MR550 プログラミングマニュアル

ユニテック・ジャパン株式会社

2004.7 初版

目　　次

1 概要

本マニュアルは Window CE 用に提供されている各種の開発キットやツールについて、そしていろいろなプログラミングインタフェースの概要について説明しています。Windows CE のアプリケーションプログラムは、Windows CE デバイスに合った開発、エミュレート、そしてリモート・デバッグをするために標準の Microsoft 開発環境 IDE – Embedded Visual C++ と Embedded Visual Basic を含む Embedded Visual Tools (EVT) で開発することができます。ツールは Win32、COM、ActiveX、MFC、ATL をサポートしています。

Win32 API セットには、複数の機能が同じタスクを果たすことのできるという意味で重複した機能が含まれています。一般にデスクトップ PC ソフトウェアの開発者はわずかの重複 OS コードは気にしませんが、Window CE の開発者にとって、コードのサイズは重要です。Windows CE のリリースでは、Win32 API の特定のサブセットが提供されています。(サポートされている API の詳細に関しては、Embedded Visual Tools のリリースノートをご覧ください。)

Microsoft 社の Component Object Model、あるいは COM は、実行時にクェリーができるしっかりしたコンポーネントを作成するための標準が提供されています。各 COM オブジェクトは、インターフェースや関連した論理メソッドを公開しています。ベース インターフェース *IUnkown* は、サポートされたインターフェースについて特別なオブジェクトを COM オブジェクトがクェリーを行うことができます。COM モデルは言語には無関係で、コンポーネントは、コーラーに変更を要求することなく言語とは関係なく更新されます。

ActiveX コントロールは、特殊なタイプの COM オブジェクトです。Visual Basic 拡張 (VBX) で始まり、次に OLE コントロール (OCX) が続く拡張可能なコントロールの最新の仕様を表します。ActiveX コントロールは通常、ユーザ・インターフェースを表し、プロパティ、メソッド、そしてイベントを公開します。コントロールは、COM インターフェースの特定セットを通して Visual Basic などのコントロール コンテナと関係します。プロパティ、メソッド、およびイベントを公開することにより、ActiveX コントロールは、スクリプトで操作することができます。

Microsoft Foundation Classes (MFC) は、C++で Windows アプリケーションを開発するクラスライブラリです。これは Win32 API セットとほとんど同じ機能ですが、完全なオブジェクト指向アプリケーション・フレームワーク内にあります。Active Template Library (ATL) は、ActiveX コントロールとその他の COM コンポーネントを作成するために特にデザインされた C++ のテンプレート ライブラリです。テンプレート クラスを使うことで ATL は、継承 (インヘリタンス) のみを使う MFC より効率的にソフトウェアを作成できます。

主要な通信インターフェースは、Windows Sockets、あるいはシリアル接続、Ethernet、RF、あるいは赤外線ポートによる通信に TCP/IP を使う Winsock です。Windows CE はまた、シリアル API、Telephony API (TAPI)、Remote Access Service (RAS)、FTP と HTTP サービスを提供する WinINet API、ネットワーク サービスをエミュレートして接続の管理をする Windows ネットワーキング API などの通信関連 Win32 API セットもサポートします。

2 SDK とは?

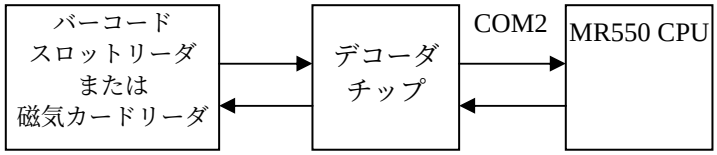
Microsoft は異なる Windows CE プラットフォーム用に 3 つの SDK (Pocket PC, Palm PC と H/PC Pro)を提供しています。これはサポートされるモジュール、GUI そして各プラットフォームのコンポーネントに違いがあるからです。ユーザがアプリケーションのコンパイルに不適当な SDK を使用すると予期できない問題が発生することがあります。ユーザはアプリケーションプログラムを開発する場合にターゲットプラットフォームに合った SDK を選択しなければなりません。Unitech のターミナルについても各プラットフォームにあった SDK を提供しております。皆様はユニテック・ジャパン(株)にお問い合わせになるか、以下の URL から最新バージョンを入手することができます。

http://adc.unitech.com.tw/pub/cs/software/MR550SDK/MR550SDK.zip

3 スキャナあるいは磁気カードリーダからデータを読み込む方法

3.1 概要)

MR550 と標準の HPC/PalmPC の大きな違いはバーコード入力ができるかどうかです。Windows CE のリファレンスマニュアルにはバーコード入力についての情報は含まれていません。このセクションはバーコードサブシステムのプログラミング構造と MR550 のプログラミング・ユーティリティ・ライブラリを紹介しています。 MR550 の内部には、バーコードスロットリーダと磁気カードリーダのコントロールとバーコードのデコードを行うための高機能なデコーダチップがあります。以下は MR550 バーコードのシステム図です:



