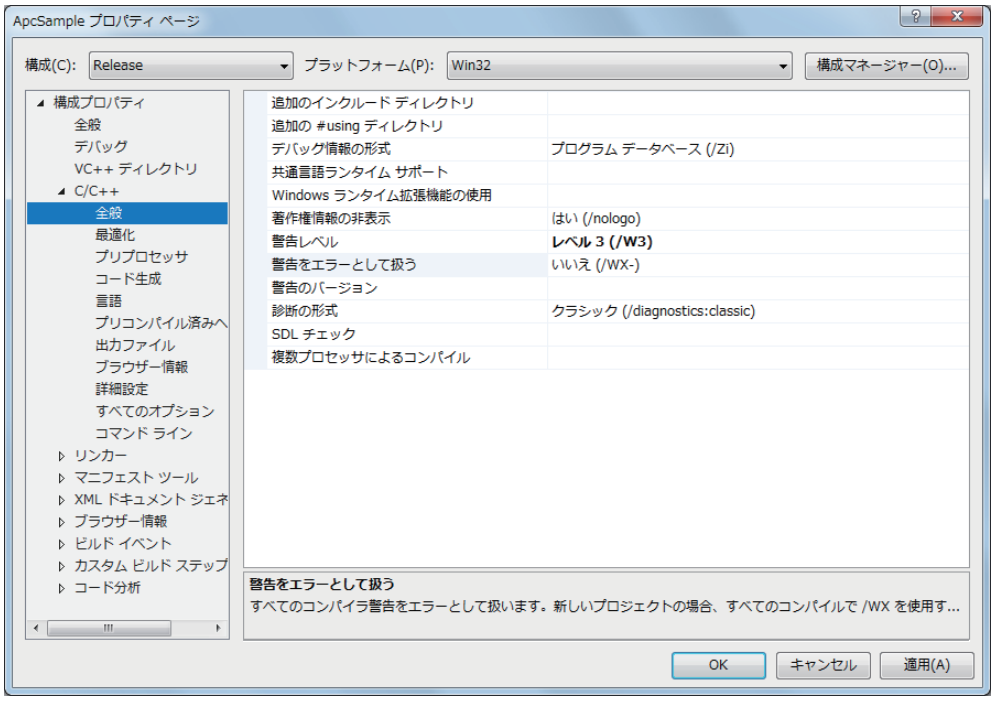
1. ［プロジェクト］メニューから［< プロジェクト名 > のプロパティ］を選択してください。
「プロパティページ」ダイアログが表示されます。



030109J.ai

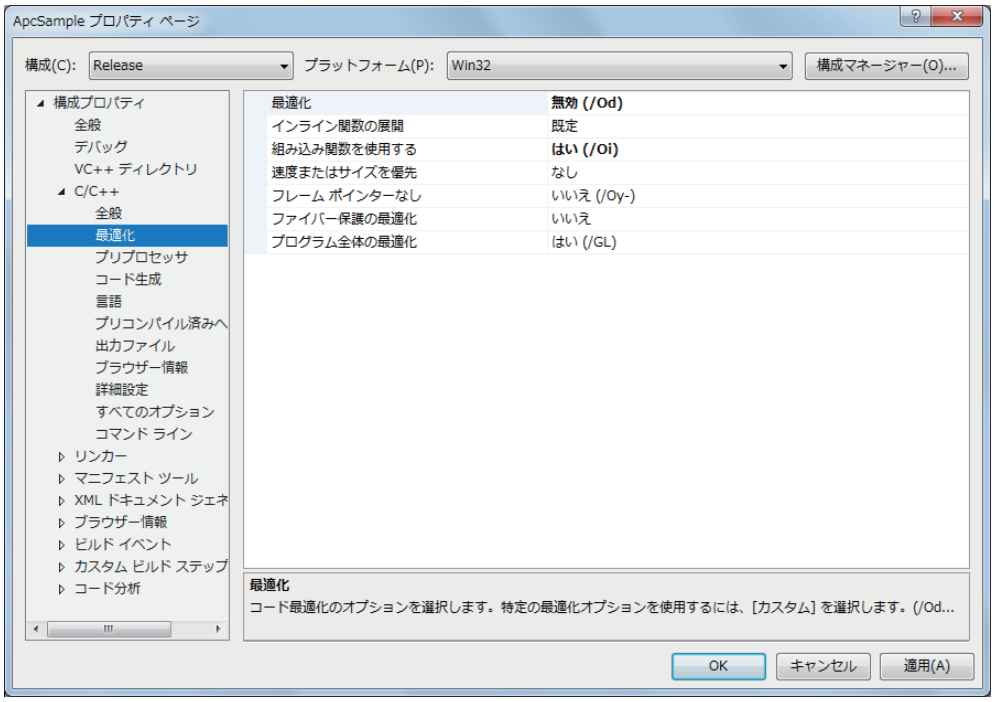
2. ［構成プロパティ］を選択してください。
3. ［構成］ドロップダウンリストから［Release］を選択してください。
4. ［構成プロパティ］－［全般］の設定を次にように行ってください。
   - ［文字セット］に［マルチバイト文字セットを使用する］を設定してください。
   - ［プログラム全体の最適化］に［プログラム全体の最適化なし］を設定してください。

5. ［構成プロパティ］－［C/C++］－［全般］の設定を次のように行ってください。
   - ［デバッグ情報の形式］に［プログラム データベース］を設定してください。
   - ［警告レベル］に［レベル 3（/W3）］を設定してください。
   - ［警告をエラーとして扱う］に［いいえ（/WX-）］を設定してください。



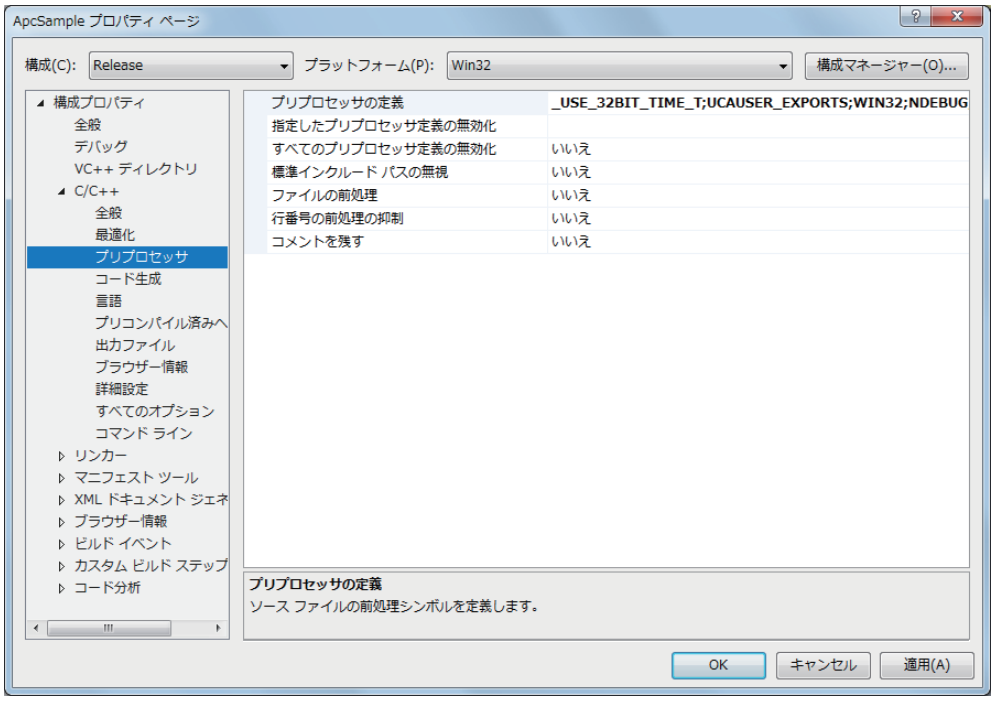
030110J.ai

6. ［構成プロパティ］－［C/C++］－［最適化］の設定を次のように行ってください。
   - ［最適化］に［無効（/Od）］を設定してください。
   - ［インライン関数の展開］に［既定］を設定してください。



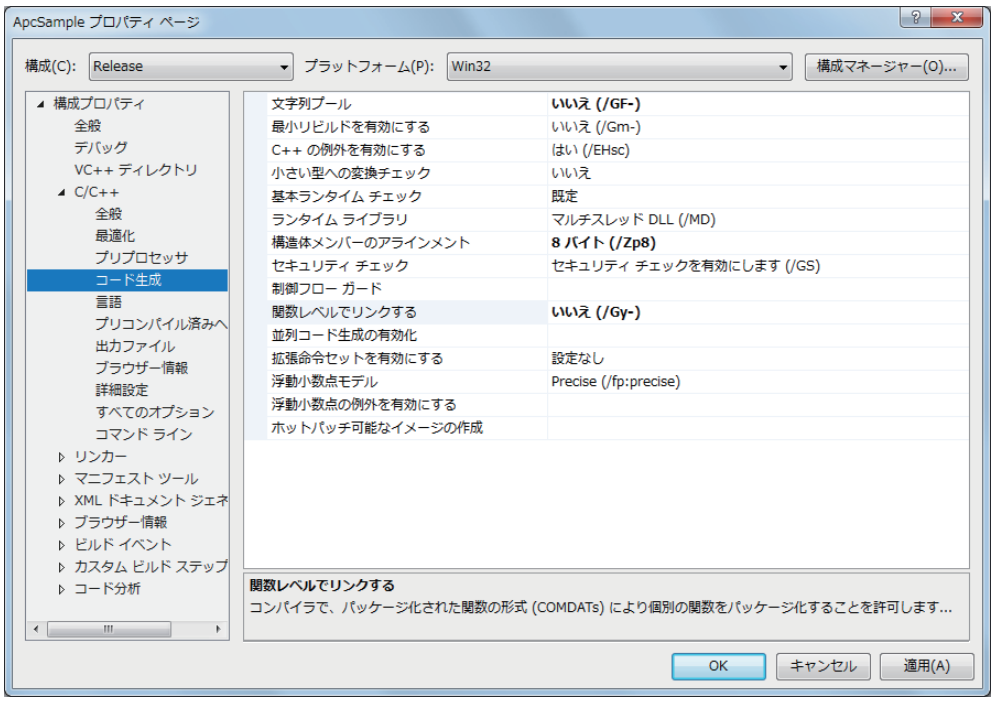
030111J.ai

7. ［構成プロパティ］－［C/C++］－［プリプロセッサ］の設定を次のように行ってください。
   - ［プリプロセッサの定義］に［_USE_32BIT_TIME_T;UCAUSER_EXPORTS］を追加してください。



