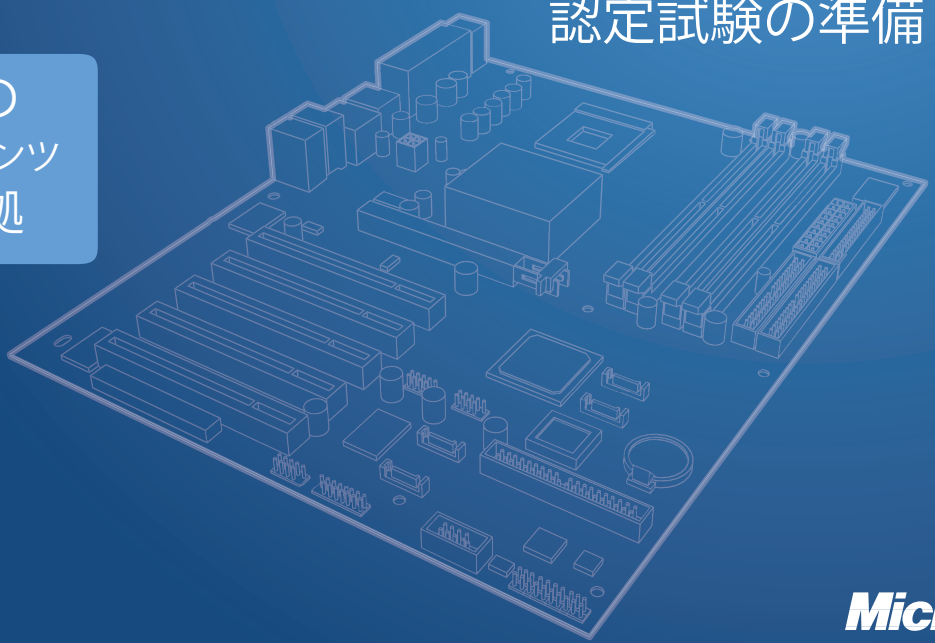
MCTS
Exam 70-571

Windows® Embedded CE 6.0

準備キット

認定試験の準備

最新の
R2 コンテンツ
に準拠



非売品

Microsoft®

出版元

Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

データ取得編集者 :	Sondra Webber、Microsoft Corporation
筆者 :	Nicolas Besson、Adeneo Corporation Ray Marcilla、Adeneo Corporation Rajesh Kakde、Adeneo Corporation
著作指導 :	Warren Lubow、Adeneo Corporation
技術レビューア :	Brigette Huang、Microsoft Corporation
編集出版 :	Biblioso Corporation

本体番号 3043-GA1

Body Part No. 098-109627

目次一覧

第 1 章

オペレーティング システムのカスタマイズ

Windows ィ Embedded CE 6.0 R2 をターゲットデバイスに展開するときは、必要なオペレーティング システム (OS) コンポーネント、機能、ドライバ、構成の設定を含むランタイム イメージを使用しなければなりません。ランタイム イメージは OS デザインのバイナリ表示です。OS デザインを作成またはカスタマイズするか、対応ランタイム イメージを生成するには、Windows Embedded CE 6.0 用 Microsoft ィ Platform Builder を使用できます。OS デザイン毎に、Microsoft ィ Visual Studio ィ 2005 で新規の開発プロジェクトを通常に作成して自分用のターゲット デバイスとアプリケーションに必要なコンポーネントだけを含めます。これにより、オペレーティング システムのスペースを縮小し、ハードウェア用件を低減します。しかし、コンパクトで機能的なランタイム イメージを作成するには、Platform Builder（ユーザ インターフェイス (UI)、カタログ コンポーネント、ビルド処理の特性）をより良く理解する必要があります。本章では、OS デザインの作成方法と、Windows Embedded CE ランタイム イメージの新規生成方法を説明します。

本章の試験対象 :

- OS デザインの作成およびカスタマイズ
- Windows Embedded CE サブプロジェクトの構成
- コンポーネントの複製
- カタログ項目の管理
- ソフトウェア開発キット (SDK) の生成

始める前に

この章のレッスンを完了するには、以下の予備知識が必要です :

- Windows Embedded CE でのソフトウェア開発の基本的な知識を有すること。
- Windows Embedded CE 6.0 R2 用 Platform Builder のディレクトリ構成とビルド処理の基本的な理解していること。
- Windows Embedded CE ランタイム イメージのバイナリ作成とターゲット デバイスへのダウンロードに精通していること。
- SDK を使用した、Windows Embedded CE 用アプリケーション開発の経験があること。
- Microsoft Visual Studio 2005 Service Pack 1 と Windows Embedded CE 6.0 用 Platform Builder がインストールされた開発用コンピュータについて。

レッスン 1: オペレーティング システム デザインの作成とカスタマイズ

目的に適っている Windows Embedded CE 6.0 R2 の提供する機能をできるだけ多く（または少なく）使用して、Visual Studio 2005 の Platform Builder で OS デザインを作成できます。例えば、ポータブル マルチメディア デバイスなどの特定なターゲット デバイス用 OS デザインを作成する、またリモート プログラム可能なワイアレス デジタル サーモスタット用の OS デザインを作成することができます。この二つのターゲット デバイスは、同じハードウェアに依存する可能性がありますが、各々の目的が異なり、OS デザイン要件も異なります。

このレッスンを終了すると、以下をマスターできます :

- OS デザインの役割と特性を理解できる。
- OS デザインを作成、カスタマイズ、使用できる。

レッスン時間 (推定): 30 分

オペレーティング システム デザイン 概要

OS デザインは、ランタイム イメージに含まれるコンポーネントと機能を定義し、本質的に、Visual Studio の Windows Embedded CE 6.0 R2 用 Platform Builder プロジェクトに対応します。OS デザインは下記の要素のいずれか、またはすべてを含みます :

- カタログ項目 (ソフトウェアコンポーネントとドライバを含む)
- サブプロジェクト形式の追加ソフトウェア コンポーネント
- カスタム レジストリ設定
- ビルドオプション (ローカリゼーションまたは Kernel Independednt Tranport Layer (KITL) を基とするデバッグ)

さらに、各 OS デザインは少なくとも一つのデバイス ドライバのボード サポート パッケージ (BSP) への参照を含み、これには ハードウェア特定 ユーティリティ、OEM アダプテーション層 (OAL) を含まれます。

OS デザインの作成

Windows Embedded CE には、OS デザインを容易に作成できる OS デザイン ウィザードがあります。ウィザードを起動するには、Windows Embedded CE 6.0 R2 用 Platform Builder を組込み済みの Visual Studio 2005 を起動し、[ファイル]

メニューの [新規作成] をポイントして、[プロジェクト] をクリックすると、[新しいプロジェクト] ダイアログ ボックスが表示されます。ダイアログの [プロジェクト種類] の下にある [CE 6.0用Platform Builder] を選択し、[Visual Studio にインストールされたテンプレート] の下にある [OS デザイン] を選択し、OS デザイン名を [名前] フィールドに入力してから [OK] をクリックすると、Windows Embedded CE 6.0 OS デザイン ウィザードが起動します。

OS デザイン ウィザードから、よく使われるオプションとすでに選択された カタログ コンポーネントが付いた形で BSP とデザイン テンプレートを使用できます。ウィザードで定義したすべての設定はあとで変更できますので、今は個々の設定にこだわる必要はありません。デザイン テンプレート ページで選択したテンプレートにもよりますが、OS デザイン ウィザードは選択テンプレートに関連した特定のオプション デザイン テンプレートバリエーション ページを追加の形で表示する場合があります。例えば、Windows シン クライアント、Enterprise Terminal、および Windows Network Projector は、すべてリモート デスクトップ プロトコル (RDP) を使用するデバイスであり、また同じ シン クライアント デザイン テンプレートのバリエーションでもあります。選択したテンプレートとバリエーションにもよりますが、OS デザイン ウィザード は OS デザイン内の特定のコンポーネントを含めるために追加ページを表示する場合があります (例：ActiveSync ィ、WMV/MPEG-4 Video Codec、あるいは IPv6)。

OS デザイン テンプレート

CE 6.0 OS デザイン テンプレートは、ある特定の用途で Windows Embedded CE を使用するために必要な カタログ コンポーネントのサブセットです。OS デザインの新規作成時にテンプレートは必ずしも必要ではありませんが、テンプレートを使用するとかなりの時間を節約することができます。[カタログ項目ビュー] . で、カタログ コンポーネントを選択すると、あとでカタログ コンポーネントを簡単に変更できます。

適切なテンプレートを選択することで、開発時間と労力を節約できます。例えば、あなたが 展示会で新しい評価ボードをデモする必要があるとします。その場合、PDA デバイス やコンシューマ メディア デバイス デザイン テンプレートから始めて、OS デザイン ウィザードの中で必要なコンポーネントと共通 Windows アプリケーション (例：.NET Compact Framework 2.0、Internet Explorer ィ、および WordPad) を追加することをお勧めします。一方、コントローラ エリア ネットワーク (CAN) コントローラのドライバを開発する際は、スモール フットプリント デバイス デバイスのデザイン テンプレート から始めて必要不可欠なコンポーネントだけを追加することで、ランタイム イメージのサイズと起動時間を最小化できます。

OS デザイン ウィザードは柔軟性があり、カスタム デザイン テンプレートをサポートしています。テンプレートファイルは、拡張マークアップ言語 (XML) ドキュメントで %_WINCEROOT%\Public\CEBase\Catalog フォルダにあります。すでに存在する Platform Builder カタログ XML (PBCXML) ファイルのコピーから始めて、必要により PBCXML の構造を変更してください。Visual Studio を起動し、Visual Studio の [カタログ項目ビュー] を更新すると、Platform Builder は自動的にカタログ フォルダ内のすべての .pbcxml ファイルを列挙します。

カタログ コンポーネント を使用した OS デザインのカスタマイズ

OS デザイン ウィザードの終了後に OS デザインを簡単に変更することができます。カタログは、OS デザインに追加するすべてのコンポーネント用のリポジトリです。統合化開発環境 (IDE) 内から直接アクセスできます。ソリューション エクスプローラ ウィンドウ ペイン内の [カタログ項目ビュー] をクリックしてください。CE の機能のほとんどは、ActiveSync から TCP/IP まで ユーザが選択可能なカタログ コンポーネントに分割されています。これらのコンポーネントはユーザ インターフェイス (UI) 内から直接選択できます。各カタログ項目は、ランタイム イメージのビルドおよび統合に必要なコンポーネントの参照です。