上記の図にあるように、MR550 はシリアル ポート COM2 経由でデコーダチップと通信します。その通信パラメータは、38400,N,8,1 に固定されています。一般にデコーダチップは、COM2 がオープンしていないときはスリープ状態です。デコーダチップは COM2 がオープンすると作動し、カードが読み込まれるとバーコードスロットリーダからのバーコード“信号”がデコードされます。デコードの後は、バーコード データとそのバーコードのタイプが直接 MR550 に送られます。

プログラマの皆様が、特に、バーコードやシリアルポートのコントロールに慣れていない場合は、プログラム言語だけでデコーダチップをコントロールすることが難しいことに気づかれるでしょう。このため、ユニテック社はデコーダチップをコントロールするためにユーザあるいはアプリケーションのプログラマの皆さんに以下のユーティリティライブラリとプログラムを提供しています。

1. アプリケーションプログラム“Bar2Key.exe”はレーザースキャナから入力データを読み、そして MR550 のキーボードバッファに直接データを入力する便利なアプリケーションプログラムです。 Bar2Key.exe はバーコードデータの入力を簡単にし、そして COM ポートのプログラミングに慣れていないプログラマに特に役に立ちます。ユーザプログラムは単にキーボードからバーコードデータを読みます。バーコードシンボルの設定については、サポートされているシンボルと区切り文字のすべてを定義するために BARSETUP プログラムを実行することができます。
2. ユーティリティライブラリ
   MR550 用に二つのライブラリ“Scankey3.dll”と“Scanner3.dll”があります。
   A. **Scanner3.dll**：この DLL はデコーダチップからスキャンしたデータを得るための関数コールを提供します。 (関数の詳細な説明はセクション **3.2** をご覧下さい)

B. **Scankey3.dll：** この DLL は入力されたスキャンデータを標準のキーボードバッファに入力するための関数コールを提供します。 (関数の詳細な説明についてはセクション **3.3** をご覧下さい。)

3.2 “Scanner3.dll”の関数コールリスト

“Scanner3.lib”と“scanner3.h”はアプリケーションをコンパイルするVCプログラマに必要なファイルです。これらのファイルはSDKをインストール後、標準のLIBとINCLUDEフォルダにあります。

3.2.1 デコーダを使用可能にする

関数の説明: この関数は COM2 ポートをオープンし、デコーダチップからのバーコード入力を得るためのスレッドを作成し、そしてシステムバッファに入力データを保存します。アプリケーションはシステムバッファから入力データを得るために関数コール**PT_GetBarcode()**を使用することができます。

関数コール: INT PT_EnableBarcode(VOID);

戻りコード:
- =1 新しいスレッドの作成に失敗
- =2 使用可能に出来ない
- =3 COM2 を開くことが出来ない
- =4 Hamster からの パラメータ アップロード失敗
- =0 OK

3.2.2 デコーダを使用不可能にする

関数の説明: この関数は COM2 ポートを閉じ、**PT_EnableBarcode()** によって作成されたスレッドを消去します。

関数コール: VOID PT_DisableBarcode(VOID);

3.2.3 バーコード入力チェック

関数の説明: この関数はデコーダチップによって正しくデコードされたバーコードデータがシステムバッファにあるかどうかをチェックするために使用します。

関数コール: BOOL PT_CheckBarcode(VOID);

戻りコード:
- =TRUE システムバッファに入力データがある。
- =FALSE システムバッファにデータがない。

3.2.4 バーコードデータを読む

関数の説明: システムバッファから入力したバーコードデータとそのタイプを得る。

関数コール: BOOL PT_GetBarcode(TCHAR *szBarcodeBuffer,TCHAR *cType);

パラメータ: (出力)

szBarcodeBuffer : 入力データを保存するためのストリングバッファ

cType : 入力データのタイプ

- =00H Full Code 39
- =01H STD Code 39
- =02H EAN-13
- =03H UPC-A
- =04H EAN-8
- =05H UPC-E
- =06H I-25
- =07H CODABAR
- =08H Code 128
- =09H Code 93
- =0Ah STD 25

=0BH　　MSI
=0CH　　EAN-128
=0DH　　Code 32
=0EH　　DELTA
=0FH　　LABEL
=10H　　PLESSEY
=11H　　Code 11
=12H　　TOSHIBA

戻りコード: TRUE　=　バーコード入力がある
FALSE　=　バーコード入力がない

3.2.5磁気ストライプリーダ(MSR)入力のチェック

関数の説明: この関数はデコーダチップによって正しくデコードされた MSR データがシステムバッファにあるかどうかをチェックするために使用されます。

関数コール: BOOL PT_CheckMSR()

戻りコード: TRUE　=　MSR 入力がある
FALSE　=　MSR 入力がない

3.2.6磁気ストライプリーダ(MSR)データを読む

関数の説明: 磁気ストライプリーダ (MSR) 入力を読む。

関数コール: BOOL PT_GetMSR(TCHAR *szTrack1,TCHAR *szTrack2,TCHAR *szTrack3, TCHAR *cType)

パラメータ: (出力)

szTrack1: トラック 1 からの入力を保存するためのストリングバッファ
szTrack2: トラック 2 からの入力を保存するためのストリングバッファ
szTrack3: トラック 3 からの入力を保存するためのストリングバッファ
cType　: MSR 入力データのインジケータ