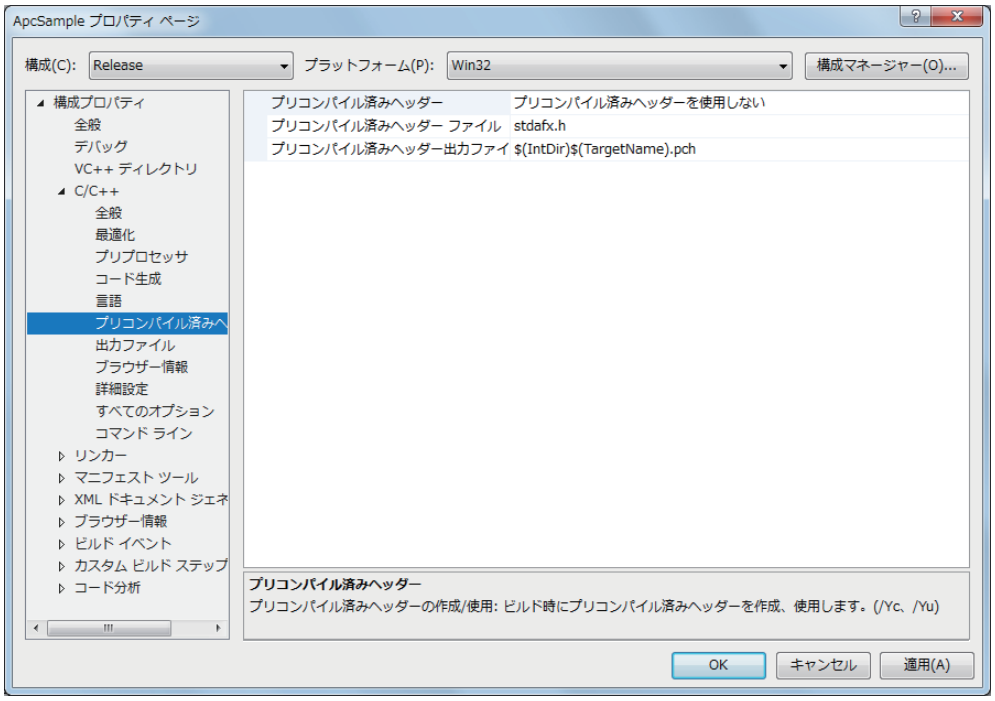
030112J.ai

8. ［構成プロパティ］－［C/C++］－［コード生成］の設定を次のように行ってください。
   - ［文字列プール］に［いいえ（/GF-）］を設定してください。
   - ［最小リビルドを有効にする］に［いいえ（/Gm-）］を設定してください。
   - ［ランタイムライブラリ］に［マルチスレッド DLL（/MD）］を設定してください。
   - ［構造体メンバーのアラインメント］に［8 バイト（/Zp8）］を設定してください。
   - ［関数レベルでリンクする］に［いいえ（/Gy-）］を設定してください。



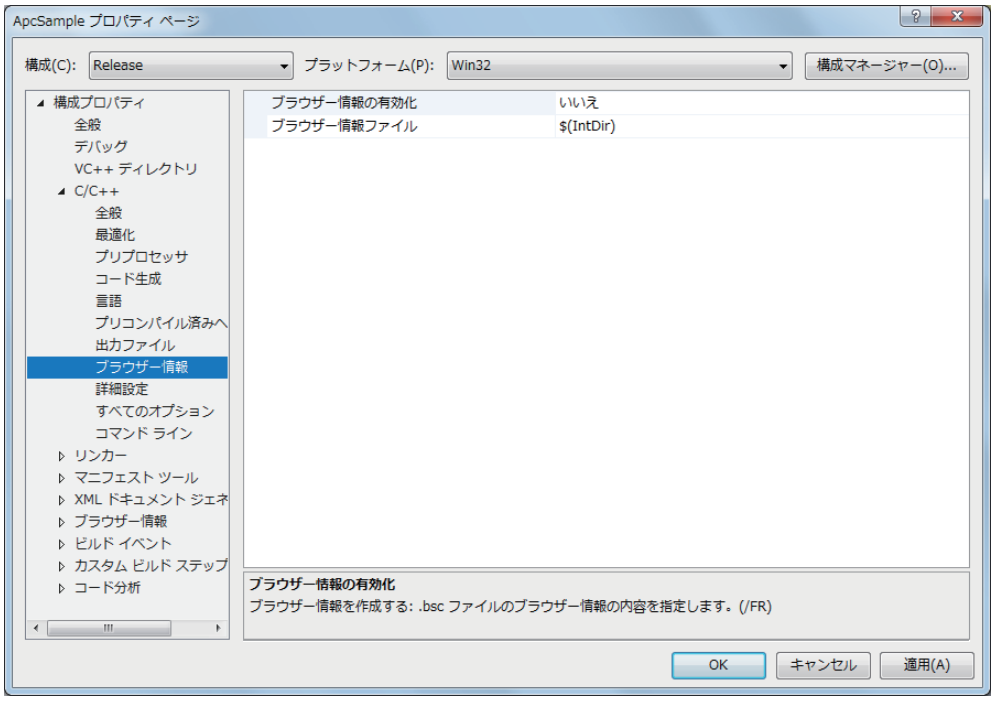
030113J.ai

9. ［構成プロパティ］－［C/C++］－［プリコンパイル済みヘッダー］の設定を次のように行ってください。
   - ［プリコンパイル済みヘッダー］に［プリコンパイル済みヘッダーを使用しない］を設定してください。



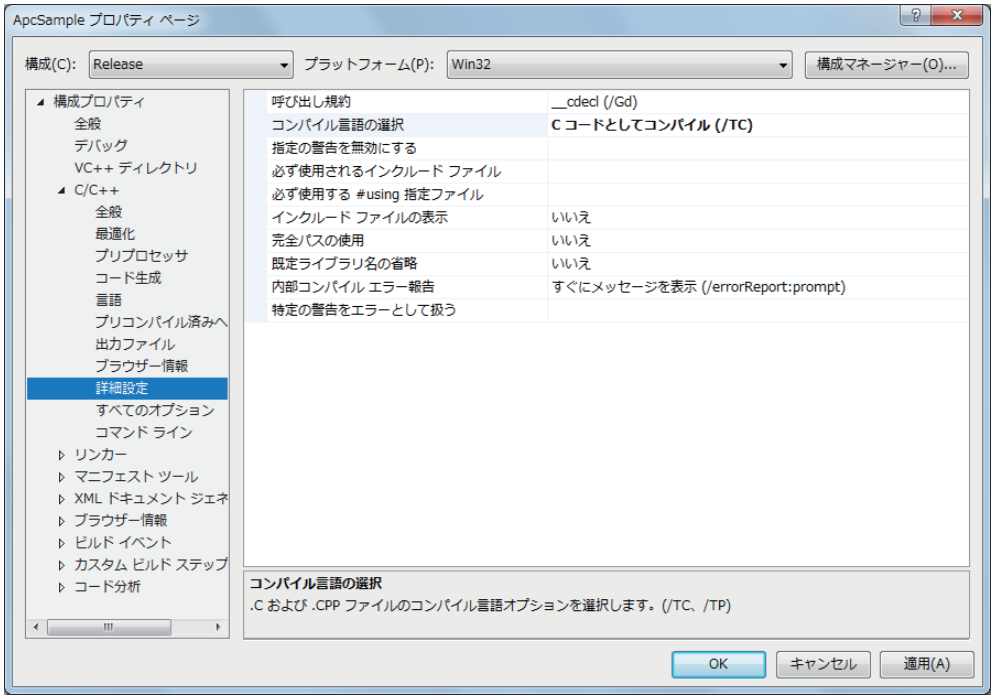
030114J.ai

10. ［構成プロパティ］－［C/C++］－［ブラウザー情報］の設定を次のように行ってください。
   - ［ブラウザー情報の有効化］に［いいえ］を設定してください。



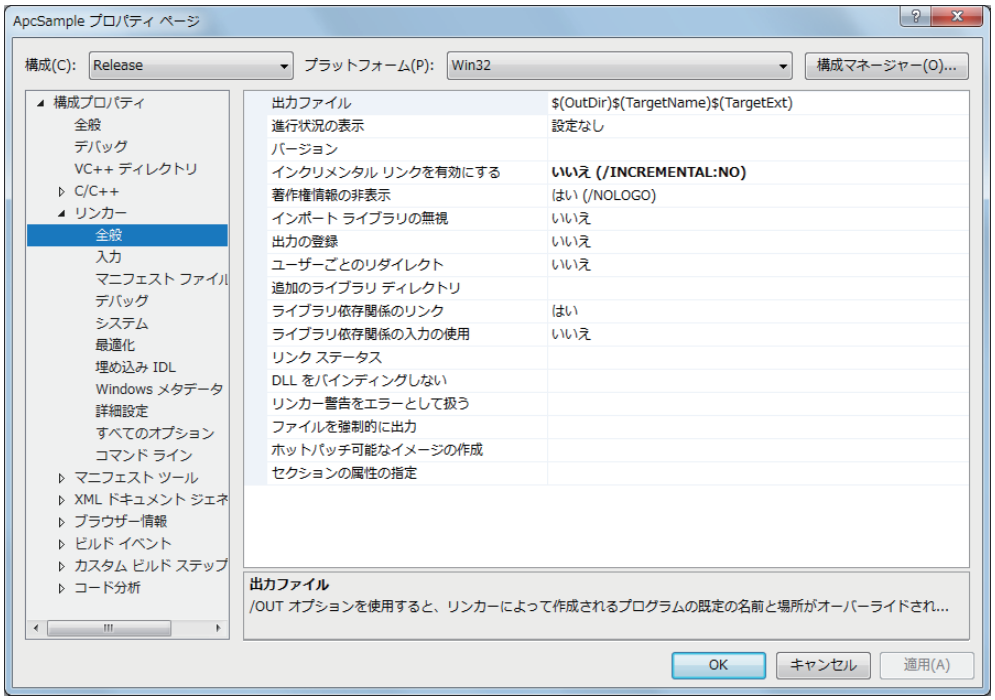
030115J.ai

11. ［構成プロパティ］－［C/C++］－［詳細設定］の設定を次のように行ってください。
    - ［呼び出し規約］に［__cdecl（/Gd）］を設定してください。
    - ［コンパイル言語の選択］に［C コードとしてコンパイル（/TC）］を設定してください。