他のカタログ項目に依存するカタログを追加するとき、暗黙的に OS デザインにも依存して追加します。これらの項目は、依存関係により OS デザインの一部を表し、カタログ項目ビュー内で緑色の四角のチェックボックスとして表示されます。一方、カタログ項目ビューは手動で選択された項目と、緑のチェックマーク付きのデザイン テンプレートに基づいて含まれた項目を表示します。

カタログ項目ビューで、すべてのカタログ コンポーネントまたはフィルタを使用して選択したカタログ項目のみを表示できます。. ソリューション エクスプローラ のカタログ項目ビューの左上隅にある [フィルタ] ボタン上の下向き矢印を押してフィルタを適用するか、カタログの [すべてのカタログ項目] オプションを選択してカタログ項目すべてを表示してください。

カタログ項目名か SYSGEN 変数コンポーネントセットを知っていると、手動で探すよりも追加または削除するカタログ項目を検索した方が便利な場合があります。項目名か SYSGEN 変数での検索では、カタログ項目ビューの一番上にあるテキストボックスに文字を入力してから緑の矢印をクリックしてください。

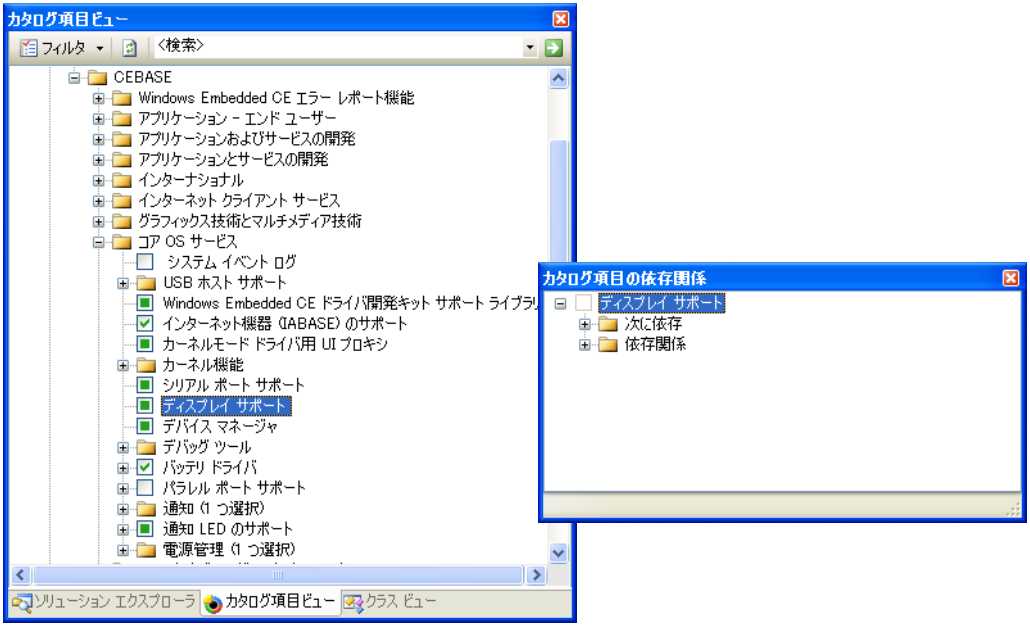


図 1-1　カタログ項目ビューの検索ボックスとカタログ項目の依存関係ウィンドウ

カタログ項目の依存関係を分析するには、項目を右クリックし [依存関係の表示] を選択すると カタログ項目依存ウィンドウが表示されます (図 1-1)。例えば、この機能を利用してある特定のカタログ項目が依存関係によって追加された理由を調べることができます。CE 6.0 R2 では、Platform Builder は動的にカタログをスキャンして選択項目に依存するすべてのコンポーネントだけでなく依存されるコンポーネントを列挙します。

ビルド構成管理

Windows Embedded CE は別々に変更可能な複数のビルド構成をサポートします。標準設定は、リリースとデバッグの 2 種類です。これらは OS デザイン作成時に自動的に利用可能になります。デバッグビルド構成では、コンパイラはデバッグ情報を生成し、デバッグを容易にする ソース コードへのリンクを プログラム データベース ファイル (.pdb) に保管し、段階的なコード実行はコードの最適化をしません。デバッグ構成でコンパイルされた Windows Embedded CE ランタイム イメージは、リリース設定でコンパイルしたイメージよりも 50 から 100 パーセント大きくなります。Visual Studio の [ビルド メニュー] を開き、[構成管理] をクリックして構成管理ダイアログ ボックス内の [アクティブ ソリューション構成] から、目的のビルド構成を選択します。標準のツールバーのプルダウン メニューを使用して、希望のビルド構成を選択することもできます。

OS デザイン プロパティ ページ

ビルド構成毎に、多くのプロジェクト プロパティ (例：ロケール、KITL を含める / 含めない、カスタム ビルド アクション、バイナリイメージのサブプロジェクト包含、カスタム SYSGEN 変数) を設定することができます。ソリューション エクスプローラ の [OS デザイン ノード] を右クリックして、[プロパティ ページ] ダイアログ ボックス を表示し [プロパティ] を選択すると、オプションにアクセスできます。OS デザイン ノードは、ソリューション最上位ノードの最初の子オブジェクトです。「OSDesign1」のように、プロジェクト名に対応します。ソリューション エクスプローラ が表示されないときは、[表示] メニューを開き [ソリューション エクスプローラ] をクリックします。カタログ項目ビューかクラス ビューが表示されているときは、[ソリューション エクスプローラ] タブをクリックしてソリューション ツリーを表示します。

ヒント　複数の構成のプロパティ設定

[プロパティ ページ] ダイアログ ボックスの左上隅にあるリスト ボックスからビルド構成を選択します。他のオプションは、いずれかまたはすべての構成を選択できます。度に複数のビルド構成のプロパティを設定するときに便利です。

ロケール オプション

[プロパティ ページ] ダイアログ ボックスの [構成プロパティ] に Windows Embedded CE イメージの言語設定を行う ロケール ノードがあります (図 1-2)。ロケール プロパティ ページは OS デザインのローカライズに必要な条件をカバーしますが、日本語のような東アジア言語は追加カタログ コンポーネントを必要とします。いくつかのインターナショナライゼーション (国際化) に関するカタログ コンポーネントを使用すると、ランタイム イメージのサイズがとても大きくなる場合があります。

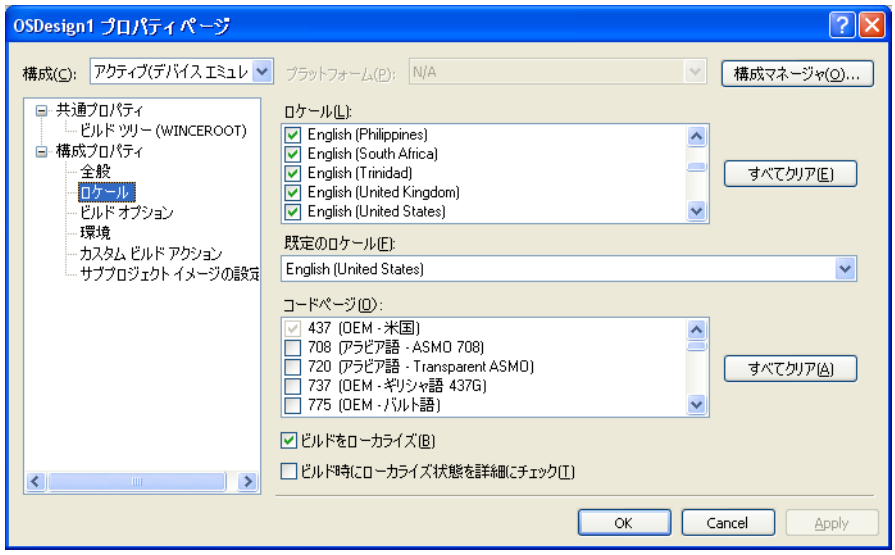


図 1-2　ロケール プロパティ ページ

ロケール プロパティ ページでランタイム イメージ用の以下のオプションを設定できます :

- ロケール　ランタイム イメージをローカライズする言語を選択します。選択言語にデフォルト ANSI と OEM コードページがある場合は、[コードページ] リストに表示されるとおり、対応するコードページが OS デザインに自動的に追加されます。
- デフォルト ロケール　OS デザイン用のデフォルトロケールを定義します。デフォルト言語は 英語 (米国)、コードページは 437 (OEM - 米国) です。
- コード ページ　OS デザインで利用可能な ANSI と OEM コードページを特定します。
- ビルドをローカライズ　ビルド プロセスにローカライズされた文字列とイメージ リソースを使用させます。Platform Builder は、OS デザイン ビルド プロセスのイメージ作成ステップ中に OS デザインのローカライズをします。ローカライズされたリソースファイルは、res2exe によって共通コンポーネント用バイナリファイルに統合されます。
- ビルド時にローカライズ状態を詳細にチェック　ローカライズ リソースが不足している場合、デフォルト ロケールのリソースを使用せずにビルド プロセスを中断します。

ビルド オプション

[プロパティ ページ] ダイアログ ボックスのロケール ノードのすぐ下に、イベント トラッキング、デバッグ、アクティブな OS デザインのビルドオプションを制御できる ビルド オプション ノードがあります (図 1-3)。

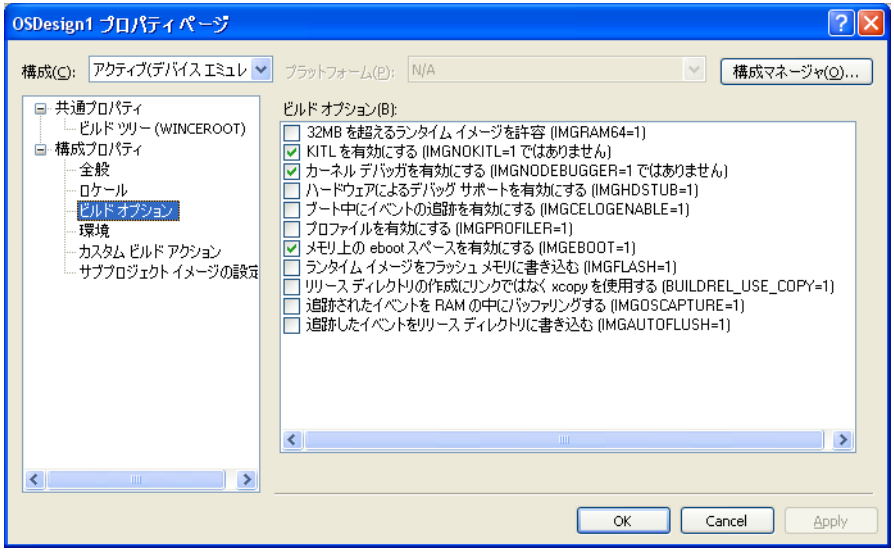


図 1-3 ビルド オプション プロパティ ページ

ビルド オプション プロパティ ページでランタイム イメージ用の以下のオプションを設定できます :