1　　トラック 1 のみ
2　　トラック 2 のみ
3　　トラック 3 のみ
4　　トラック 1 とトラック 2
5　　トラック 2 とトラック 3
6　　トラック 1、トラック 2 とトラック 3

戻りコード: TRUE　=　MSR 入力がある
FALSE　=　MSR 入力がない

3.2.7 デコーダチップにバーコードシンボルをセット

関数の説明: この関数は個々のバーコードシンボルを設定するために使用されます。
関数コール: BOOL PT_SetBarcodePara(unsigned char type, unsigned char status);
戻り) : TRUE ---)) 設定 OK.
FALSE -)) 設定失敗
パラメータ:) (入力)

シンボル	タイプ	ステータス	動作
Code39 全設定	0x39	Bit0 = 0	使用不可
		Bit0 = 1	*使用可能*
		Bit1 = 0	*全 Code39*
		Bit1 = 1	標準 Code39
		Bit3&2 = 0	チェックデジットを計算して送信
		Bit3&2 = 1	チェックデジットを計算するが送信しない
		Bit3&2 = 2	*チェックデジットを計算しない*
		Bit4 = 0	スタートとストップビットを送信
		Bit4 = 1	*スタートとストップビットを送信しない*
Code39	0x3A	0	使用不可
		1	*使用可能*
Code39 全 ASCII	0x3B	*0*	*使用可能*
		1	使用不可
Code39 チェックデジット	0x3C	0	チェックデジットを計算して送信
		1	チェックデジットを計算するが送信しない
		2	*チェックデジットを計算しない*
Code39 SS	0x3D	0	スタートとストップビットを送信
		1	*スタートとストップビットを送信しない*
Code39 最少長	0x3E	*0 ~ 48*	標準値 = 0
Code39 最大長	0x3F	*0 ~ 48*	標準値 = 48

)

シンボル)	タイプ)	ステータス)	動作)
I25)	0x40)	Bit0) =) 0)	使用不可)
		Bit0) =) 1)	***使用可能)***
		Bit1) =) 0)	固定長可))
		Bit1) =) 1)	***固定長使用不可)))***
		Bit2&3) =) 0)	チェックデジットを計算して送信)
		Bit2&3) =) 1)	チェックデジットを計算するが送信しない)
		Bit2&3) =) 2)	***チェックデジットを計算しない)***
		Bit4&5) =) 0)	最初の桁をサプレス)
		Bit4&5) =) 1)	最後の桁をサプレス)
		Bit4&5) =) 2)	***サプレスしない)***
I25)	0x41)	0)	使用不可)
		1)	***使用可能)***
I25) 固定長)	0x42)	0)	固定長可))
		1)	***固定長使用不可)***
I25) CD)	0x43)	0)	チェックデジットを計算して送信)
		1)	チェックデジットを計算するが送信しない)
		2)	***チェックデジットを計算しない)***
I25) SS)	0x44)	0)	最初の桁をサプレス)
		1)	最後の桁をサプレス)
		2)	***サプレスしない)***
I25) 最少長)	0x45)	**2) ¯) 64)**	**標準値) =) 10)**
I25) 最大長)	0x46)	**2) ¯) 64)**	**標準値) =) 64)**
S25&Toshiba)	0x47)	***Bit0) =) 0)***	***使用不可)***
		Bit0) =) 1)	使用可能)
		Bit1) =) 0)	***固定長可))***
		Bit1) =) 1)	固定長使用不可)))
		Bit2&3) =) 0)	チェックデジットを計算して送信)
		Bit2&3) =) 1)	チェックデジットを計算するが送信しない)
		Bit2&3) =) 2)	***チェックデジットを計算しない)***
S25&Toshiba)	0x48)	***0)***	***使用不可)***
		1)	使用可能)
S25&Toshiba) 固定長)	0x49)	***0)***	***固定長可)***
		1)	固定長使用不可)
S25&Toshiba) CD)	0x4A)	0)	チェックデジットを計算して送信)
		1)	チェックデジットを計算するが送信しない)
		2)	***チェックデジットを計算しない)***
S25&Toshiba) 最少長)	0x4B)	1) ¯) 48)	標準値) =) 4)
S25&Toshiba) 最大長)	0x4C)	1) ¯) 48)	標準値) =) 48)

シンボル	タイプ	ステータス	動作
Code32	0x4d	***Bit0 = 0***	***使用不可***
		Bit0 = 1	使用可能
		Bit1 = 0	***先頭文字を送信***
		Bit1 = 1	先頭文字を送信しない
		Bit2 = 0	***末尾文字を送信***
		Bit2 = 1	末尾文字を送信しない
Code32	0x4e	***0***	***使用不可***
		1	使用可能
Code32 先頭文字送信	0x4f	***0***	***先頭文字を送信***
		1	先頭文字を送信しない
Code32 末尾文字送信	0x50	***0***	***末尾文字を送信***
		1	末尾文字を送信しない
Telpen	0x51	***Bit0=0***	***使用不可***
		Bit0=1	使用可能
		Bit1=0	***文字セット = 標準***
		Bit1=1	文字セット = 数字
Telpen	0x52	***0***	***使用不可***
		1	使用可能
Telpen 文字セット	0x53	***0***	***標準***
		1	数字
EAN 128/UCC	0x54	0	使用不可
		1	***使用可能***
EAN 128/UCC	0x55	0	使用可能
		1	***使用不可***
EAN 128/UCC Code ID	0x56	***0***	***使用不可***
		1	使用可能
EAN 128/UCC ダブルラベルのセパレータ	0x57	ASCII	ASCII 文字
Code 128	0x58	0	使用不可
		1	***使用可能***
Code 128 最少長	0x59	0 ~ 64	標準値 = 1
Code 128 最大長	0x5a	0 ~ 64	標準値 = 64