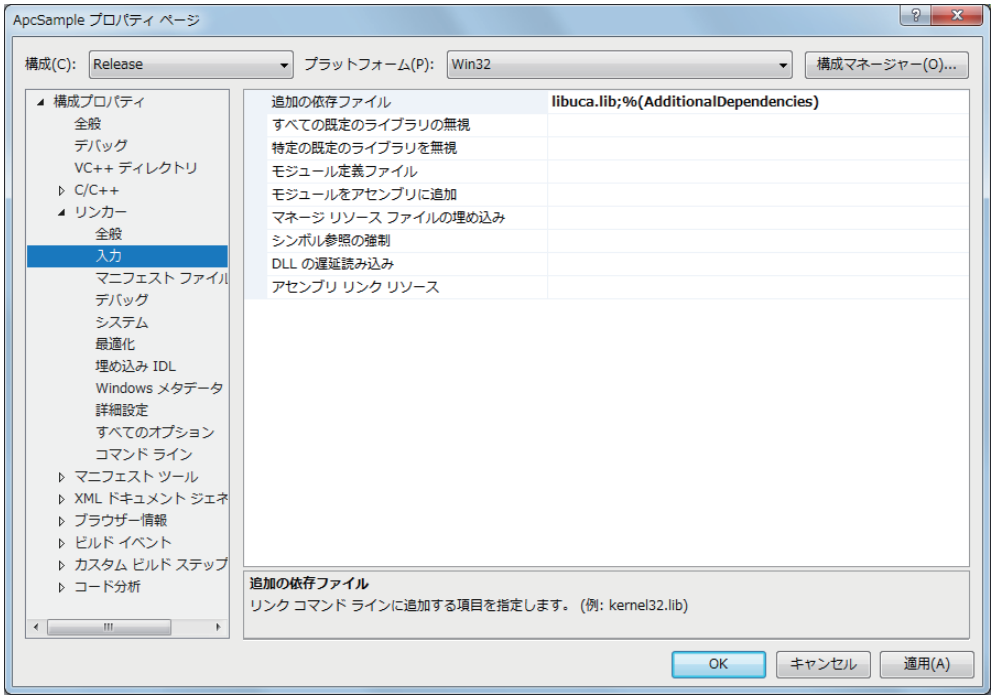
030116J.ai

12. ［構成プロパティ］－［リンカー］－［全般］の設定を次のように行ってください。
    - ［出力ファイル］に［$(OutDir)$(TargetName)$(TargetExt)］を設定してください。
    - ［インクリメンタル リンクを有効にする］に［いいえ（/INCREMENTAL:NO）］を設定してください。



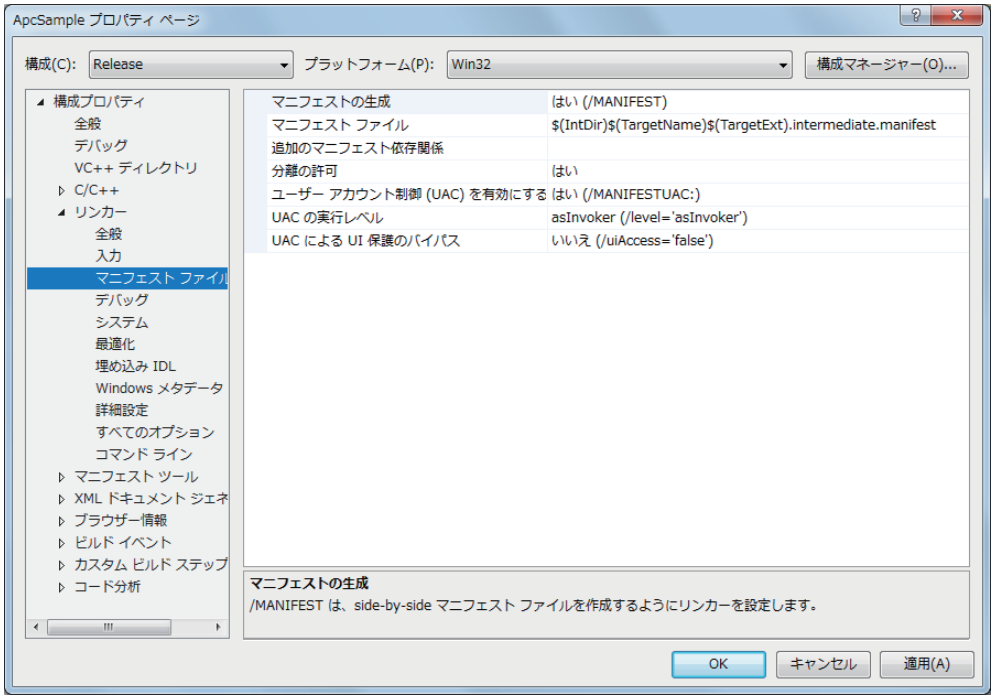
030117J.ai

13. ［構成プロパティ］ー［リンカー］ー［入力］の設定を次のように行ってください。
    - ［追加の依存ファイル］に［libuca.lib］を追加してください。



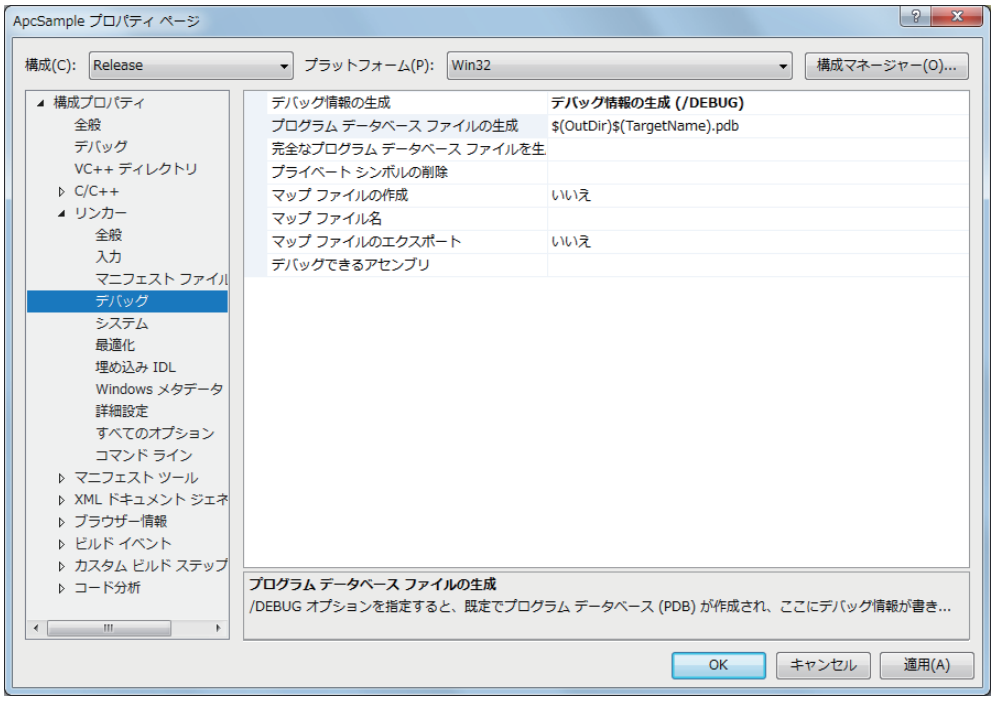
030118J.ai

14. ［構成プロパティ］ー［リンカー］ー［マニフェスト ファイル］の設定を次のように行ってください。
    - ［マニフェストの生成］に［はい（/MANIFEST)］を設定してください。



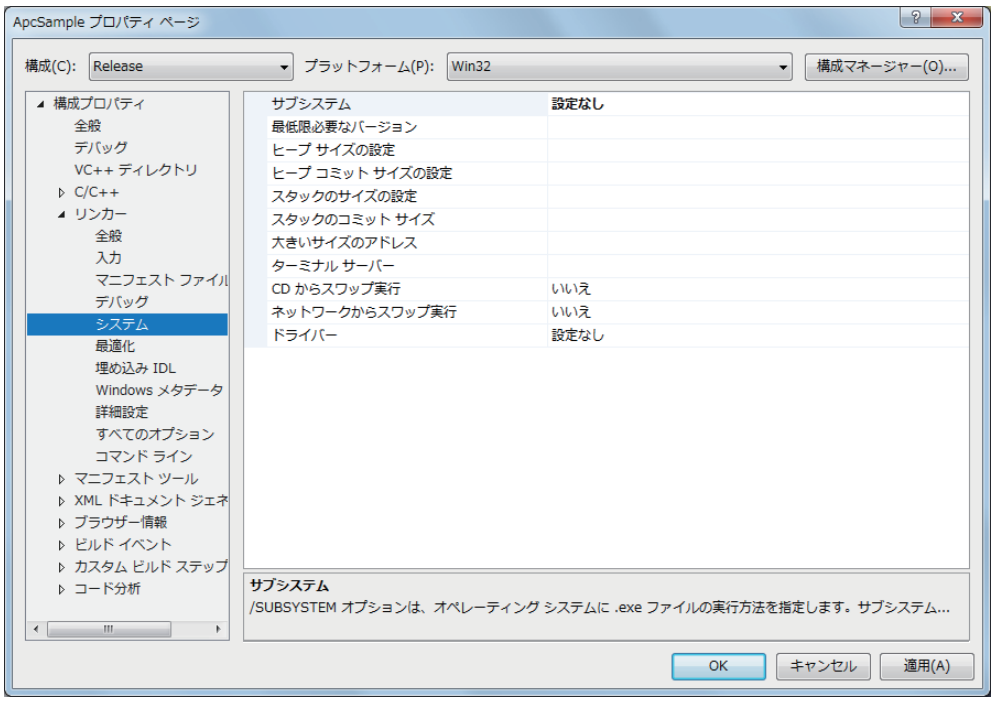
030119J.ai

15. ［構成プロパティ］－［リンカー］－［デバッグ］の設定を次のように行ってください。
    - ［デバッグ情報の生成］に［デバッグ情報の生成（/DEBUG)］を設定してください。
    - ［プログラム データベース ファイルの生成］に［$(OutDir)$(TargetName).pdb］を設定してください。