- 追跡されたイベントを RAM の中にバッファリングする　Platform Builder は CE イメージ内に OSCapture.exe を含めます。また OSCapture.exe によって追跡されたオペレーティング システム イベントを RAM にログすることで、ファイルにフラッシュしてあとで見ることができます。
- メモリ上の Eboot スペースを有効にする　起動時に Ethernet boot loader (EBOOT) はデータを Windows Embedded CE OS に渡せるようにします。
- ブート中にイベントの追跡を有効にする　起動時に通常よりも CE イベント ログ収集をもっと早く可能にします。If you activate this option, event tracking starts before most of the kernel and file system initialization is complete.
- ハードウェアによるデバッグ　サポートを有効にする　サードパーティのハードウェア デバッグ ツールに必要です (JTAG は exdi2 との準拠を調査します)。

- カーネル デバッガを有効にする　Windows Embedded CE デバッガを有効にしてランタイム イメージ内のステップ実行ができます。カーネル デバッグには、KITL と Platform Builder の実行時の通信が必要です。
- KITL を有効にする　KITL をランタイム イメージに追加します。KITL は便利なデバッグツールで、開発者がカーネル デバッガを使用したり、リモートデバイスのファイルシステム・レジストリ・他のコンポーネントと対話したりコードを実行できます。オペレーティング システムの最終ビルドに KITL を入れると、オーバーヘッドを引き起こし起動時にホスト コンピュータとの接続時間を浪費するので入れないでください。
- プロファイルを有効にする　ランタイム イメージ内にカーネル プロファイラを有効にし、タイミングとパフォーマンスデータの収集と表示を可能にします。ターゲット デバイス上の Windows Embedded CE のパフォーマンスを最適化するには カーネル プロファイラは便利なツールです。
- 追跡したイベントをリリース ディレクトリに書き込む　ランタイム イメージに CeLogFlush.exe を追加します。CeLogFlush.exe は OSCapture.exe によって収集したログ データを開発コンピュータ上のリリースディレクトリの Celog.clg ファイルに自動的に書き込みます。
- 32MB を超えるランタイム イメージを許容　32 MB を超えるイメージの作成を可能にします。しかし 64 MB を超えるイメージをビルドする場合はこのオプションを使用しないでください。その場合、環境変数を設定してください (例 : IMGRAM128)。
- リリース ディレクトリの作成にリンクではなく xcopy を使用する　リンクではなく xcopy を使用して実際にファイルをコピーします。Copylink を使用すると、開発コンピュータ上で NTFS が必要になる ハードリンクを作成する可能性があります。
- ランタイム イメージをフラッシュ メモリに書き込む　EBOOT はランタイム イメージをターゲット デバイスのフラッシュ メモリに書き込みます。

環境オプション

[プロパティ ページ] ダイアログ ボックスにある [環境 オプション] は、ビルド処理に使用される環境変数設定に使用します。Windows Embedded CE 6.0 R2 のほとんどの機能はカタログ コンポーネントで有効化できますが、いくつかのオプションは、Platform Builder がランタイム イメージで必要なコードをコンパイルするために SYSGEN 変数を設定する必要があります。BSP の開発ではビルド処理に影響する環境変数の設定は有効な場合があります。環境変数は、Windows Embedded CE ビルド処理中にコマンド ラインからアクセス可能です。ソース、

バイナリイメージ ビルダ (.bib)、レジストリファイル (.reg) の情報を特定する場合も環境変数を使用できます。

ヒント　デバッグ では動作するがリリースで動作しない
デバッグ構成でランタイム イメージをビルドできるのに、リリース構成でビルドできない場合は、[プロパティ ページ] ダイアログ ボックスを表示し、構成リストボックスの [すべての構成] から環境オプションを選択してデバッグとリリース両方に同じ環境変数を設定します。

高度な OS デザイン構成

このセクションでは OS デザインに関する高度なトピックをいくつか説明します。特に、同じ OS デザインを持つ複数のプラットフォームのサポート方法 と、OS デザインにある通常のファイル ロケーションとファイル タイプについて説明します。

OS デザインと複数プラットフォームの関連付け

OS デザイン ウィザードを使用して新しい OS デザインを作成するとき、ボード サポート パッケージ ウィザード ページでひとつ以上の BSP を選択できます。OS デザインを複数の BSP に関連付けると、, 複数のプラットフォーム用に同じコンテンツを使用して別のランタイム イメージを生成することができます。この場合、プロジェクトに複数の開発チームが関わっていて特にターゲット ハードウェアが利用できない時に便利です。例えば、エミュレータ ベースのプラットフォーム用ランタイム イメージを生成できるので、ハードウェアの最終版が出来上がる前にアプリケーション開発チームは作業を開始できます。OS 機能性に関しては、アプリケーション開発チームはターゲット プラットフォームの最終版が出来上がる前にアプリケーション プログラミング インターフェイス (API) を使用できます。2 つのランタイム イメージは同じコンポーネントと構成設定を共有するので、API は最後のターゲットに含まれます。

初期作成のあとに複数のプラットフォーム用のサポートをOS デザインに追加できます。ソリューション エクスプローラの [カタログ項目ビュー] 内の BSP 下にある対応するチェックボックスを選択します。BSP を選択するとリリースとデバッグ構成に追加のプラットフォームを自動的に追加します。Visual Studio の [ビルド] メニューにある [構成マネージャ] から、違うプラットフォームとビルド構成の切り替えを行えます。しかしプラットフォーム毎に時間のかかる SYSGEN フェーズを入れてすべてのビルド処理を実行する必要があります。

OS デザイン パスとファイル

OS デザインを使用して再頒布するには、構成ファイルと開発コンピュータ上のロケーションをを確実に知ることが必要です。OS デザインのデフォルト ロケーションは、「%_WINCEROOT%\OSDesigns」です。各プロジェクトはそれぞれ子ディレクトリに対応します。OS デザインは次のファイルとディレクトリ構成に対応します :

- <ソリューション名>　Visual Studio がプロジェクト用に作成した親ディレクトリ。
  - < ソリューション名 >.sln　OS デザインプロジェクトの設定を保存する Visual Studio ソリューションファイル (.sln)。通常、ファイル名は OS デザインと同じになります。
  - < ソリューション名 >.suo　ソリューション エクスプローラ ビューの状態などユーザ依存の情報を含む Visual Studio ユーザ オプション ファイル (.suo)。通常、ファイル名は OS デザインと同じになります。
  - <OS デザイン名 >　OS デザイン プロジェクトに含まれる残りのファイルの親ディレクトリ。
    - <OS デザイン名 >.pbxml　OS デザインのカタログ ファイル。選択したカタログ コンポーネントの参照と、OS デザインに関するすべての設定が含まれます。
    - サブプロジェクト　このディレクトリは OS デザインの一部として作成されたサブプロジェクトのサブフォルダをそれぞれ含みます。
    - SDK　このディレクトリは OS デザイン用に作成されたソフトウェア開発キット (SDKs) を含みます。
    - Reldir　リリースディレクトリ。Platform builder は、ターゲット デバイスにダウンロード可能なランタイム イメージの作成中に、ファイルをこのディレクトリにコピーします。
    - WinCE600　SYSGEN フェーズの完了後、現在の OS デザイン用のリソースファイルや構成ファイルがコピーされます。

ソース コントロール ソフトウェア の留意事項

基本的に OS デザインは Windows Embedded CE ランタイム イメージを生成するための Platform Builder 構成ファイルの集まりです。開発チームでソース コントロール ソフトウェアを使用する場合、これらの構成ファイルを ソース コントロール リポジトリに保管するだけで済みます。CE SYSGEN フォルダ (ランタイムイメージのビルド処理中に使用される) や Reldir ディレクトリからのファ

イルは、Platform Builder と BSP がインストールされたどのワークステーションでも再構成できるためリポジトリに含める必要はありません。また、拡張子が .user や .suo で終わるファイルは IDE 用のユーザ固有設定のため省略することができます。.ncb ファイルは IntelliSense ィ データを含むため省略できます。

レッスン概要

Windows Embedded CE 6.0 R2 用 Platform Builder にある OS デザイン ウィザードを使って OS デザインをすばやく簡単に作成することができます。一つまたは複数の BSP を選択して、ターゲット プラットフォーム用ハードウェア固有デバイス ドライバおよびユーティリティと、カタログ項目を追加するためのテンプレート バリエーションが付いたデザイン テンプレートを含めることができます。OS デザイン ウィザード終了後、OS デザインをさらにカスタマイズできます。不必要なカタログ項目を削除したり、コンポーネントを追加したり、デバッグおよびリリースビルドオプションなどのプロジェクト プロパティを設定できます。デバッグ ビルド構成で Platform Builder は、リリース ビルドに比較して 50 ～ 100 パーセント増のランタイム イメージ デバッグ情報を含みます。しかしデバッグ ビルドは開発中にコードのデバッグやステップ実行ができます。デバッグとリリース ビルド オプションを別々に構成できるため、OS デザインをデバッグ構成でコンパイルできるものの、リリース構成でコンパイルできないという状況に直面する場合があります。この場合、デバッグとリリースの構成に全く同じ環境変数をセットするとうまくいく場合があります。OS デザインを配布する場合、検索可能なデフォルトのディレクトリ %_WINCEROOT%\OSDesigns にソースファイルを配置する必要があります。開発チームで作業を調整するために、ソース 管理 ソフトウェアを使用することができます。

レッスン 2: Windows Embedded CE サブプロジェクトの構成

サブプロジェクトは、比較的独立したコンポーネントを全体ソリューションに含めるための親プロジェクトに加えられた Visual Studio プロジェクトです。この場合、親プロジェクトは OS デザインに対応します。サブプロジェクトは以下のフォームになります :