)

シンボル)	タイプ)	ステータス)	動作)
MSI/Plessey)	0x5B)	***bit0=0)***	***使用不可)***
		bit0=1)	使用可能)
		bit1=0)	チェックデジット送信)
		bit1=1)	***チェックデジット送信しない)***
		bit3&2=0)	***チェックデジット・ダブルモジュール) 10)***
		bit3&2=1)	チェックデジット・モジュール) 11 プラス) 10)
		bit3&2=2)	チェックデジット・モジュール 10)
MSI/Plessey)	0x5C)	***0)***	***使用不可)***
		1)	使用可能)
MSI/Plessey) CD)	0x5D)	0)	チェックデジット送信)
		1)	***チェックデジット送信しない)***
MSI/Plessey) CDMODE)	0x5E)	***0)***	***チェックデジット・ダブルモジュール) 10)***
		1)	チェックデジット・モジュール) 11 プラス) 10)
		2)	チェックデジット・モジュール 10)
MSI/Plessey) 最少長)	0x5F)	***1) ~) 64)***	***標準値) =) 1)***
MSI/Plessey) 最大長)	0x60)	***1) ~) 64)***	***標準値) =) 64)***
Code93)	0x61)	0)	使用不可)
		1)	***使用可能)***
Code93) 最少長)	0x62)	1) ~) 48)	標準値) =) 1)
Code93) 最大長)	0x63)	1) ~) 48)	標準値) =) 48)
Code11)	0x64)	***bit0=0)***	***使用不可)***
		bit0=1)	使用可能)
		bit1=0)	1) チェックデジット)
		bit1=1)	***2) チェックデジット)***
		bit2=0)	チェックデジット送信)
		bit2=1)	***チェックデジット送信しない)***
Code11)	0x65)	***0)***	***使用不可)***
		1)	使用可能)
Code11) チェックデジット数))))	0x66)	0)	1) チェックデジット)
		1)	***2) チェックデジット)***
Code11) CD)	0x67)	0)	チェックデジット送信)
		1)	***チェックデジット送信しない)***
Code11) 最少長)	0x68)	1) ~) 48)	標準値) =) 1)
Code11) 最大長)	0x69)	1) ~) 48)	標準値) =) 48)

シンボル	タイプ	ステータス	動作
CodaBar	0x6a	*Bit0 = 0*	*使用不可*
		Bit0 = 1	使用可能
		Bit1 = 0	スタートとストップ文字を送信
		Bit1 = 1	*スタートとストップ文字を送信しない*
		Bit2&3 = 0	チェックデジットを計算して送信
		Bit2&3 = 1	チェックデジットを計算するが送信しない
		Bit2&3 = 2	*チェックデジットを計算しない*
		Bit4 = 0	CLSI フォーマットオン
		Bit4 = 1	*CLSI フォーマットオフ*
CodaBar	0x6b	*0*	*使用不可*
		1	使用可能
CodaBar SS	0x6c	0	スタートとストップ文字を送信
		1	*スタートとストップ文字を送信しない*
CodaBar CD	0x6d	0	チェックデジットを計算して送信
		1	チェックデジットを計算するが送信しない
		2	*チェックデジットを計算しない*
Codabar CLSI フォーマット	0x6e	0	オン
		1	*オフ*
Codabar 最少長	0x6f	3 ~ 48	標準値 = 3
Codabar 最大長	0x70	3 ~ 48	標準値 = 48
Label Code	0x71	*Bit0 = 0*	*使用不可*
		Bit0 = 1	使用可能
		Bit1 = 0	*チェックデジット送信*
		Bit1 = 1	チェックデジット送信しない
Label Code	0x72	*0*	*使用不可*
		1	使用可能
Label Code CD	0x73	*0*	*チェックデジット送信*
		1	チェックデジット送信しない
UPC-A	0x74	bit0=0	使用不可
		bit0=1	*使用可能*
		bit1=0	*先頭桁を送信*
		bit1=1	先頭桁を送信しない
		bit2=0	*チェックデジットを送信*
		bit2=1	チェックデジットを送信しない
UPC-A	0x75	0	使用不可
		1	*使用可能*
UPC-A LD	0x76	*0*	*先頭桁を送信*
		1	先頭桁を送信しない
UPC-A CD	0x77	*0*	*チェックデジットを送信*
		1	チェックデジットを送信しない

)