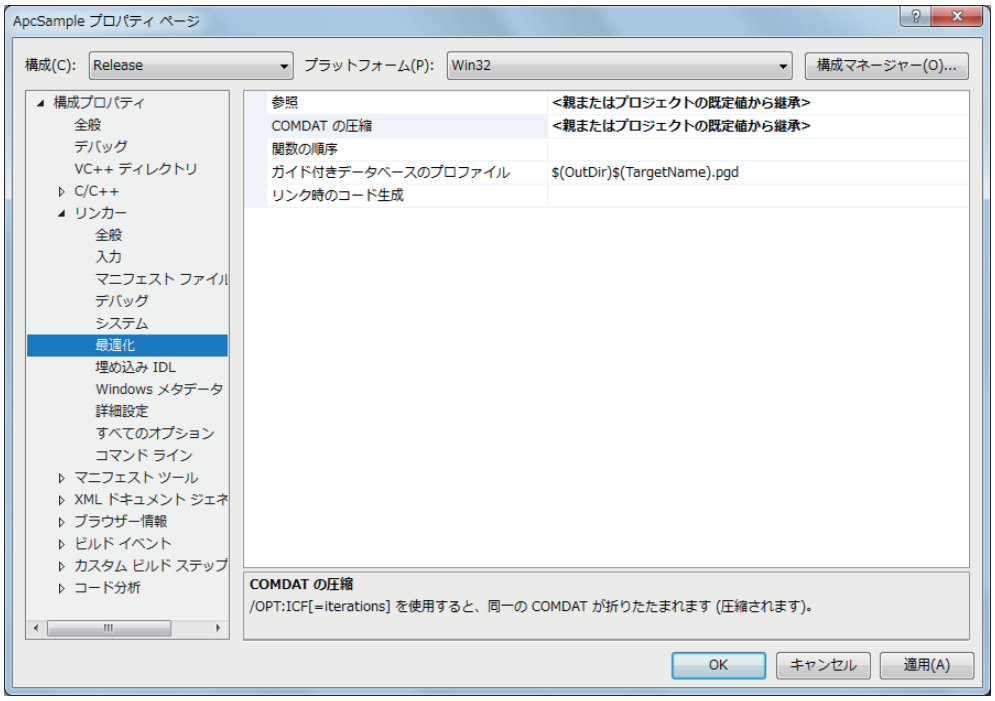
030120J.ai

16. ［構成プロパティ］－［リンカー］－［システム］の設定を次のように行ってください。
    - ［サブシステム］に［設定なし］を設定してください。



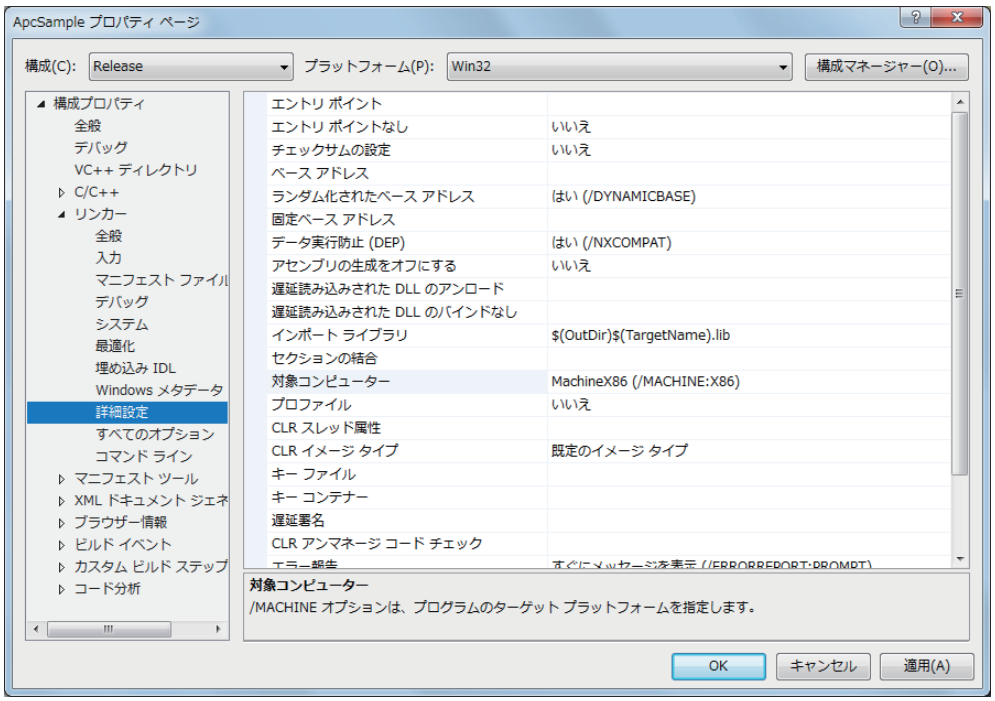
030121J.ai

17. ［構成プロパティ］－［リンカー］－［最適化］の設定を次のように行ってください。
    - ［参照］に［< 親またはプロジェクトの既定値から継承 >］を設定してください。
    - ［COMDAT の圧縮］に［< 親またはプロジェクトの既定値から継承 >］を設定してください。



030122J.ai

18. ［構成プロパティ］－［リンカー］－［詳細設定］の設定を次のように行ってください。
    - ［対象コンピューター］に［MachineX86（/MACHINE:X86)］を設定してください。



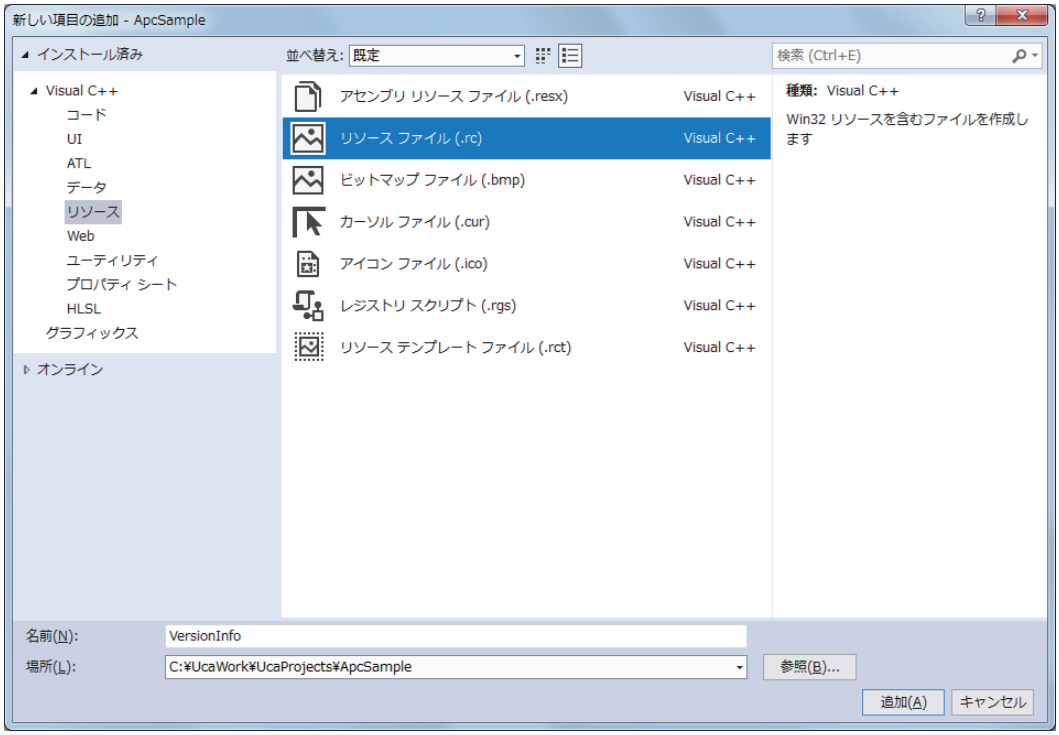
030123J.ai

3.1.5 バージョン情報の設定

ユーザカスタムアルゴリズムのバージョン情報を設定します。

■ バージョン情報設定手順

1. ［プロジェクト］メニューの［新しい項目の追加］を選択してください。
「新しい項目の追加」ダイアログが表示されます。



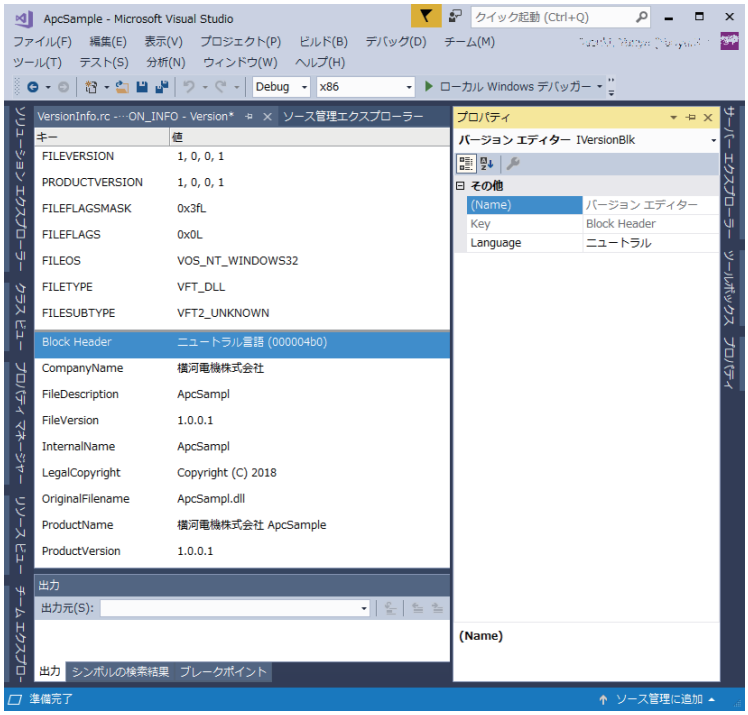
030124J.ai

2. 左ペインの［インストール済み］－［Visual C++］－［リソース］を選択したあとで、右ペインの［リソース ファイル（.rc）］を選択してください。
3. ［名前］ボックスにファイル名を入力し、［追加］ボタンをクリックしてください。
4. リソースビューでリソースファイル名を右クリックし、メニューから［リソースの追加］を選択してください。
「リソースの追加」ダイアログが表示されます。

030125J.ai

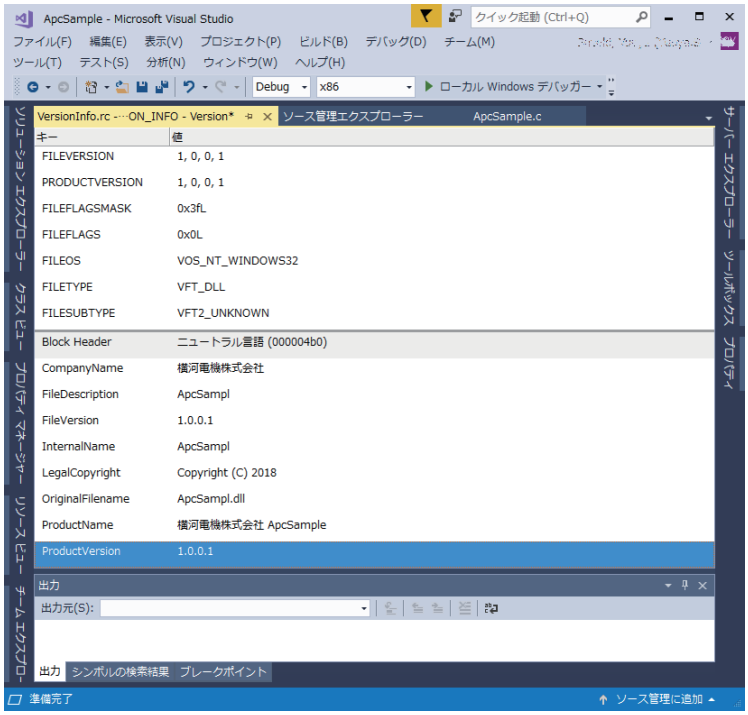
5. ［リソースの種類］から［Version］を選択し、［新規作成］ボタンをクリックしてください。