- アプリケーション (マネージまたはネイティブ)
- ダイナミック リンク ライブラリ (DLL)
- スタティック ライブラリ
- 構成設定のみを含む空のプロジェクト

サブプロジェクトは、特定のアプリケーションやデバイス ドライバまたは他のコードモジュールを OS デザインに含め、コードと OS デザインをひとつのソリューションとして保持する際に便利です。

このレッスンを終了すると、以下をマスターできます :

- サブプロジェクトを作成して設定ができる
- サブプロジェクトをビルドして使用できる

レッスン時間 (推定): 20 分

Windows Embedded サブプロジェクト概要

Windows Embedded CE 6.0 用 Platform Builder を使って OS デザインの一部のサブプロジェクトを作成できます。サブプロジェクトはモジュール式で再頒布可能のため、アプリケーション、ドライバ、その他のファイルを OS デザインに追加するときに BSP の一部として手動でビルドツリーに追加する必要がなく便利です。またテストアプリケーションおよび開発ツール用サブプロジェクトを作成してテストデバイス上でツールの作成と実行を簡単に行うことができます。

サブプロジェクトの種類

Windows Embedded CE は以下のサブプロジェクトをサポートします :

- アプリケーション　C　または　C++　言語でプログラムされたグラフィック ユーザ インターフェース (GUI) Win32 ィアプリケーション。

- コンソール アプリケーション　C または C++ 言語でプログラムされた GUI なしのアプリケーション。
- ダイナミック リンク ライブラリ (DLL)　ランタイムにロードして使用するドライバまたは他のコード ライブラリ。
- スタティック ライブラリ　ライブラリファイル (.lib) 型のコードモジュールで、他のサブプロジェクトにリンクしたり、OS デザインの SDK の一部としてエクスポートできます。
- TUX ダイナミック リンク ライブラリ　Microsoft Windows CE テストキット (CETK) 用 Windows Embedded CE カスタム テスト コンポーネント (第 4 章で説明)。

サブプロジェクトを作成し、OS デザインに追加する方法

簡単にサブプロジェクトを新規に作成し、既存のプロジェクトをサブプロジェクトとして OS デザインに追加することができます。ほとんどの場合、Windows Embedded CE サブプロジェクト ウィザード を使用して、これらの作業を行うことができます。ソリューション エクスプローラのサブプロジェクトのフォルダを右クリックし、[新しいサブプロジェクトの追加] または [既存のサブプロジェクトの追加] をクリックすると、ウィザードが起動します。ただし、各種のサブプロジェクトが果たす目的や CE サブプロジェクト ウィザードで作成したファイルおよび設定、ビルド プロセス、サブプロジェクトのカスタマイズ オプションなどの詳細を理解しておけば役に立ちます。

CE サブプロジェクト ウィザードは、必須の構成ファイルがすべて保存されている OS デザイン フォルダに、サブフォルダを 1 つ作成します。このフォルダには、次の必須ファイルを含みます :

- <名前>.pbpxml　サブプロジェクトに関するメタデータ情報を含む XML ベースのファイル。このファイルはサブプロジェクトをビルドするための .bib, .reg、SOURCES、および DIRS ファイルを参照します。
- < 名前 >.bib　ビルド処理中の makeimg で使用されるバイナリ イメージ ビルダ (BIB) ファイルで、バイナリ イメージに含めるファイルを指定します。
- < 名前 >.reg　最終ランタイム イメージに含める設定を使用したレジストリ ファイル。
- Sources　Windows Embedded CE のソース ファイル。Windows Embedded CE のビルド処理を制御するオプションを含むメイクファイルです。

- メイクファイル　Windows Embedded CE ビルド処理で SOURCES ファイルと共に使用されるファイル。

あとで使うためにサブプロジェクトのコピーを作成する場合は、OSDesigns フォルダ (%_WINCEROOT%\OSDesigns) を開いてから、OS デザイン用のソリューション ファイルを開きます。ソリューション フォルダには、通常 <OS デザイン名 >.sln ファイルおよび OS デザインに対応して名付けられたフォルダ 1 つがあります。. このフォルダには、OS デザインの定義ファイルである <OS デザイン名 >.pbxml といくつかのサブディレクトリがあります。これらのサブディレクトリの 1 つがサブプロジェクト フォルダになります (図 1-4)。このフォルダをバックアップしておくことをお勧めします。このフォルダは、ソリューション エクスプローラのサブプロジェクト コンテナを右クリックし [既存のサブプロジェクトの追加] をクリックすると、あとから任意の OS デザインに追加することができます。

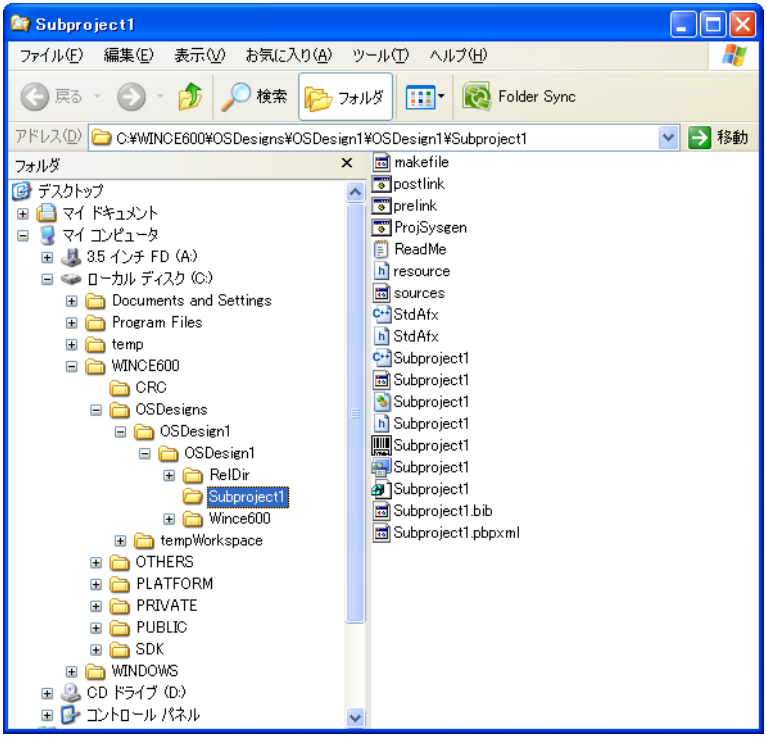


図 1-4　OS デザイン プロジェクトのサブプロジェクト フォルダ

Windows Embedded CE アプリケーションおよびダイナミック リンク ライブラリの作成方法

Windows Embedded CE アプリケーションまたは DLL を OS デザインに追加するには、CE サブプロジェクト ウィザードを使用して対応するサブプロジェクトを作成します。空のサブプロジェクトで最初から始めることもできますが、ベーシック コンソールまたは GUI アプリケーション テンプレートを使用し、必要に応じて後から独自のコードを追加する方が便利です。

スタティック ライブラリの作成

CE サブプロジェクト ウィザードにはスタティック ライブラリを作成するオプションもあり、あとから別のサブプロジェクトにリンクするか、SDK の一部として書き出すことができます。このオプションは複雑なサブプロジェクトに分割し、ハードウェアやファームウェア用のソリューションを開発するアプリケーション開発者により多くのオプションを提供するうえで有益な機能です。OS デザインのほかのサブプロジェクトがスタティック ライブラリに依存している場合、ライブラリを効果的に使用するためにサブプロジェクトのビルド順を調整する必要があるかもしれません。例えば、Windows Embedded CE アプリケーションがスタティック ライブラリを使用する場合、そのライブラリを最初にビルドしておけば、アプリケーションのビルド プロセスで最新のライブラリを使用できます。

ファイルや環境変数をランタイム イメージに追加するためのサブプロジェクトの作成

サブプロジェクトには、必ずしもソース コードが含まれているとは限りません。例えば、CE サブプロジェクト ウィザードを使用して、空のサブプロジェクトを作成して SOURCES ファイルを編集し、TARGETTYPE=NOTARGET を設定することで、バイナリ ターゲット ファイルを作成しないよう指定することができます。そのあとサブプロジェクトの .bib ファイルに対応する参照を追加し、ファイルをランタイム イメージに追加することができます。レジストリの設定をサブプロジェクトの .reg ファイルに追加し、サブプロジェクトの Projsysgen.bat ファイルを編集して SYSGEN 変数を追加することもできます。通常は .reg と .bib ファイル、および OS デザインのプロジェクト プロパティを直接変更したほうがより早く簡単にランタイム イメージに追加することができますが、サブプロジェクトを作成しておくと、今後複数の OS デザインのカスタマイズで再利用する予定がある場合、あとになって重宝することがあります。

サブプロジェクトの構成

Visual Studio はプロジェクトのプロパティを構成できるいくつかのオプションがあり、サブプロジェクトのビルド プロセスをカスタマイズすることができます。これらの設定を構成するには、この章の前半で説明した OS デザインのプロパティのページを表示します。[サブプロジェクト イメージの設定] でサブプロジェクトのプロパティを探します。現在の OS デザインで作成した、あるいは追加されたサブプロジェクトごとに、次のパラメータを構成することができます :

- ビルドから除く　このオプションを有効にすると、サブプロジェクトを OS デザインのビルド処理から排除します。つまり、ビルド エンジンは選択されたサブプロジェクトのソース ファイルを処理しません。
- イメージから除く　サブプロジェクトを変更した場合、ランタイム イメージの展開に時間がかかることがあります。サブプロジェクトを変更した場合、そのたびにターゲット プラットフォームから切断し、プロジェクトをリビルドして新しいイメージを作成し、ターゲット プラットフォームに再接続してから、最新のイメージをダウンロードしなければなりません。サブプロジェクトで作業をする際に時間と労力を節約するには、[イメージから除く] オプションでサブプロジェクトをランタイム イメージから排除しておきます。その場合、KITL、ActiveSync、またはその他の方法で、デバイスに転送し、デバイスのファイルを更新する方法を用意しておきます。
- 常にデバッグとしてビルドおよびリンクする　現在の OS デザイン ビルド処理がリリース設定を使用してる間に、デバッグ ビルド構成を使用してサブプロジェクトをビルドします。この方法を使用すると。オペレーティング システムがリリース設定を使用して実行している間、カーネル デバッガを使用して、サブプロジェクト コードをデバッグすることができます (このオプションをオンにしても、カーネル デバッガが自動的に有効になるわけではありません)。