シンボル)	タイプ)	ステータス)	動作)
UPC-E)	0x78)	bit0=0)	使用不可)
		bit0=1)	***使用可能)***
		bit1=0)	***先頭桁を送信)***
		bit1=1)	先頭桁を送信しない)
		bit2=0)	チェックデジット送信)
		bit2=1)	***チェックデジット送信しない)***
		bit3=0)	ゼロ表示オン)
		bit3=1)	***ゼロ表示オフ)***
		bit4=0)	***NSC) 使用可能)***
		bit4=1)	NSC) 使用不可)
UPC-E)	0x79)	0)	使用不可)
		1)	***使用可能)***
UPC-E) LD)	0x7a)	***0)***	***先頭桁を送信)***
		1)	先頭桁を送信しない)
UPC-E) CD)	0x7b)	0)	チェックデジット送信)
		1)	***チェックデジット送信しない)***
UPC-E) 拡張)	0x7c)	0)	ゼロ表示オン)
		1)	***ゼロ表示オフ)***
UPC-E) NSC)	0x7d)	***0)***	***使用不可)***
		1)	使用可能)
EAN13)	0x7e)	bit0=0)	使用不可)
		bit0=1)	***使用可能)***
		bit1=0)	***先頭桁を送信)***
		bit1=1)	先頭桁を送信しない)
		bit2=0)	***チェックデジット送信)***
		bit2=1)	チェックデジット送信しない)
		Bit3=0)	ブックランド有効)
		Bit3=1)	***ブックランド無効)***
EAN13)	0x7F)	0)	使用不可)
		1)	***使用可能)***
EAN13) LD)	0x80)	***0)***	***先頭桁を送信)***
		1)	先頭桁を送信しない)
EAN13) チェックデジット)	0x81)	***0)***	***チェックデジット送信)***
		1)	送信しない)
EAN13) ブックランド)	0x82)	***0)***	***使用可能)***
		1)	使用不可)

)

シンボル)	タイプ)	ステータス)	動作)
EAN8)	0x83)	Bit0=0)	使用不可)
		Bit0=1)	***使用可能)***
		Bit1=0)	***先頭桁を送信)***
		Bit1=1)	先頭桁を送信しない)
		Bit2=0)	***チェックデジット送信)***
		Bit2=1)	チェックデジット送信しない)
EAN8)	0x84)	0)	使用不可)
		1)	***使用可能)***
EAN8) LD)	0x85)	*0)*	***先頭桁を送信)***
		1)	先頭桁を送信しない)
EAN8) CD)	0x86)	*0)*	***チェックデジット送信)***
		1)	チェックデジット送信しない)
Supplement) 2) /) 5)	0x87)	***bit0=0)***	***Supplement) 2) 使用不可)***
		bit0=1)	Supplement) 2) 使用可能)
		Bit1=0)	***Supplement) 5) 使用不可)***
		Bit1=1)	Supplement) 5) 使用可能)
		Bit2=0)	***あれば送信)***
		Bit2=1)	必ず送信)
Supplement) 2)	0x88)	***Bit0=0)***	***使用不可)***
		Bit0=1)	使用可能)
Supplement) 5)	0x89)	***Bit0=0)***	***使用不可)***
		Bit0=1)	使用可能)
Supplement) MH)	0x8A)	***Bit0=0)***	***あれば送信)***
		Bit0=1)	必ず送信)
Supplement) SSI)	0x8B)	Bit0=0)	スペースセパレータ挿入)
		Bit0=1)	***挿入しない)***
Delta) Code)	0x8C)	***Bit0=0)***	***使用不可)***
		Bit0=1)	使用可能)
		Bit1=0)	***チェックデジット計算)***
		Bit1=1)	チェックデジット計算しない)
		Bit2=0)	チェックデジット送信)
		Bit2=1)	***チェックデジット送信しない)***
Delta) Code)	0x8D)	***Bit0=0)***	***使用不可)***
		Bit0=1)	使用可能)
Delta) Code) CDC)	0x8E)	***Bit0=0)***	***チェックデジット計算)***
		Bit0=1)	チェックデジットを計算しない)
Delta) Code) CDD)	0x8F)	Bit0=0)	チェックデジットを送信)
		Bit0=1)	***チェックデジットを送信しない)***
MSR)	6)	*0)*	*全トラック)*
		1)	トラック) 1) とトラック) 2)
		2)	トラック) 1) とトラック) 3)
		3)	トラック) 2) とトラック) 3)
		4)	トラック) 1) のみ)
		5)	トラック) 2) のみ)
		6)	トラック) 3) のみ)

3.2.8 デコーダチップからバーコートシンボル設定を得る

関数の説明: この関数はデコーダチップから個々のバーコードシンボルを得るために使用されます。

関数コール: BOOL PT_GetBarcodePara(unsigned char type, int *status);

入力/出力: セクション 3.2.7の表をご覧下さい。

戻り) : TRUE --- OK.
FALSE - 失敗

3.2.9 DLL バージョン番号を得る

関数の説明: この関数は DLL のバージョン番号を得るために使用されます。

関数コール: INT PT_DllVersion(void);

戻り) : Integer

3.2.10 すべてのシンボルを標準値にリセット

関数の説明: この関数コールはデコーダチップのシンボル設定を標準値にリセットします。

VC 用関数コール: int PT_SetToDefault (VOID)

VB 用関数コール: PT_SetToDefault

3.3)“Scankey3.dll”)の関数コールリスト)