6. 左ペインの［Block Header］をダブルクリックし、プロパティウィンドウで［Language］に［ニュートラル］を設定してください。



030126J.ai

7. 同様にして［ProductName］［ProductVersion］も設定してください。



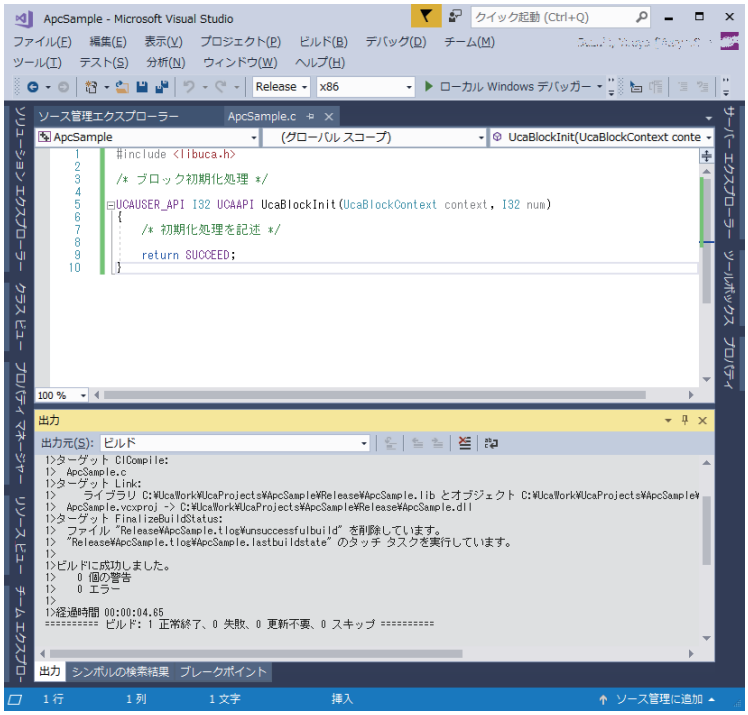
030127J.ai

3.1.6 ユーザカスタムアルゴリズムの作成

ユーザカスタムアルゴリズムの作成手順について説明します。
Microsoft Visual C++でアルゴリズムを作成してください。
ユーザカスタムアルゴリズムは、リリース版で作成します。Microsoft Visual C++ではデバッグ版も作成できますが、ユーザカスタムアルゴリズムの作成には、デバッグ版は使用しません。

■ 作成手順

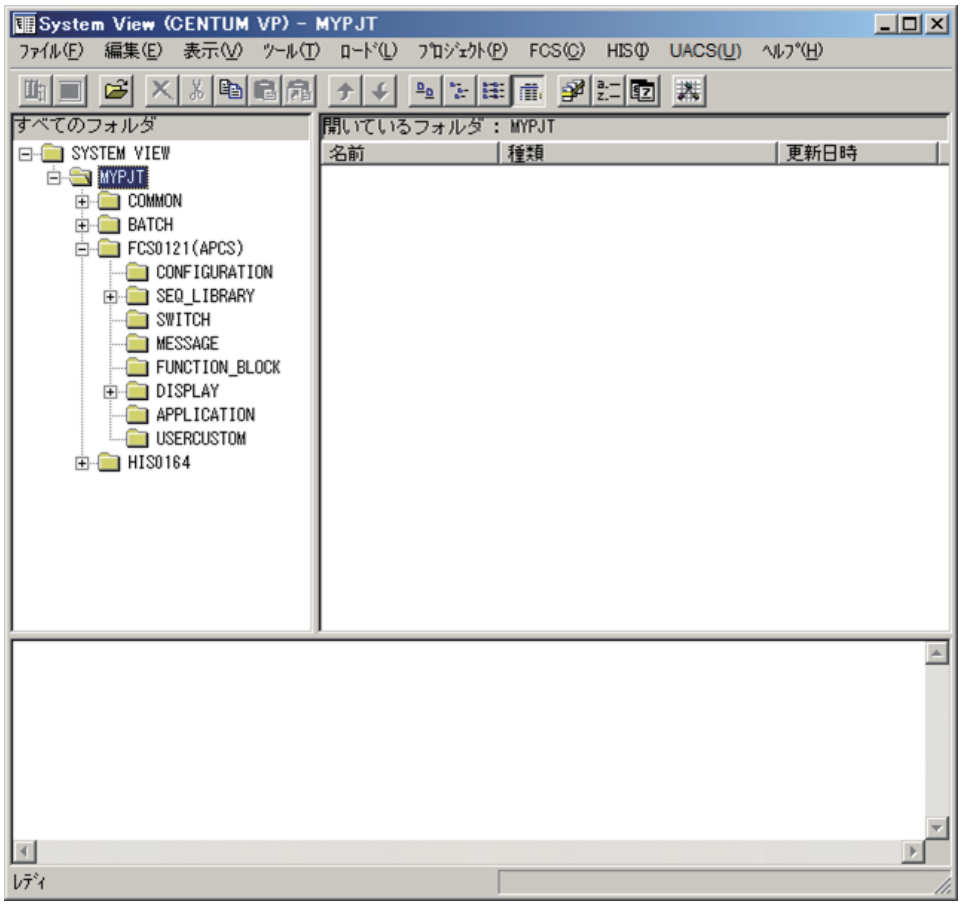
1. Microsoft Visual C++ 上で C 言語を使用し、ユーザカスタムアルゴリズムを作成してください。
2. コンパイル／ビルドを行い、リリース版の DLL を作成してください。



030128J.ai

3.2 CENTUMプロジェクトの作成

システムビューでユーザカスタムアルゴリズムのデバッグ用にプロジェクトを作成します。
FCSの新規作成で「APCS アドバンストプロセスコントロールステーション」を追加します。

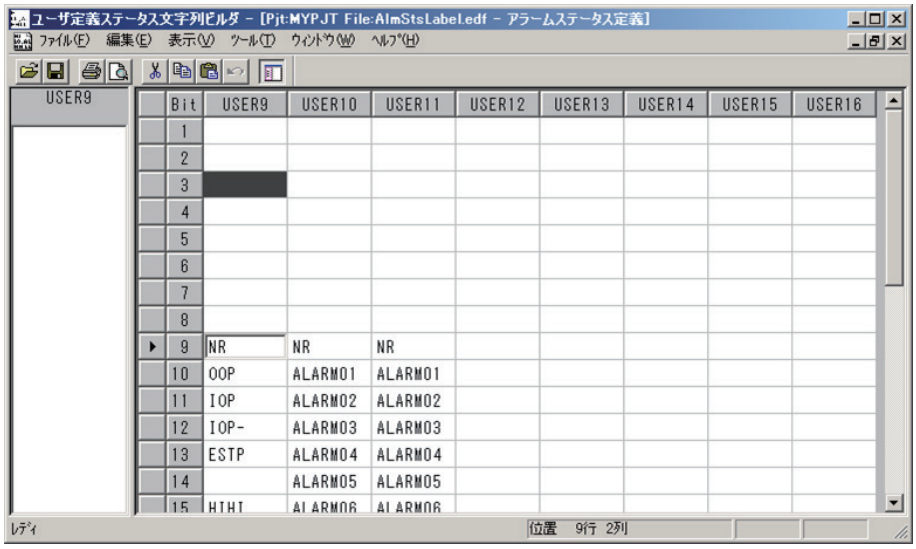


030201J.ai

3.2.1 アラームステータスの定義

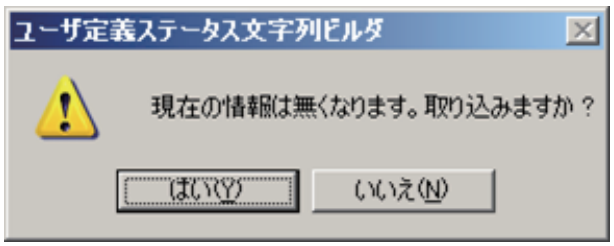
ユーザカスタムブロックのアラームステータスの初期値を定義します。

1. システムビューでプロジェクトの COMMON にある AlmStsLabel をダブルクリックしてください。
   ［ユーザ定義ステータス文字列ビルダ］が現れます。



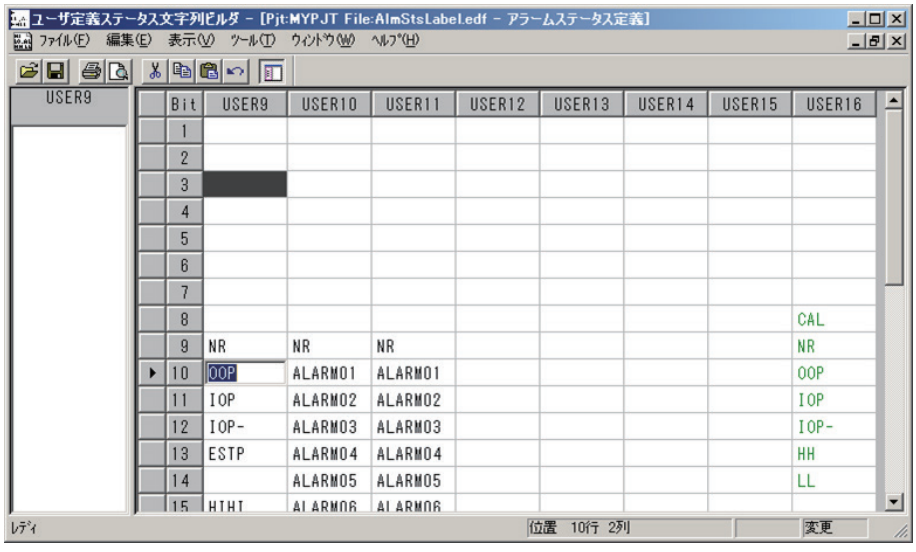
030202J.ai

2. ［編集］メニューの［ユーザカスタムブロックデフォルトの取り込み］を選択してください。
3. 次のダイアログが表示されるので、［はい］をクリックしてください。