ノート　ランタイム イメージから除く

サブプロジェクトからランタイム イメージを除く場合は、ターゲット デバイスにダウンロードした Nk.bin ファイルからサブプロジェクトのファイルを明示的に排除してください。その代わり Windows Embedded CE は、必要に応じて (KITL が有効であれば) KITL を使用して、リリース ディレクトリのサブプロジェクト ファイルに直接アクセスします。つまり、ランタイム イメージをいちいち再展開しなくても、ドライバまたはアプリケーションのサブプロジェクトのコードを変更することができます。リモート デバイスがコードが実行していないことを確認したら、コードをリビルドしてからもう一度実行します。

レッスン概要

Windows Embedded CE サブプロジェクトを使用して、アプリケーション、ドライバ、DLL、静的ライブラリを OS デザインに追加することができます。サブプロジェクトは、多数のアプリケーションとコンポーネントを含む複雑な Windows Embedded CE 展開プロジェクトを管理する際に有益な機能です。例えば、USB 周辺装置のカスタム シェル アプリケーションまたはデバイス ドライバをサブプロジェクトの形で OS デザインに含めておき、別の開発チームがこれらのコンポーネントを実装することができます。You can also use Windows Embedded CE サブプロジェクトを使用して、レジストリ設定、環境変数、または特定のファイルを、Core Connectivity (CoreCon) インターフェイスやテスト アプリケーションなどのさまざまな OS デザインに追加することもできます。サブプロジェクトを個別にバックアップしておき、既存のサブプロジェクトに追加して、のちの OS デザインに使用することもできます。

レッスン 3: コンポーネントの複製

Windows Embedded CE 6.0 R2 用 Platform Builder には、多彩な目的に応用できる再使用可能な Public ソース コード が用意されています。You can analyze and modify the source code for most of the components included in Windows Embedded CE に含まれている、シェルからシリアル ドライバのモデル デバイス ドライバ (MDD) 層 に及ぶほとんどのコンポーネントのソース コードは、分析および変更することができますが、Public ソース コードは直接変更しないでください。その代わり Public コードのコピーを作成し、それを機能用として自由に変更します。この方法なら、オリジナルの Windows Embedded CE 6.0 R2 コード ベースには影響を与えません。

このレッスンを終了すると、以下をマスターできます :

- 複製するコンポーネントを特定する。
- 既存のコンポーネントを複製する。

レッスン時間 (推定): 15 分

Public ツリーの編集とコンポーネントの複製

変更したいコードが %_WINCEROOT%\Public フォルダにあることがわかると、このコードを変更して別のフォルダに移動せずに、そのままビルドしたいと思うかもしれません。しかし Public ツリーを変更できない、いくつかの理由があります :

- Public ディレクトリをバックアップし、OS デザイン プロジェクトごとに、WINCE600\PUBLIC_Company1、WINCE600\PUBLIC_Company2、および WINCE600\PUBLIC_Backupなどのように個別のディレクトリを作成して、管理する必要がある。
- Windows Embedded CE 向けの更新プログラム、QFE (quick fix engineering) が提供する修正プログラム、および Service Pack が編集を上書きしてしまったり、編集した内容と互換性を持たない可能性がある。
- コードの再頒布が困難なうえ、エラーが発生しやすくなる。
- Public ディレクトリ ツリーのコードを変更した場合、オペレーティング システムのビルドに最長 3 時間かかります。Public フォルダ全体をリビルドせずに、特定のコードだけをリビルドできるくらい CE のビルドプロセスについて熟知しているのであれば、コンポーネントの複製についても同様です。

注意 Public コードの変更
Public フォルダ ツリーの内容は、絶対に変更しないでください。

コンポーネントの複製は面倒な作業に思えるかもしれませんが、長い目でみると開発のために費やす時間と労力を節約してくれます。

Public ソース コード

Platform Builder は Windows Embedded CE コンポーネントのいくつかに対して、インスタント コピーをサポートしています。これらのコンポーネントを複製するには、ソリューション エクスプローラの [カタログ項目ビュー] 内のカタログ項目を右クリックし、[カタログ項目の複製] を選択します。Platform Builder は OS デザインで選択したコンポーネントのサブプロジェクトをコードのコピーを使用して自動的に作成します。SYSGEN キャプチャ ツール などのほかの方法を使用する前に、複製したいカタログ コンポーネントがカタログ項目の複製オプションをサポートしているかどうか調べてください。サポートしている場合、クリックを 2 回するだけで複製を完了することができます (図 1-5)。

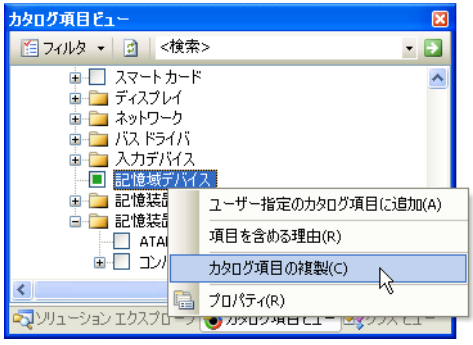


図 1-5 カタログ項目の複製

IDE で自動的にコンポーネントを複製できない場合は、手動で複製しなければなりません。しかし、Public ディレクトリ ツリーの .dll ファイルまたは .exe ファイルの Sources ファイルを調べると、このファイルはプラットフォームのディレクトリまたはサブプロジェクトのディレクトリにある Sources ファイルと違うことがわかります。これは、Public ディレクトリ ツリーのビルド処理が BSP のビルド処理と異なるためです。ビルド インストラクションはすべて、Sources ファイルと関連付けられたディレクトリと同じディレクトリにあるメイクファイルに定義されています。Public ディレクトリ ツリーは、必須のコンポーネントが相対的にリンクされる SYSGEN フェーズをサポートする必要があります。

Public ディレクトリ ツリーのコンポーネントを BSP コンポーネントまたはサブプロジェクトに変換するには、いくつかの手順を行う必要があります。詳しくは、Platform Builder for Microsoft Windows Embedded CE の付属ドキュメンテーションの「Using the Sysgen Capture Tool (Sysgen Capture Tool の使い方)」*http://msdn2.microsoft.com/en-us/library/aa924385.aspx* (英語) を参照してください。

基本的には、次の手順に従ってください :

1. Public コンポーネントのコードを、新しいディレクトリにコピーします。
2. 新しいディレクトリの Sources ファイルを編集します。RELEASETYPE=PLATFORM 行を追加するか、またはこの行が既に挿入されている場合は値を PLATFORM に変更し、ビルド エンジンが %_TARGETPLATROOT% フォルダに出力するように設定します。
3. WINCEOEM=1 を Sources ファイルに追加し、新しいディレクトリのコンポーネントをビルドします。ビルド エラーをすべて解消するために、さらに変更が必要な場合もあります。
4. Sysgen Capture tool を使用して、モジュラー Sources および Dirs ファイルを作成します。
5. Sysgen Capture Tool が作成したファイルの名前を変更し、このファイルとメイクファイルを一緒に使用して、新しく複製したモジュールをリビルドします。

必要なすべての変更を複製したコンポーネントに適用すると、他のコードと同じ要領で簡単い編集したり、再頒布することができます。

レッスン概要

Windows Embedded CE には、ほとんどの CE コンポーネントのソース コードを含む Public ディレクトリがありますが、この Public ディレクトリ ツリーのソース コードを直接変更しないようにします。その代わり、項目を自動または手動で複製します。複製による方法を使う理由を既に説明したように、Public ディレクトリ ツリーのソース コードをそのまま変更すると、問題を抱える原因になります。

レッスン 4: カタログ項目の管理

Windows Embedded CE の最も有益な機能の 1 つはカタログ システムです。カタログを使用すると、Windows Embedded CE ファームウェアを必要に応じてすばやく簡単にカスタマイズすることができます。各コンポーネントに対してカスタム カタログを作成すると、コンポーネントのインストールおよび構成が容易になります。これはアドホックとプロフェッショナルの Windows Embedded CE ソリューションを差別化する要因になります。アドホック ソリューションの場合、基本的なインストールに関するメモと必須の SYSGEN 変数を提供すれば十分かもしれませんが、プロフェッショナルなソフトウェアの場合には、カタログ項目と SYSGEN 変数と構成ための適切な値を提供する必要があります。

このレッスンを終了すると、以下をマスターできます :

- カタログのコンテンツをカスタマイズする。
- 新しいコンポーネント エントリを BSP カタログに追加する。

レッスン時間 (推定): 20 分

カタログ ファイル の概要

Windows Embedded CE カタログは、拡張マークアップ言語 (XML) を使用しており、ファイルには .pbcxml の拡張子が付けられます。カタログには多数の .pbcxml ファイルがあり、WINCEROOT ディレクトリにあります。Platform Builder はこれらのファイルを自動的に列挙し、ソリューション エクスプローラのカタログ項目ビューを生成します。

Platform Builder は次のディレクトリを解析し、カタログ項目を列挙します :

- Public カタログ ファイル %_WINCEROOT%\Public\<*any subdirectory*>\Catalog\
- BSP カタログ ファイル %_WINCEROOT%\Platform\<*any subdirectory*>\Catalog\
- サードパーティ カタログ ファイル %_WINCEROOT%\3rdParty\<*any subdirectory*>\Catalog\
- 汎用のシステムオンチップ (SOC) ファイル %_WINCEROOT%\Platform\Common\Src\soc\<*any subdirectory*>\Catalog\

ノート　サードパーティフォルダ

サードパーティフォルダ には、通常 OS デザインの一部として含めて配布することができる、スタンドアローン アプリケーションまたはソース アプリケーションがあります。サードパーティフォルダの .pbcxml ファイルを列挙することで、Platform Builder はこれらのコンポーネントのエントリをカタログ項目ビューに追加する方法を提供しています。

カタログのエントリの作成および編集

新しいカタログ項目を Windows Embedded CE カタログに追加するには、既存のカタログ ファイル (.pbcxml file) のコピーを作成してから、このファイルを Platform Builder と一緒に提供されているカタログ エディタを使用して編集します。Visual Studio の [ファイル] メニューにある [新規作成] をポイントして [ファイル] を選択して、新しいカタログ ファイルを作成することもできます。[新しいファイル] ダイアログ ボックスの Platform Builder for CE 6.0 R2 で、[Platform Builder カタログ ファイル] を選択してから [開く] をクリックします。