ユニテック社では以下の三つのファイルを提供しています。))

)　"Scankey3.lib"))　VC)プログラミング用)

)　"Scankey3.h"))　VC)プログラミング用)

)　"Scankey3.bas"))　VB)プログラミング用)

)

3.3.1 デコーダを有効にする

関数の説明:)この関数は)COM2)ポートを開き、デコーダチップからバーコード入力を得るためにスレッドを作成し、そしてスキャンしたデータをキーボードバッファに送ります。ユーザアプリケーションは標準のキーボード入力としてデータを得ることが出来ます。))

)

VC 用関数コール:))int)PT_EnableBarToKey(VOID))

VB 用関数コール:))PT_EnableBarToKey())as)Long)

戻りコード:)=1)　新しいスレッド作成に失敗)

=2)　再度有効にできない)

=3)　COM2 を開けない)

=4)　Hamster からパラメータのアップロード失敗)

=0)　OK)

3.3.2 デコーダを無効にする

関数の説明:)この関数は COM2 ポートを閉じ、**PT_EnableBarToKey()**で作成されたスレッドを消去します。)

VC 用関数コール:)VOID)PT_DisableBarToKey)())VOID))))

VB 用関数コール:)PT_DisableBarToKey)())

3.3.3 バーコードシンボルをデコーダチップにセット

関数の説明:)この関数は個々のバーコードシンボルをセットするために使用されます。)

VC 用関数コール:))BOOL)PT_SetBarToKeyPara(unsigned)char),unsigned)char))))

VB 用関数コール:))PT_SetBarToKeyPara)(ByVal)strType)As)String,)ByVal)strStatus)As)String))

As)Boolean)

戻り):)　TRUE)---)設定)OK.)

FALSE)-)設定失敗)

パラメータ:))**(入力)**)セクション)3.2.7と同じ。))

3.3.4 デコーダチップからバーコードシンボル設定を得る

関数の説明:)この関数はデコーダチップから個々のバーコードシンボルを得るために使用されます。)

VC 用関数コール:))BOOL)PT_GetBarToKeyPara(unsigned)char)cType,int*)iStatus))

VB 用関数コール:))VB 用の関数コールはありません。)

入力/出力:)セクション)**3.2.7**の表をご覧下さい。)

戻り):)　TRUE)---)OK.)

FALSE)-)失敗)

3.3.5)DLL)バージョン番号を得る)

関数の説明:)この関数は)DLL)バージョン番号を得るために使用されます。)

VC 用関数コール: INT PT_Version(void);
VB 用関数コール: PT_Version As Long
戻り: Integer

3.3.6 プリアンブル文字列をセット

関数の説明: この関数は このファンクションは、“Scankey3.dll”がキーボードバッファにスキャンしたデータの前にプリアンブル文字列を送らせます。標準のプリアンブル文字列は NULL で、その最大長は 48 文字です。標準のプリアンブル文字列 NULL で、最大長は 48 文字です。
VC 用関数コール: void PT_SetPreamble(TCHAR *sPreambleStr)
VB 用関数コール: void PT_SetPreamble(ByVal strPreamble As String)
入力: プリアンブル文字列のストリングバッファ
戻り: なし

3.3.7 プリアンブル文字列を得る

関数の説明: 現在のプリアンブル文字列を得る
VC 用関数コール: void PT_GetPreamble(TCHAR *) sPreambleStr)
VB 用関数コール: No 関数コール available for VB
出力: プリアンブル文字列のストリングバッファ
戻り: なし

3.3.8 ポストアンブル文字列をセット

関数の説明: この関数は、“Scankey3.dll”がキーボードバッファにスキャンしたデータの後にポストアンブル文字列を送らせます。標準のポストアンブル文字列は NULL で、その最大長は 48 文字です。
VC 用関数コール: void PT_SetPostamble(TCHAR *sPostambleStr)
VB 用関数コール: void PT_SetPostamble(ByVal strPostamble As String)
入力: ボスとアンブル文字列のストリングバッファ
戻り: なし

3.3.9 ポストアンブル文字列を得る

関数の説明: 現在のポストアンブル文字列を得る
VC 用関数コール: void PT_GetPostamble(TCHAR *) sPostambleStr)
VB 用関数コール: VB 用の関数コールはありません。
出力: ポストアンブル文字列のストリングバッファ
戻り: なし

3.3.10 区切り文字のセット

関数の説明: このファンクションは、“Scankey3.dll”がキーボードバッファにポストアンブルの後に区切り文字を追加させます。標準の区切り文字は NULL で、その最大長は 10 文字です。
VC 用関数コール: void PT_SetDelimiter(TCHAR *sDelimiterStr)
VB 用関数コール: PT_SetPostamble (ByVal strPostamble As String)
入力: 区切り文字列のストリングバッファ
戻り: なし

3.3.11 区切り文字列を得る

関数の説明: 現在の区切り文字列を得る

VC 用関数コール: void WINAPI PT_GetDelimiter(TCHAR *sDelimiterStr)
VB 用関数コール: VB 用の関数コールはありません
出力: 区切り文字列のストリングバッファ
戻り: なし