030203J.ai

USER16 にアラームステータス定義が読み込まれます。
USER16 がユーザカスタムブロックのアラームステータス定義です。

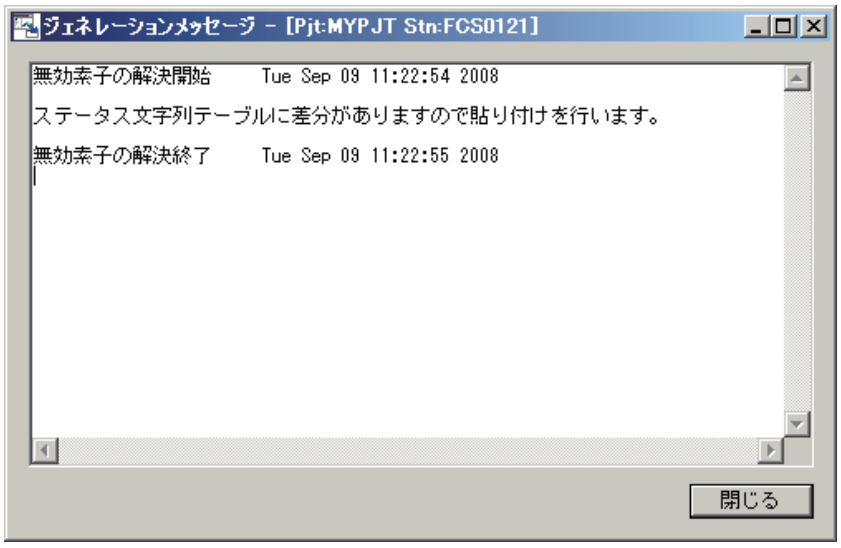


030204J.ai

4. ［ファイル］メニューの［上書き保存］を選択して上書き保存を行い、ビルダを終了してください。
5. システムビューで［プロジェクト］メニューの［無効素子の解決］を選択してください。
［FCS 状況表示ダイアログ］が表示されます。

030205J.ai

6. APCS を選択し、［開始］ボタンをクリックしてください。
次のダイアログボックスが表示され、ブロックステータス定義の変更が APCS に反映されたことが確認されます。



030206J.ai

7. ［FCS 状況表示ダイアログ］と［ジェネレーションメッセージ］を閉じてください。

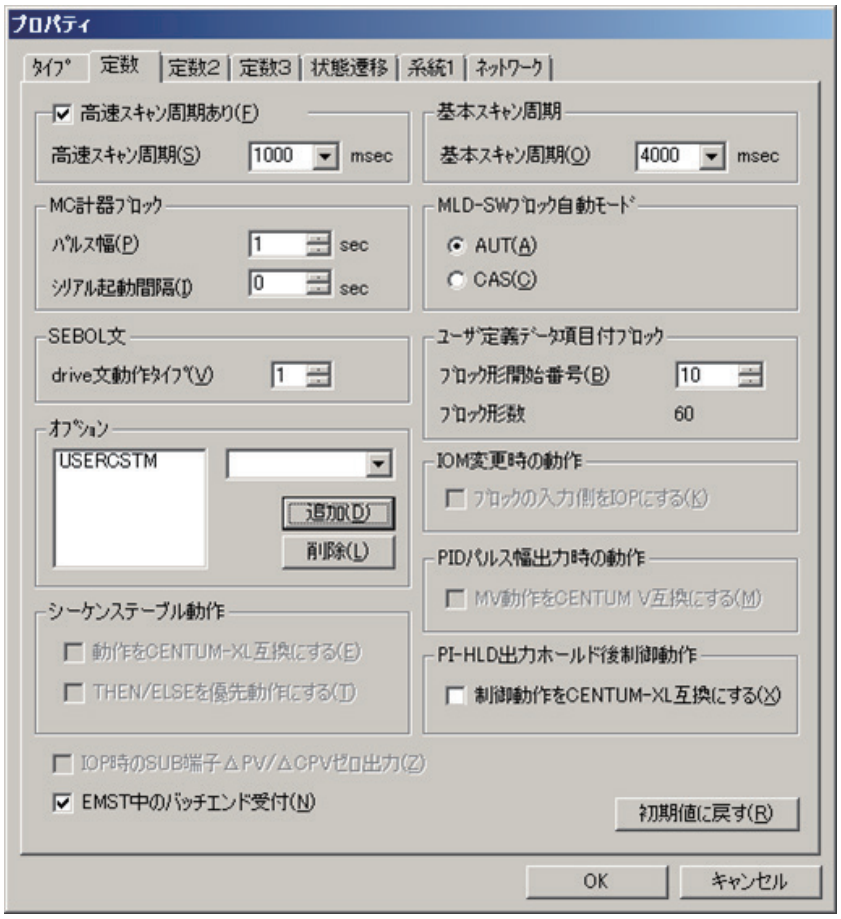
3.2.2 ユーザカスタムブロックの定義

ユーザカスタムブロックを制御ドローイングビルダで定義します。

■ APCSのプロパティの設定

ユーザカスタムブロックを使用するには、APCS のオプション設定が必要となります。

1. システムビューを起動し、APCS のプロパティを開いてください。
2. ［定数］タブを選択後、オプションの選択エリアで “USERCSTM” を選択し、［追加］ボタンをクリックしてください。



030207J.ai

3. ［OK］ボタンをクリックしてください。

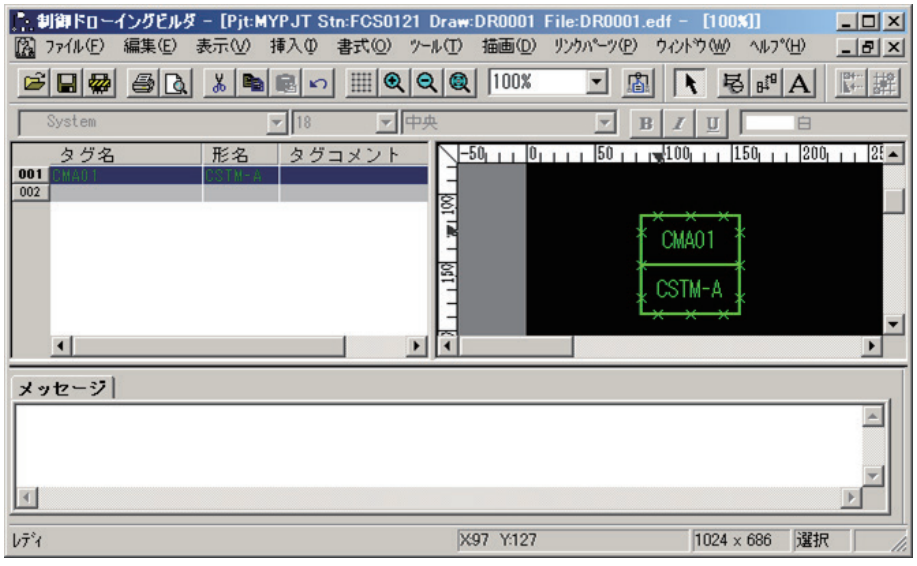
■ ユーザカスタムブロックの定義

制御ドローイングビルダでユーザカスタムブロックを定義します。

1. 制御ドローイングビルダを開いてください。
2. 「機能ブロック選択」ダイアログボックスで、［ユーザカスタム］の［CSTM-A］または［CSTM-C］を選択してください。

030208J.ai

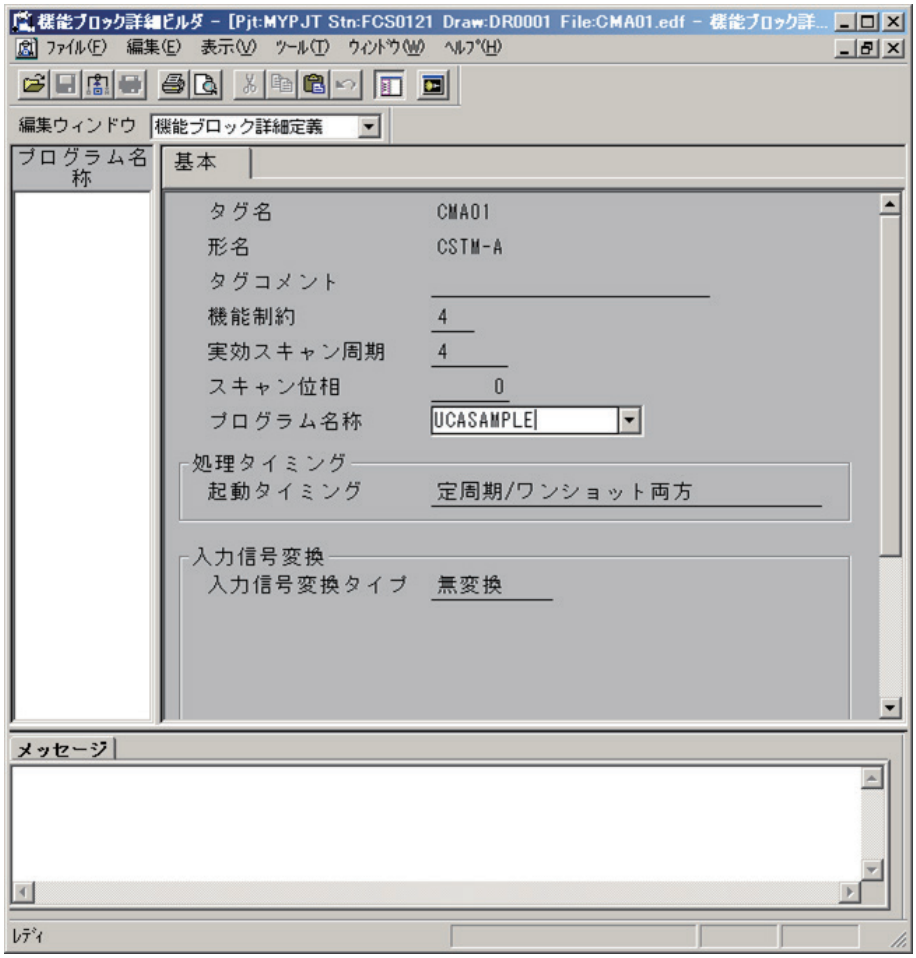
3. ユーザカスタムブロックが配置されるので、タグ名を入力してください。



030209J.ai

4. 作成したユーザカスタムブロックを選択し、機能ブロック詳細定義ビルダを起動してください。

5. プログラム名称にユーザカスタムアルゴリズム名称を入力してください。



030210J.ai

ここではユーザカスタムアルゴリズム名称を文字入力していますが、あらかじめユーザカスタムアルゴリズムの登録を行っていれば、選択エリアで登録済みのユーザカスタムアルゴリズム名称を選択できます。

参照 ユーザカスタムアルゴリズムの登録方法は、以下を参照してください。
「3.4 ユーザカスタムアルゴリズムの登録」

3.3 ユーザカスタムアルゴリズムのデバッグ

ユーザカスタムアルゴリズムのデバッグの手順について説明します。
デバッグを行うには、あらかじめユーザカスタムブロックにデバッグ対象のユーザカスタムアルゴリズム名称を定義し、関連付けを行っておく必要があります。