ノート　カタログ ファイルの編集

カタログ ファイルの編集は Platform Builder のカタログ エディタを使用します。メモ帳などのテキスト エディタの使用するための設定はできません。カタログ ファイルを Platform Builder 以外で開いたり編集すると、必要以上に時間がかかります。

カタログのエントリのプロパティ

それぞれのカタログのエントリには、Platform Builder で編集できるいくつかのプロパティがあります (図 1-6)。最も重要なプロパティには、次があります :

- 一意な ID　一意な ID 文字列。
- 名前　カタログ項目ビューに表示されるカタログ コンポーネントの名前。
- 説明　コンポーネントの説明で、ユーザがマウス ポインタをカタログ項目の上に数秒置くと表示されます。
- モジュール　このカタログ コンポーネントに付属するファイル一覧。
- SYSGEN　変数　カタログ項目の環境変数。カタログ　コンポーネントが SYSGEN 変数を設定する場合、ここで設定します。
- 追加変数　カタログ項目の追加の環境変数。このフィールドは　Sources、.bib,、および .reg ファイルで使用される環境変数を設定し、ビルド処理を制御することができるため、BSP のカタログ コンポーネントうちで最も重要な部分だと言えます。またこのフィールドを使用して、他のコンポーネントの依存関係を生成することもできます。

- プラットフォーム ディレクトリ　カタログ項目ファイルのある場所。新しい BSP の場合、このプロパティを BSP のディレクトリの名前に設定します。

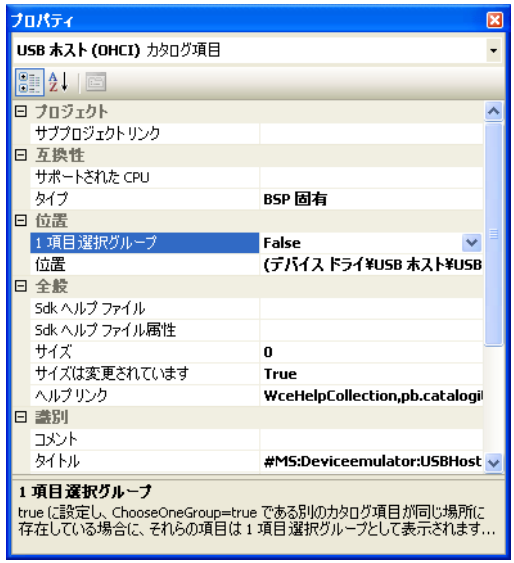


図 1-6　カタログ項目のプロパティ

ノート　一意の名前

各カタログ コンポーネントには、通常、ベンダとコンポーネントの名前で構成された一意の ID があります。カタログ項目の複製 機能を使用して、BSP を複製した場合、Platform Builder は複製したコンポーネントの一意の名前を自動的に作成します。カタログ ファイルを手動で編集する場合は、必ず一意の ID を使用してください。

新しいカタログ項目を OS デザインに追加する

新しいカタログ ファイルまたはカタログ項目を使用するには、対応する .pbcxml ファイルが サードパーティ ディレクトリまたは Platform ディレクトリのサブディレクトリにある Catalog という名前のサブフォルダにあることを確認してから、Visual Studio のカタログ項目ビューの [カタログツリーの更新] ボタンをクリックします。Platform Builder は サードパーティ ディレクトリおよび Platform ディレクトリをスキャンして、既存のカタログファイルを処理し、カタログを動的に再生成します。レッスン 1 で説明したように、カタログ項目ビューに一覧された新しいコンポーネントのチェック ボックスをオンにし、OS デザインに含めることができます。

カタログ項目を利用した BSP 開発

新しいカタログ項目を追加し、項目特有の環境変数の設定方法を学びました。このテクニックを使用して、コンポーネントを BSP に含め、C/C++ ビルド命令を設定し、ランタイム イメージのシステム レジストリの設定を変更することができます。ほかの開発者がこの BSP を使用する場合、OS デザイン プロジェクトのカタログ項目を選択すると、師弟した設定を無条件で使用します。カタログ コンポーネントを BSP に含めるには、BSP の Platform.bib ファイル を編集し、設定に基づいた条件付きステートメントを追加します。if-else ステートメントを使用して定義されているかどうかに関わらず、コンポーネントを含めることができます。.bib ファイルと .reg ファイルへの変更を反映させるために、Visual Studio の [ビルド] メニューの [詳細なビルド コマンド] の [現在の BSP およびサブプロジェクトのリビルド] コマンドを実行する必要があるかもしれません。[現在の BSP およびサブプロジェクトのリビルド] コマンドについては、第 2 章で詳しく説明します。

カタログ項目のプロパティで指定した環境変数に基づいて C/C++ 命令を設定するには、Sources ファイルで変数に基づいて条件付きステートメントを使用し、CDEFINES エントリを追加します。カタログ項目のプロパティに基づいて C/C++ ビルド命令を設定すると、将来における BSP のバイナリ バージョンの配布が困難になるため、通常は避けるようにしてください。

条件付きステートメントを使用して、システム レジストリのエントリを変更することもできます。新しいコンポーネントに関連した特定のレジストリ ファイルを含めたり、除いたりするには、.reg ファイルのみを編集します。

カタログからカタログ項目をエクスポートする

カタログ項目には直接複製できないものもあります。このようなコンポーネントを複製するには、新しいカタログ ファイルを作成しなければなりません。サードパーティ フォルダに新しいエントリを作成する場合、カタログ ファイルを新規に作成するか、または BSP の既存カタログ ファイルに新しいエントリを作成します。いずれの場合でも、すべての SYSGEN のオリジナルの値と追加変数が保持されているかどうかを確認するようにします。このレッスンで前述したように、カタログの各項目には一意の ID が必要ですので、ID を変更することを忘れないでください。

カタログ コンポーネントの依存関係

Windows Embedded CE 6.0 R2 用 Platform Builder のカタログは、コンポーネントの依存関係をサポートしています。コンポーネントが別のコンポーネントに依存していることを指定するには、カタログ項目のコンポーネントの SYSGEN または [追加変数] フィールド設定してから、その値を依存コンポーネントに追加環境変数の形で含めます。例えば、ディスプレイ用のディスプレイ ドライバとバックライト ドライバの両方のカタログ コンポーネントが BSP にある場合、ディスプレイ ドライバの [追加変数] フィールドを BSP_DISPLAY に設定し、バックライト ドライバの [追加変数] フィールドを BSP_BACKLIGHT に設定します。ディスプレイ ドライバをバックライト ドライバに依存させる場合、カタログ エディタで BSP_DISPLAY のカタログ エントリを編集し、BSP_BACKLIGHT を追加環境変数に追加します。ディスプレイ ドライバを OS デザインに含めると、Platform Builder は自動的にバックライト ドライバも含めます。カタログ項目ビューには、バックライト ドライバのチェック ボックスが表示され、緑の四角形はそのコンポーネントがディスプレイ ドライバに依存していることを示しています。

レッスン概要

Windows Embedded CE 6.0 R2 用 Platform Builder には、ファイル ベースのカタログ システムがあります。%_WINCEROOT% ディレクトリ ツリーの Platform ディレクトリまたは サードパーティ ディレクトリの個別のカタログ ファイルに、独自のカタログ項目を含めることができます。カタログ ファイルの形式は XML で、.pbcxml の拡張子が付いています。Visual Studio を起動し、ソリューション エクスプローラの [カタログ項目ビュー] を更新すると、Platform Builder は自動的に .pbcxml ファイルを列挙します。新しいカタログ項目を Windows Embedded CE カタログに追加するには、カタログ ファイルを新規に作成するか、または既存のカタログ項目のコピーを作成し、そのファイルの内容をカタログ エディタで編集します。すべての設定は Platform Builder で直接設定できるため、.pbcxml ファイルをメモ帳などのテキスト エディタで編集する必要はありません。条件付きの C/C++ ビルド命令、レジストリの変更、および依存関係の定義に対して、SYSGEN および追加環境変数を指定することができます。

レッスン 5: ソフトウェア開発キットの生成

ターゲット デバイス用のアプリケーションを作成するには、ソフトウェア開発キット (SDK) が必要です。SDK は自動的に OS デザインに対応するため、開発者は実際に使うことのできる機能だけを扱うことができます。SDK には OS デザインにある機能も含まれているため、アプリケーション開発者はサポートされていない API が原因で、ランタイム エラーを発生させるコードを誤って作成せずにすみます。

このレッスンを終了すると、以下をマスターできます :

- SDK の目的を特定する。
- SDK を生成する。
- ハード ドライブの SDK ファイルのローカライズする。
- SDK を使用する。

レッスン時間 (推定): 20 分

ソフトウェア開発キットの概要

OS デザインに対して有効なアプリケーションをコンパイルおよび作成するには、必須のヘッダー ファイルとリンクを含めて、開発プロジェクトの適切なライブラリにリンクする必要があります。OS デザインの SDK が、アプリケーション開発者に提供するカスタム コンポーネント用のヘッダー ファイルとライブラリを含む、すべての必須のヘッダー ファイルとライブラリを含んでいることを確認してください。Windows Embedded CE 6.0 R2 用 Platform Builder は、必須のヘッダー ファイルとライブラリをエクスポートすることで、OS デザイン用の SDK を作成することができます。

SDK の生成

カスタマイズした SDK の生成および配布は、通常 OS デザインの制作者の作業です。Platform Builder はこの目的のために SDK をエクスポートする機能を備えています。SDK エクスポート機能は、OS デザイン用にカスタマイズした SDK と SDK セットアップ ウィザードを含む .msi ファイルを作成します。

SDK の生成と構成

Platform Builder の SDK のエクスポート機能を使用して SDK を作成および構成するには、次の手順にしたがいます :

1. OS デザインを構成し、リリース構成で最低 1 回ビルドを行います。

2. ソリューション エクスプローラを表示し、SDK を右クリックして [新規追加] を選択して [SDK プロパティ ページ] ダイアログ ボックスを表示します。
3. [SDK プロパティ ページ] ダイアログ ボックスで、SDK の [全般] プロパティを構成し、[MSI フォルダ パス]、[MSI ファイル名]、および [ロケール] を定義します (図 1-7)。カスタム設定の名前もしていすることができます。
4. 追加のファイルを含めるには、[SDK プロパティのページ] ダイアログ ボックスの [追加フォルダ] ノードを選択します。
5. [OK] をクリックします。