3.3.12 すべてのシンボルを標準値にセット

関数の説明: この関数コールはデコーダチップのシンボル設定をシステムの標準値にリセットします。

VC 用関数コール: int PT_SetToDefault (VOID)
VB 用関数コール: PT_SetToDefault

4 MR550 プログラミングのための便利なライブラリ

MR550 はリレー、フォトカップル入力、セキュリティアラーム等の I/O デバイスをコントロールするための GIOAPI.DLL も提供しています。

4.1.1 リレー1 コントロール

関数の説明: この関数はリレーポート 1 のステータスをコントロールするために使用されます。

関数コール: DWORD GIO_Relay1 (BOOL status)

パラメータ: (入力)

status : TRUE => リレー 1 オープン
FALSE => リレー 1 クローズ

戻りコード: なし

4.1.2 リレー 2 コントロール

関数の説明: この関数はリレーポート 2 のステータスをコントロールするために使用されます。

関数コール: DWORD GIO_Relay2 (BOOL status)

パラメータ: (入力)

status : TRUE => リレー 2 オープン
FALSE => リレー2 クローズ

戻りコード: なし

4.1.3 リレー1 のステータスを得る

関数の説明: この関数はリレーポート 1 のステータスを得るために使用されます。

関数コール: DWORD GIO_Relay1Status ()

パラメータ: なし

戻りコード:

0 => リレー 1 は開かれている
1 => リレー 1 は閉じられている

4.1.4 リレー 2 のステータスを得る

関数の説明: この関数はリレーポート 2 のステータスを得るために使用されます。

関数コール: DWORD GIO_Relay2Status ()

パラメータ: なし

戻りコード:

0 => リレー2 は開かれている
1 => リレー2 は閉じている

4.1.5 デジタル入力を得る

関数の説明:) この関数はデジタル入力(フォトカップル)のステータスを得るために使用されます。

関数コール:) DWORD GIO_Digit()

パラメータ:) なし

戻りコード:)

0 => low

1 => high

4.1.6 セキュリティアラームを有効にする

関数の説明:) この関数はセキュリティアラームを有効なするために使用されます。従って、MR550 はバックプレートが開けられたら音を発します。

関数コール:) DWORD GIO_EnableAlarm()

パラメータ:) なし

戻りコード:) なし

4.1.7 セキュリティアラームを無効にする

関数の説明:) この関数はセキュリティアラームを無効にするために使用されます。従って、MR550 はバックプレートが開けられても音を出しません。

関数コール:) DWORD GIO_DisableAlarm()

パラメータ:) なし

戻りコード:) なし

5 Visual C/C++ with SDK または MFC によるプログラミング

5.1 システムの必要条件:

HW	CPU	Pentium 90-MHz 以上のプロセッサを推奨。
	メモリ	Windows NT Workstation 4.0, Windows 2000 - 128MB Windows 95/98 - 64MB
	CDROM ドライブ	マルチメディア・ディスクトップ・コンピュータ仕様に対応しているもの
	ディスプレイ	VGA 以上の解像度が必要。Super VGA モニタを推奨
	マウス	マウスまたは相当品。
SW	OS	Microsoft® Windows NT/2000/XP ® Workstation 4.0, Microsoft® Windows® 95/98/Me
	開発ツール	Microsoft eMbedded Visual Tools および MR550 SDK for VC/VB

注意: 開発環境には Microsoft Windows 2000, XP, NT Workstation 4.0 をデバッグ用のホストとして推奨いたします。ツールキットは Windows 95/98/Me にインストールすることができ、これらからアプリケーションを構築することができます。*しかし、エミュレーションは Windows 95/98/Me では動作しません; 代わりにホスト機として Windows NT,2000,XP を使用して下さい。*

5.2 インストール

ユーザはアプリケーション・プログラムの開発に Microsoft eMbedded Visual Tools (EVT) と MR550 SDK for VC++ を使用して下さい。

Microsoft® eMbedded Visual C++® 3.0 はデスクトップ Visual C++アプリケーション開発で使用するのと同様な統合開発環境(IDE)を使用して Windows CE ベースのアプリケーションを開発することができます。しかし、この IDE はアプリケーションを作成、テスト、および改良するために使用する多数の標準開発ツールの Windows CE 専用のバージョンを含んでいます。これは Windows CE プラットフォームとデバイスに独特な新しいソフトウェアを開発するための各種ツールも含んでいます。

ユーザ独自の Windows CE アプリケーションを開発するために、eMbedded Visual C++ IDE は学習が容易で、他の Visual C++ファミリの経験のあるプログラマはすぐに慣れることができます。アプリケーションは Handheld PC Pro (H/PC Pro)、Palm-size PC 1.2、Pocket PC そして MR550 プラットフォーム上で実行するために eMbedded Visual C++ で作成することができます。独自の Windows CE ベースのプラットフォーム、あるいは Windows CE ベースのプラットフォームをシミュレートするデスクトップ・エミュレータの中で実行するアプリケーションを作成するために eMbedded Visual C++ を使用することもできます。

Microsoft eMbedded Visual Tools のインストールが終わった後で、ダイアローグボックスは H/PC と Palm-size PC の SDK をインストールするための入力要求を出します。この時点でこれらの SDK をインストールする場合は、“ はい ” を選択して下さい。)