重要 デバッグを行っているときに、オンラインメンテナンスは行わないでください。
（デバッグ途中では APCS シミュレータが実行を停止しているので、オンラインメンテナンス処理が行えません）
オンラインメンテナンスのデバッグを行いたい場合には、以下の処理で代用してください。

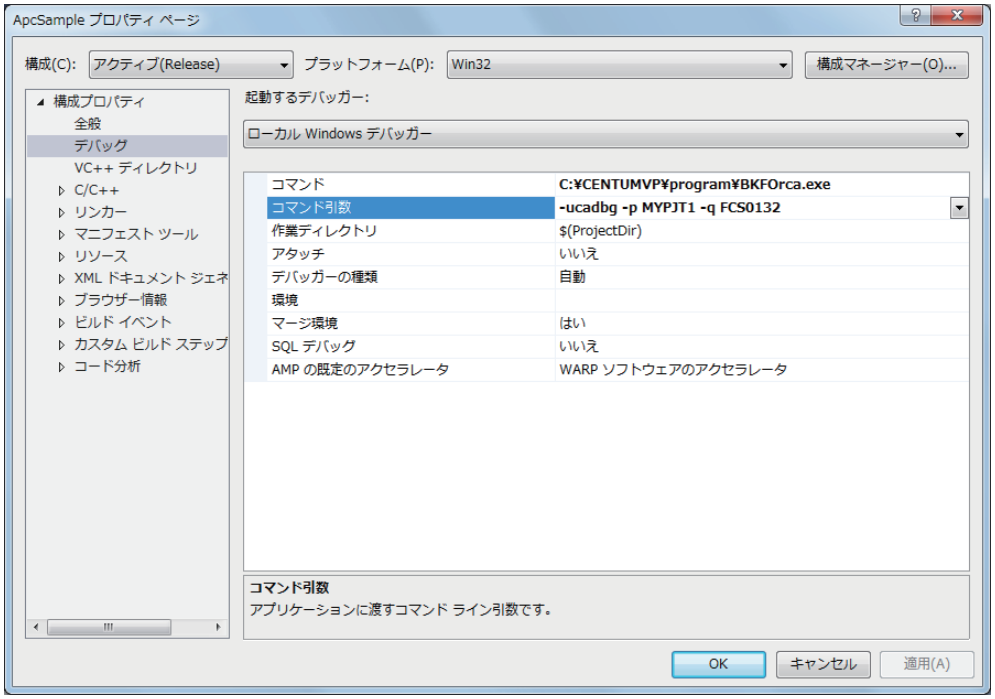
- MODE がサービスオフ（O/S）以外からサービスオフ（O/S）になる処理
- MODE がサービスオフ（O/S）からサービスオフ（O/S）以外になる処理

3.3.1 ユーザカスタムアルゴリズムのデバッグ

■ ソリューションへAPCSシミュレータコマンドの登録

Visual Studio から APCS シミュレータを起動させる設定を行います。

1. Visual Studio で［プロジェクト］メニューの［プロパティ］を選択してください。
「プロパティページ」ダイアログが表示されます。



030301J.ai

2. ［構成プロパティ］の［デバッグ］を選択し、［コマンド］に次の値を設定してください。
＜インストール先＞ ¥program¥bkforca.exe
例）
C:¥CENTUMVP¥program¥bkforca.exe

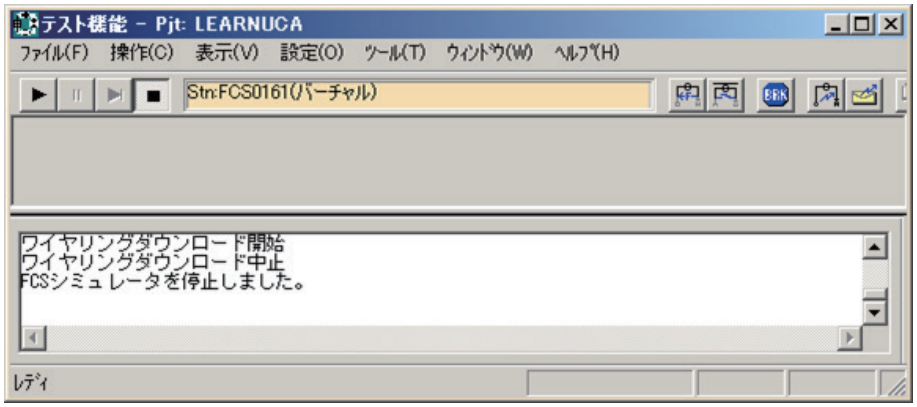
3. ［コマンド引数］に次の値を設定してください。
-ucadbg -p pjt -q stnName
ucadbg ：ユーザカスタムブロックデバッグ指定
pjt ：CENTUM プロジェクト名
stnName ：ステーション名（（例）FCS0101）

4. ［OK］ボタンをクリックしてください。

補足 本デバッグ時にロードされるユーザカスタムアルゴリズム（DLL）は、ユーザカスタムアルゴリズム登録エリアにある DLL ではなく、ビルドで作成された［ソリューション名￥Release］下にある DLL がロードされます。ユーザカスタムアルゴリズム登録エリアにデバッグする DLL と同じ名称の DLL が存在する可能性があります。そのときは、Release 下にある DLL のほうが有効になります。Release 下に存在しない DLL は、ユーザカスタムアルゴリズム登録エリアにある DLL がロードされます。

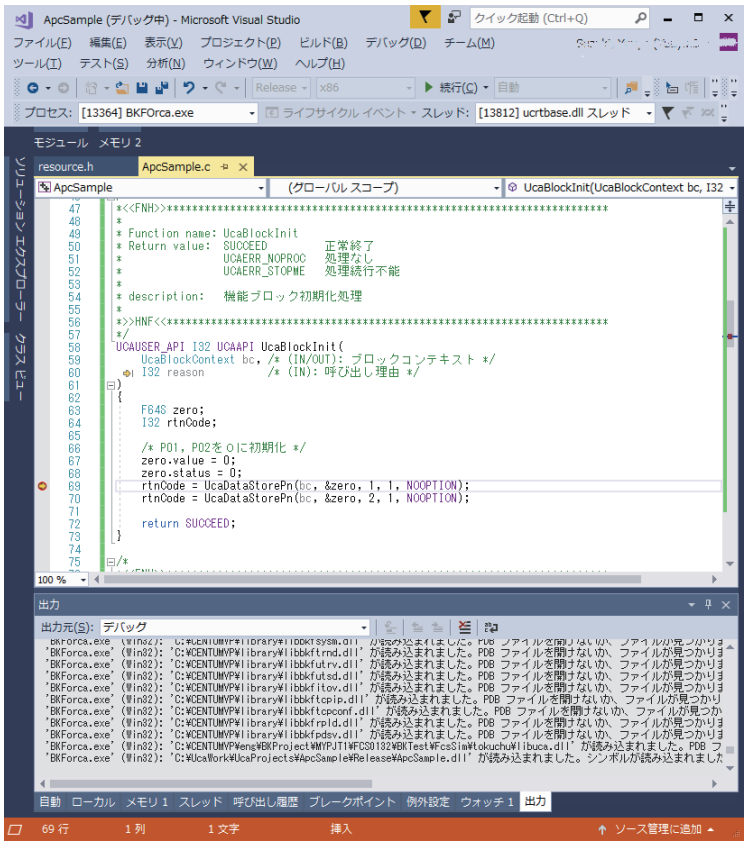
■ デバッグの開始

1. システムビューからテスト機能でバーチャル HIS と APCS シミュレータを起動してください。
   必要な場合は、FCS シミュレータも起動してください。
2. 起動した APCS シミュレータを、テスト機能画面で停止してください。



030302J.ai

3. Visual Studio でデバッグ対象のソリューションを開いてください。
4. ユーザカスタムブロックのソースプログラムを表示し、デバッグしたい個所にブレークポイントを設定してください。
5. ［デバッグ］メニューの［デバッグ開始］を選択してください。
   デバッグが開始されます。
   デバッグに関しては Visual Studio のデバッグ機能をそのまま使用するので Visual Studio のデバッグ機能を参照してください。



030303J.ai

6. デバッグを終了する場合には、［デバッグ］メニューから［デバッグの停止］を選択してください。

3.4 ユーザカスタムアルゴリズムの登録

ユーザカスタムアルゴリズムのAPCSへの登録について説明します。
デバッグが完了したユーザカスタムアルゴリズムを、システムビューでAPCSへ登録します。
登録後、テスト機能、および実機APCSを起動することで、総合的なユーザカスタムアルゴリズムの動作確認が可能になります。

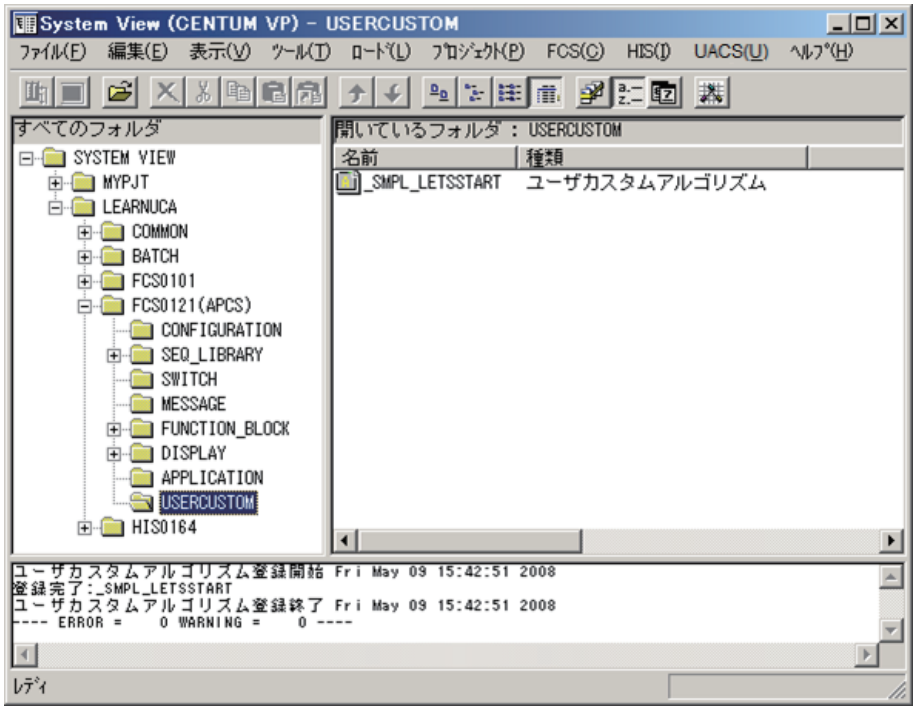
重要 ユーザカスタムアルゴリズムを登録するときは、Visual Studio より該当のソリューションを閉じてください。Visual Studio で開いているソリューション（ユーザカスタムアルゴリズム）を登録することはできません。登録時にエラーになります。