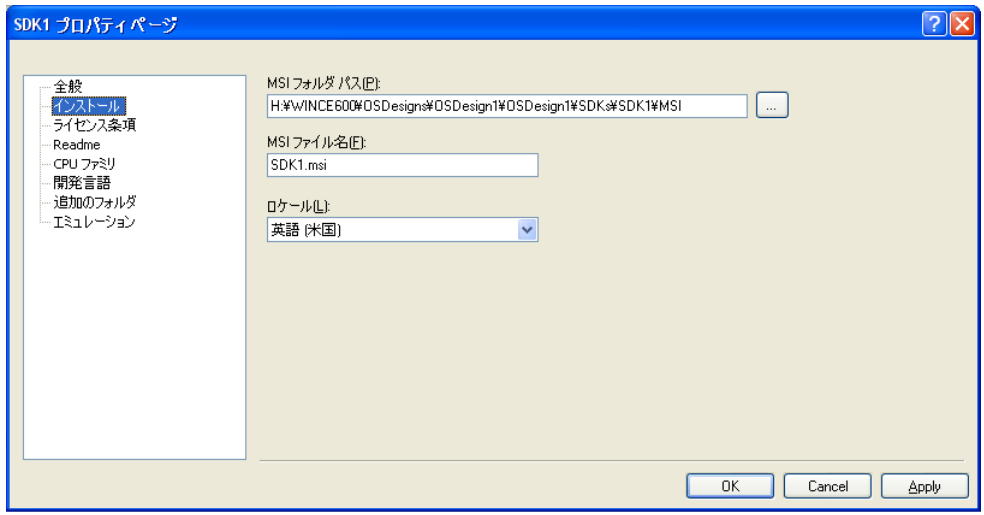


図 1-7 [SDK プロパティ ページ] ダイアログ ボックス

SDK への新しいファイルの追加

[SDK プロパティ ページ] の [追加フォルダ] オプションを使用するか、または OS デザインの SDK ディレクトリ (通常は \OSDesigns\< ソリューション名前 >\<OS デザイン名 >\WinCE600\< プラットフォーム名 >\SDK) にコピーし、ファイルを手動で SDK に追加することができます。ビルド エンジンがビルドのたびに最新バージョンのファイルを SDK にコピーするよう、.bat ファイルと Sources ファイルを使用して処理を自動化させることもできます。

ファイルは、必ず次の SDK サブディレクトリにコピーします :

- Inc　SDK のヘッダー ファイルを含みます。.
- Lib\<*Processor Type*>\<*Build Type*>　SDK のライブラリを含みます。

SDK のインストール

SDK ビルド プロセスを完了後、OS デザイン フォルダの SDK サブディレクトリの .msi ファイル (通常は %_WINCEROOT%\OSDesigns\< ソリューション名 >\<OS デザイン名 >\SDKs\SDK1\MSI\<SDK 名 >.msi) を探すことができます。この MSI は Platform Builder および該当する場合はサードパーティ コンポーネントのライセンス契約に従い、自由に再頒布することができます。

この MSI パッケージは Visual Studio 2005 がインストールされたどのコンピュータ にでもインストールすることができ、ターゲット デバイスの Windows Embedded CE アプリケーションの開発に使用できます。SDK がインストールされたコンピュータで %PROGRAMFILES%\Windows Embedded CE Tools\WCE600 のファイルを探します。

レッスン概要

Windows Embedded CE 6.0 R2 はコンポーネント化されたオペレーティング システムです。そのため、ターゲット デバイスで動作するアプリケーションを開発するアプリケーション開発者は OS デザインに応じてカスタマイズされた SKD が必要になります。ファイルやライブラリが足りないためにビルトや実行時に発生する問題を回避するため、カスタム SDK には Windows Embedded CE コンポーネントのみでなく、OS デザインに含まれるカスタム コンポーネントのヘッダー とライブラリも含める必要があります。Platform Builder の SDK をエクスポートする機能を利用して、SDK の生成と MSI パッケージを作成し、SDK セットアップ ウィザードで、アプリケーション開発コンピュータで SDK を展開することができます。

ラボ 1: OS デザインの作成、構成、およびビルド

このラボでは OS デザインを作成したあと、カタログからコンポーネントを追加して、デザインをカスタマイズします。このラボの内容は、『Microsoft Windows Embedded CE 6.0 R2 受験対策キット』の他の章のレッスンで必要なため、ここに示されている手順は必ずすべて行ってください。

ノート　詳しい手順
このラボで紹介する手順を正しく習得するために、この本の付属教材の「ラボ 1 の詳しい手順」を参照してください。

OS デザインの作成

1. Windows Embedded CE 6.0 R2 用 Platform Builder をプラグインした Visual Studio 2005 で [ファイル] メニューの [新規作成] をポイントして、[プロジェクト] をクリックすると、新しい OS デザイン プロジェクトが作成されます。
2. 既定の OS デザイン名 (OSDesign1) を使用します。
3. Visual Studio の Windows Embedded CE 6.0 OS デザイン ウィザードが開始します。
4. BSP リストの [デバイス エミュレータ :ARMV4I] チェック ボックスをオンにして、[次へ] をクリックします。
5. [使用可能なデザイン テンプレート] から [PDA デバイス] を選択します。[使用可能なデザイン テンプレート] から [モバイル ハンドヘルド] を選択します。.
6. ウィザードの次のページで .NET Compact Framework 2.0 と ActiveSync のチェックをオフにします (図 1-8)。[インターネット ブラウザ] と [Quarter VGA リソース - 縦モード] のチェックはオンのままにしておきます。
7. Networking Communications (ネットワーク接続) ウィザード ページで [TCP/IPv6 サポート] と [パーソナル エリア ネットワーク (PAN)] のチェックをオフにし、Bluetooth と Infrared Data Association (IrDA) のサポートを排除します。[ローカル エリア ネットワーク (LAN)] はオンにしたままにしておきます。
8. [完了] をクリックして、Windows Embedded CE 6.0 OS デザイン ウィザードを閉じると、Visual Studio OS デザイン プロジェクトが開きます。[ソ

リューション エクスプローラ] タブが有効になり、[ソリューション コンテナ] に新しい OS デザイン プロジェクトが表示されます。

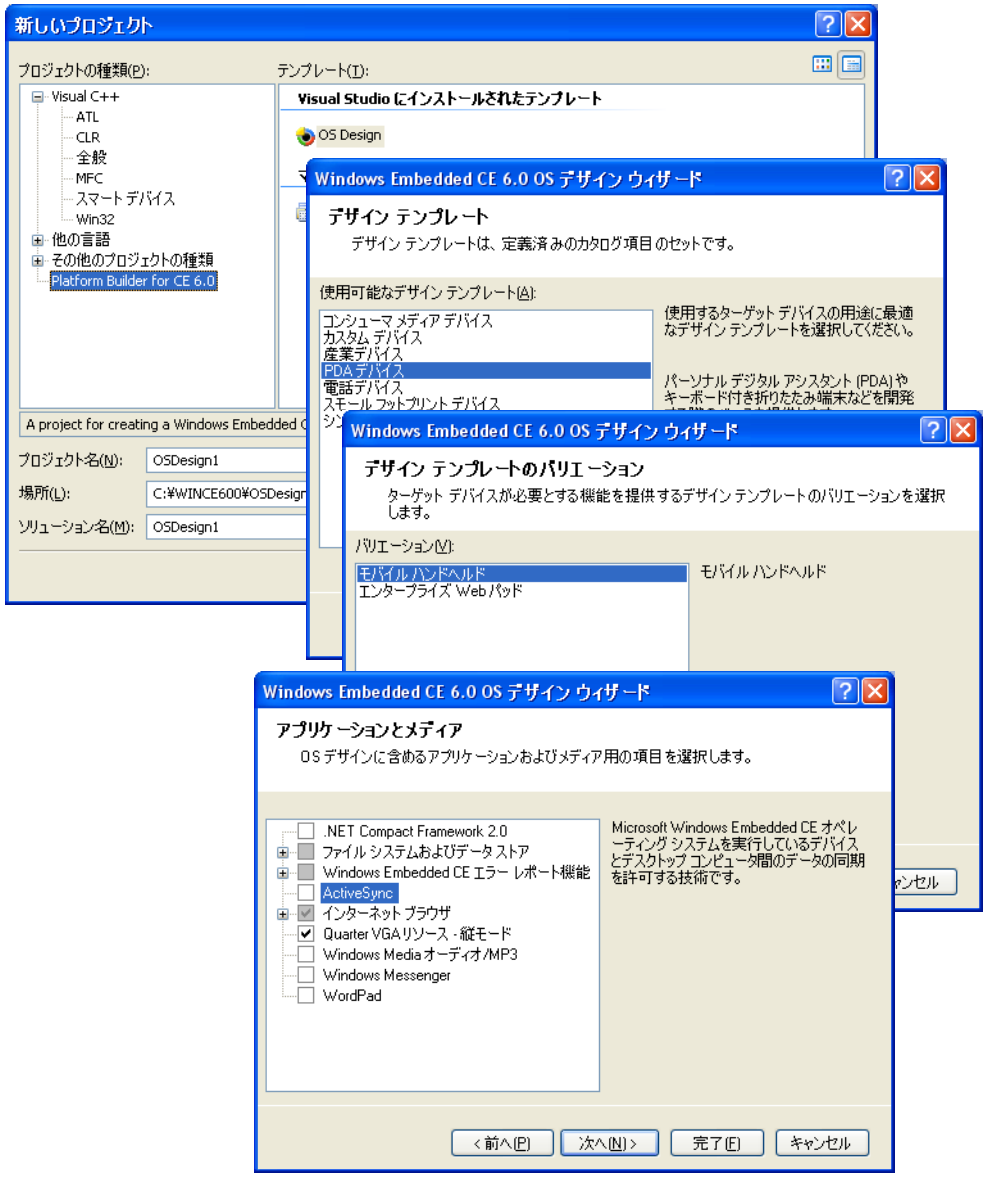


図 1-8　PDA デバイスの OS デザインを作成する

ノート　このあとの OS デザイン変更
OS デザイン ウィザードで作成される OS デザインの初期構成は、ウィザードの終了後に変更することができます。

- **OS カタログを調べる**

1. Visual Studio のソリューション エクスプローラで、[カタログ項目ビュー] をクリックします。
2. 個々のコンテナ　ノードを展開し、カタログのオンになっているチェック ボックスとアイコンを調べます。緑のチェック マークが表示された項目は、OS デザインの一部として特に追加されたことを示しています。緑の四角が表示された項目は、依存関係によって OS デザインに含まれていることを示しています。何も表示されていない項目はこの OS デザインには含まれていませんが、いつでも追加できることを示しています。
3. 緑の四角が表示されたカタログ項目を探します。
4. このカタログ項目を右クリックして、[項目を含める理由] を選択します。選択されたカタログ項目を OS デザインに含める理由になっているカタログ項目を示した [依存関係にあるカタログ項目を削除] ダイアログ ボックスが表示されます (図 1-9)。
5. カタログの Core OS | CEBASE | Applications ミ End User | Active Sync node を展開します。
6. ActiveSync システム CPL の項目のいずれかを右クリックして、[ソリューション ビューに表示] を選択します。ActiveSync コンポーネントを含むサブプロジェクトを示したソリューション エクスプローラのタブが表示されます。これは Windows Embedded CE 6.0 に用意されているソース コードをナビゲートする最適な方法です。