■ ユーザカスタムアルゴリズムのAPCSへの登録

作成したユーザカスタムアルゴリズムをシステムビューで APCS に登録する手順を以下に示します。

1. システムビューで、あらかじめ作成してある APCS を選択してください。
2. ［ファイル］メニューより、［新規作成］ー［ユーザカスタムアルゴリズム］を選択してください。
   「ユーザカスタムアルゴリズム登録」ダイアログが表示されます。
3. ダイアログボックス上部の「フォルダ」の右側にあるの「…」ボタンをクリックしてください。
   「フォルダの参照」ダイアログが表示されます。
4. 作成したユーザカスタムアルゴリズムのソリューションフォルダの 1 つ上の階層のフォルダを指定してください。
   （例）
   ソリューションフォルダ：F:¥UCAWork¥UCASample¥_SMPL_LETSSTART
   登録時に選択するフォルダ：F:¥UCAWork¥UCASample
   ビルドされているユーザカスタムアルゴリズムの一覧が表示されます。
5. 登録するユーザカスタムアルゴリズムを選択し、［OK］ボタンをクリックしてください。

システムビューに登録したユーザカスタムアルゴリズムが表示されます。



030401J.ai

3.5 テスト機能での動作確認

テスト機能を使用し、ユーザカスタムアルゴリズムの動作確認を行う手順について説明します。
システムビューでユーザカスタムアルゴリズムを登録後、テスト機能を起動して総合的な動作確認を行います。
ここで問題が発生した場合、Microsoft Visual C++のデバッグ機能（プロセスアタッチデバッグ機能）を利用してデバッグを行うことができます。
通常は、Microsoft Visual C++のデバッグ環境へ戻り、デバッグを行ってください。

■ 動作確認手順

1. System View よりテスト機能でバーチャル HIS と APCS シミュレータを起動してください。
   必要な場合は、FCS シミュレータも起動します。
2. テスト機能またはバーチャル HIS を使用して動作確認を行ってください。

■ 問題が発生した場合のデバッグ手順

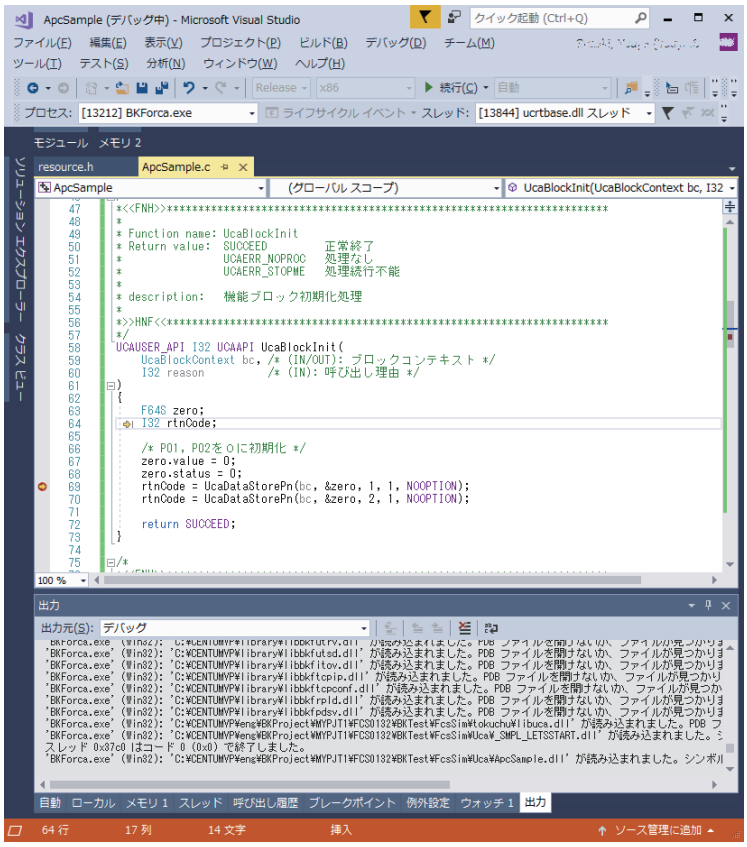
問題が発生し、デバッグが必要になった場合、Visual Studio の操作でデバッグを行います。

1. Visual Studio で、デバッグするユーザカスタムアルゴリズムのソリューションを開いてください。
2. ［デバッグ］メニューの［プロセスにアタッチ］を選択してください。
   「プロセスにアタッチ」ダイアログが表示されます。

補足 BKFOrca プロセスは、FCS シミュレータ／ APCS シミュレータ用のプロセスで、シミュレートを行うステーション数分存在します。
そのため、テスト機能で複数の FCS シミュレータ／ APCS シミュレータを起動した場合、複数の BKFOrca がプロセスの列に表示されます。ここでは、デバッグを行う APCS ステーションの BKFOrca を選択する必要があるため、タイトル列にデバッグしたいステーション名が表示されている BKFOrca プロセスを選択してください。

---

3. プロセスの列の中から、タイトルがデバッグしたい APCS 名となっている BKFOrca をクリックして［アタッチ］ボタンをクリックしてください。
4. ソリューションエクスプローラでデバッグしたいユーザカスタムアルゴリズムのソースファイルを開いてください。
5. デバッグしたい場所にブレークポイントを設定してください。



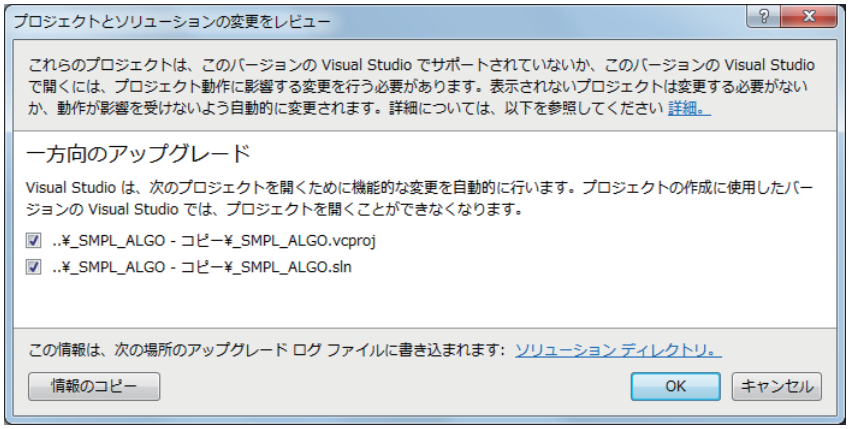
6. Visual Studio のデバッグ機能でデバッグを行ってください。
7. デバッグを終了する場合には、［デバッグ］メニューの［すべてデタッチ］を選択してください。

Blank Page

Appendix A. VS2008ワークスペースのコンバート

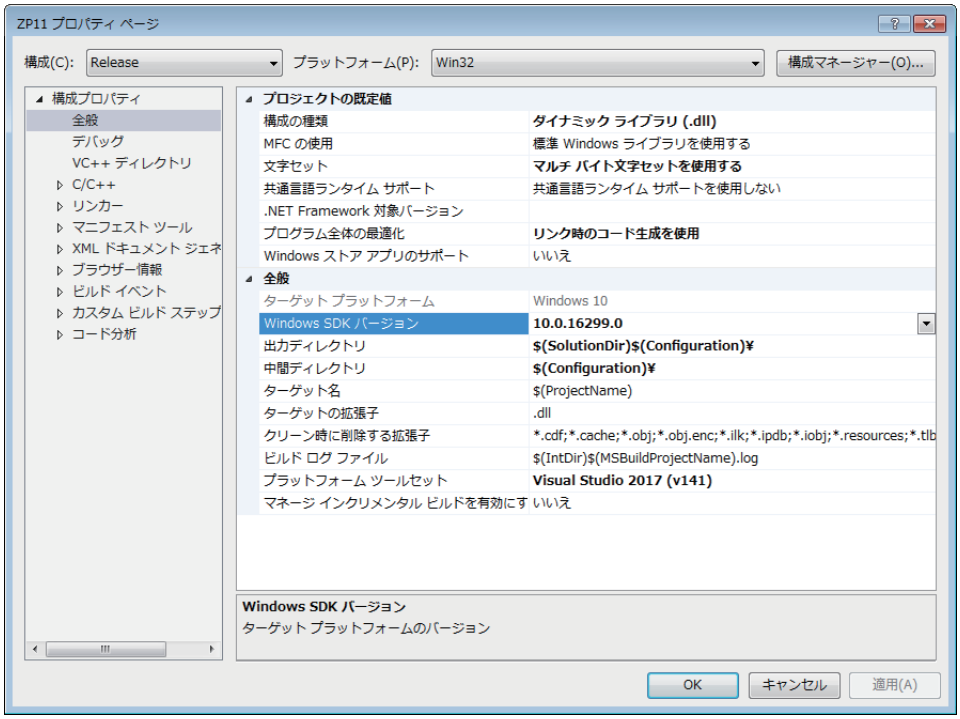
VS2008で作成されたワークスペースを、VS2017形式にコンバートする手順を示します。

1. VS2008 で作成されたソリューション（*.sln）を VS2017 で開いてください。
2. 次のダイアログが表示されたら、［OK］ボタンをクリックしてください。



APA0001J.ai

3. ［プロジェクト］メニューの［プロパティ］を選択してください。
「プロパティページ」ダイアログが表示されます。
4. ［構成プロパティ］－［全般］の設定を次のように行ってください。
   - ［Windows SDK バージョン］にインストール済みの Windows 10 SDK のバージョンを設定してください。



APA0002J.ai

補足 Windows 10 SDK のバージョンは、マイクロソフトから取得したタイミングで変わります。例では、「10.0.16299.0」であることを示しています。

5. 「3.1.4 プロジェクトの設定」の手順に従い、プロジェクトを設定してください。
6. ビルドを行い、モジュールを作成してください。

参照 プロジェクトの設定方法については、以下を参照してください。
「3.1.4 プロジェクトの設定」

APCS ユーザカスタムブロック 開発環境

IM 33J15U23-01JA 2 版

索　引

Blank Page

改訂情報

資料名称　　：APCS ユーザカスタムブロック開発環境
資料番号　　：IM 33J15U23-01JA

2019年8月／2版／R6.07以降
前書き　　「■ 商標」の記述変更

2018年8月／初版／R6.06
新規発行

■ お問い合わせについて
問い合わせ：http://www.yokogawa.co.jp/dcs より、お問い合わせフォームをご利用ください。
■ 著作者　　横河電機株式会社
■ 発行者　　横河電機株式会社
〒 180-8750　　東京都武蔵野市中町 2-9-32

Blank Page