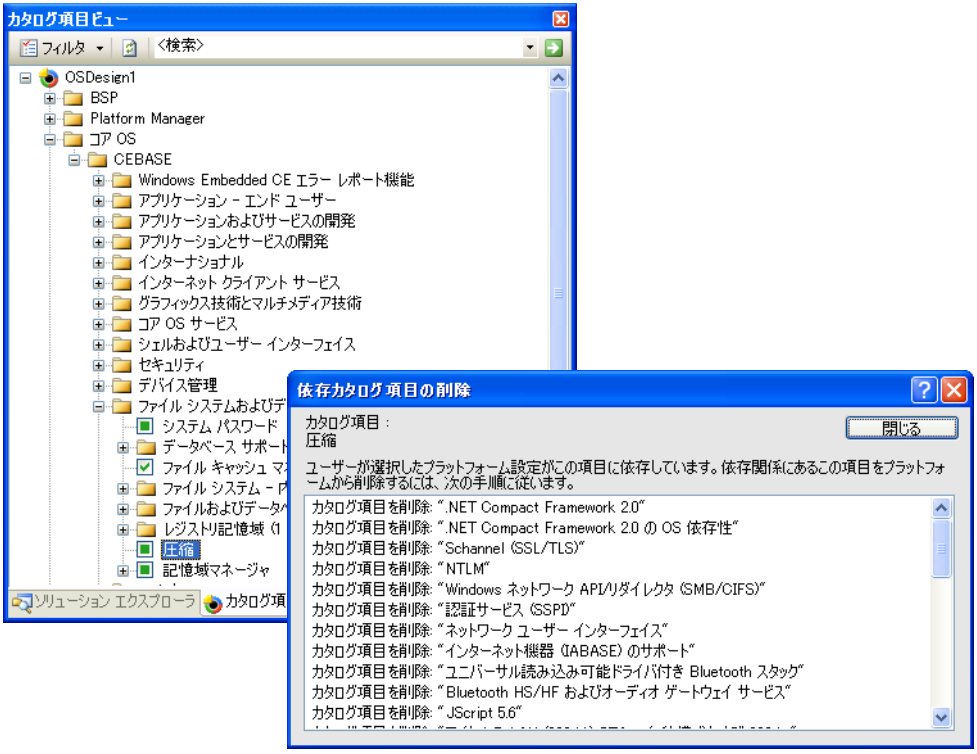


図 1-9　カタログ項目を依存した項目として含める理由

- **Internet Explorer 6.0 のサンプル ブラウザ カタログ項目のサポートを追加**

1. [カタログ項目ビュー] タブを選択して、OS デザイン カタログを表示します。フィルタ オプションが [カタログ内のすべてのカタログ項目] に設定されているかどうか確認します。
2. カタログ項目ビューの [フィルタ] ボタンの右側にある [検索] ボックスに、「Internet Explorer 6.0 のサンプル」と入力し、Enter キーを押すか、緑の矢印をクリックします。
3. 検索結果から Internet Explorer 6.0 の サンプル ブラウザ カタログ項目を探します。該当するチェック ボックスをオンにして、このカタログ項目を OS デザインに含めます (図 1-10)。

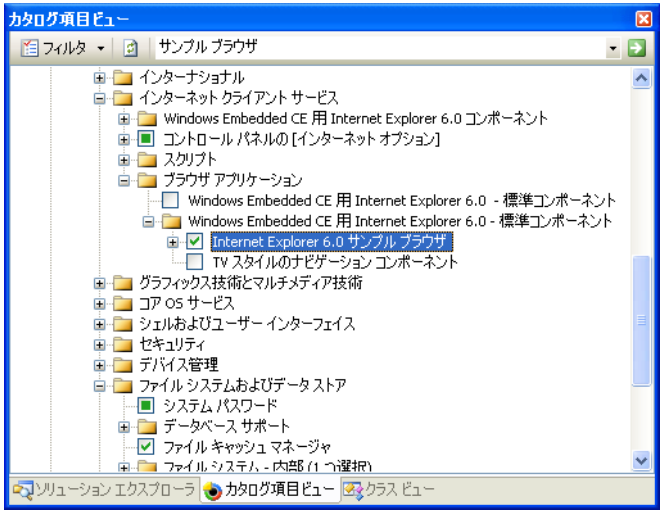


図 1-10 Internet Explorer 6.0 のサンプル ブラウザ カタログ項目を OS デザインに含める

- **OS デザインにおける管理コード開発のサポート**

1. [検索] ボックスに、「ipconfig」と入力して Enter キーを押します。
2. 検索結果からネットワーク ユーティリティ (IpConfig、Ping、Route) カタログ項目を探します。
3. ネットワーク ユーティリティ (IpConfig、Ping、Route) カタログ項目のチェック ボックスをオンにして、OS デザインに追加します。
4. [検索] ボックスに、「wceload 」と入力して Enter キーを押します。
5. 検索結果から CAB ファイル インストーラ/アンインストーラ カタログ項目を探します。このカタログ項目の SYSGEN 変数は wceload に設定されているため、検索結果に表示されるはずです。
6. CAB ファイル インストーラ / アンインストーラ カタログ項目を OS デザインに追加します。
7. 同じ要領で検索機能を使用し、.NET Compact Framework 1.0 への OS 依存関係を探します。.NET Compact Framework 1.0 への OS 依存関係カタログ項目が OS デザインに含まれているかどうか確認します (図 1-11)。

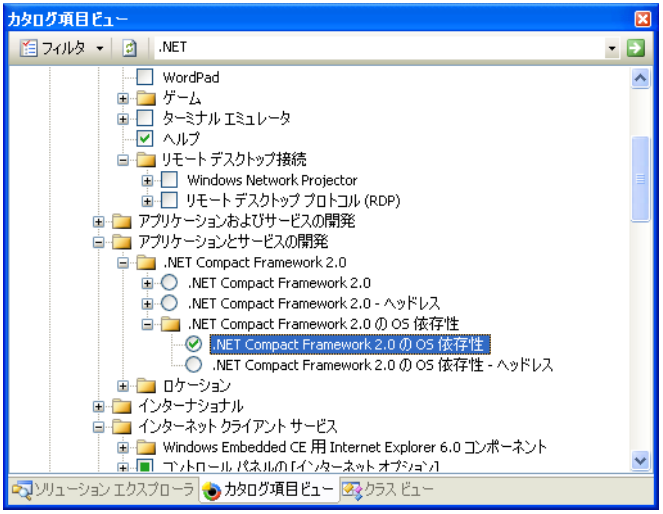


図 1-11　.NET Compact Framework 1.0 への OS 依存関係カタログ項目を OS デザインに含める

ノート　ヘッドレス .NET Compact Framework

このカテゴリには、2 つのコンポーネントがあります。ヘッドレス バージョンは、ディスプレイのないデバイス用ですので、記述にヘッドレス修飾子がないコンポーネントを必ず選択してください。

第 1 章の復習

Windows ィ Embedded CE 6.0 R2 をターゲットデバイスに展開するときは、必要なオペレーティング システム (OS) コンポーネント、機能、ドライバ、構成の設定を含む OS デザインを作成しなければなりません。Platform Builder を使用して、展開のために対応するランタイム イメージをビルドします。要件に応じてカスタマイズした OS デザインを作成する場合、次の最重要な作業を行う必要があります :

- Visual Studio の OS デザイン ウィザードを使用して、OS デザイン プロジェクトを作成する。
- 依存関係を使用して、手動で OS のコンポーネントを追加または削除する。
- カタログ エディタで環境変数と SYSGEN 変数を設定する。
- OS デザインをローカライズする地域の設定を構成する。
- カタログのコンポーネントを [カタログ項目の複製] をクリックして自動的に複製するか、または Sysgen Capture tool を使用して手動で複製する。
- OS デザインのカスタム SDK をエクスポートし、ターゲット デバイスのアプリケーションの依存関係を容易にする。

重要な用語

次の用語の意味を。巻末の用語集を調べ、回答の正誤を確認してみてください。

- OS デザイン
- コンポーネント
- SYSGEN 変数
- 環境変数
- ソフトウェア開発キット

推奨する練習内容

この章で扱っている試験範囲の内容を十分に理解するには、次の作業を完了してください :

カスタム OS デザインの作成

OS デザイン ウィザードを使用して、デバイス エミュレータ : ARMV4I BSP およびカスタム デバイス デザイン テンプレートに基づいて OS デザインを作成します。OS デザインが作成されたら、次の作業を行います :

- .Net Compact Framework 2.0 の追加　カタログ項目ビューの検索機能を使用して、このカタログ項目を追加します。
- ランタイム イメージのローカライズ　OS デザインのプロパティ ページを表示し、OS デザインをフランス語にローカライズします。

SDK の生成とテスト

ラボ 1 で生成した OS デザインを基に、次の作業を行います :

- バイナリ イメージのビルドと生成　リリース構成で生成した OS デザインのバイナリ イメージをビルドおよび生成します。
- SDK の作成とインストール　ビルド処理が正常に完了したかどうか確認してから、新しい SDK を作成してビルドし、アプリケーション開発用コンピュータにインストールします。
- SDK の使用　Visual Studio の別のインスタンスを使用して、Win32 Smart Device アプリケーションを作成します。カスタム SDK をプロジェクトの SDK 参照として使用し、アプリケーションをビルドします。