これは Microsoft eMbedded Visual Tools のセットアップ・プログラムから立ち上がり、その結果 H/PC、Palm-size PC と Pocket PC 用の Windows CE SDK のセットアップが立ち上がります。三つの設定プログラムのすべてを正しいインストール作業を行うために行わなくてはなりません。Windows CE 用の MFC を使用したい場合は、これらの SDK をインストールしなければなりません。

さらに、ユニテック社は当社の MR550 プラットフォーム用の SDK も提供しています。ユニテックの提供する CD-ROM のフォルダ \SDK\VC からインストール・プログラムを実行することができます。ユニテックの SDK は Palm-Size SDK と特別なスキャナ・ライブラリ を含んでおり、これは"scanner3.lib", "scanner3.h", "scankey3.lib" と "scankey3.h" を皆様の開発用のフォルダにコピーする必要がないことを意味しています。言い換えると、これらは標準の \lib と \include フォルダに置かれます。)

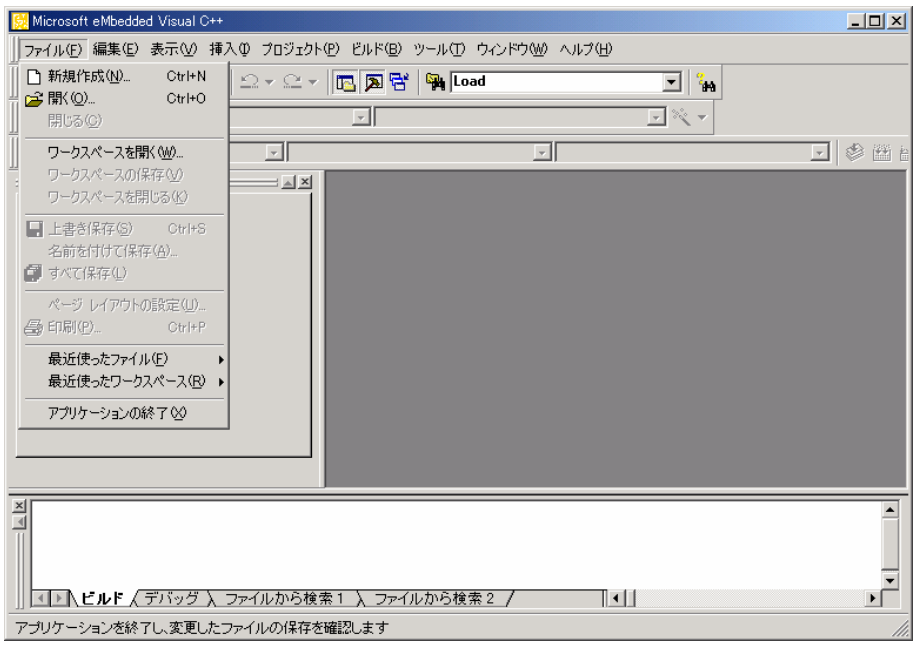
注意: Windows CE Toolkit for Visual C++ が正しく機能するために少なくとも一つの Windows CE SDK をインストールしなければなりません。

より詳しい情報については、eMbedded Visual C++のリリース文書をご覧下さい。

5.3 VC++で Windows CE プログラムを作成する方法

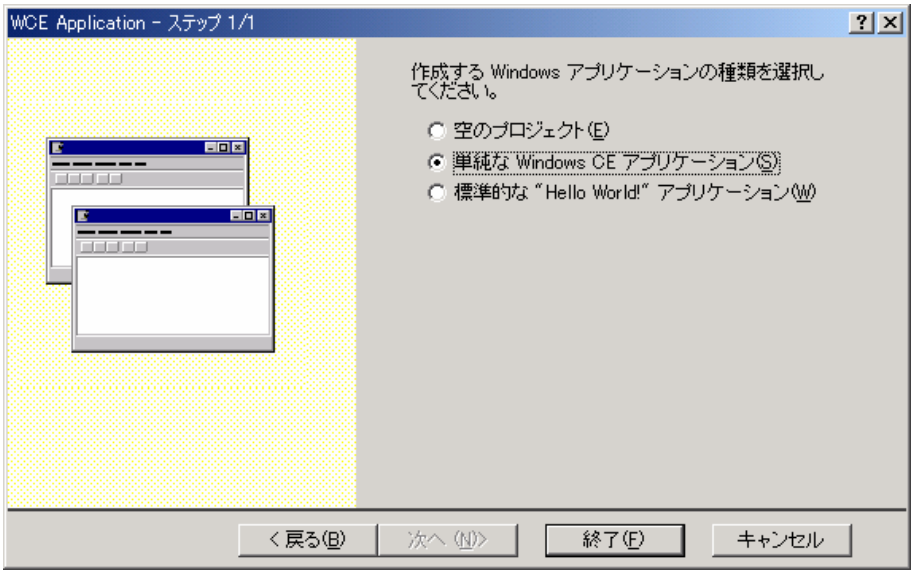
アプリケーションを作成するために以下のステップに従って下さい:

1. eMbedded Visual C++を実行します。
2. メインメニューから“ ファィル / 新規作成 ”を選択します。

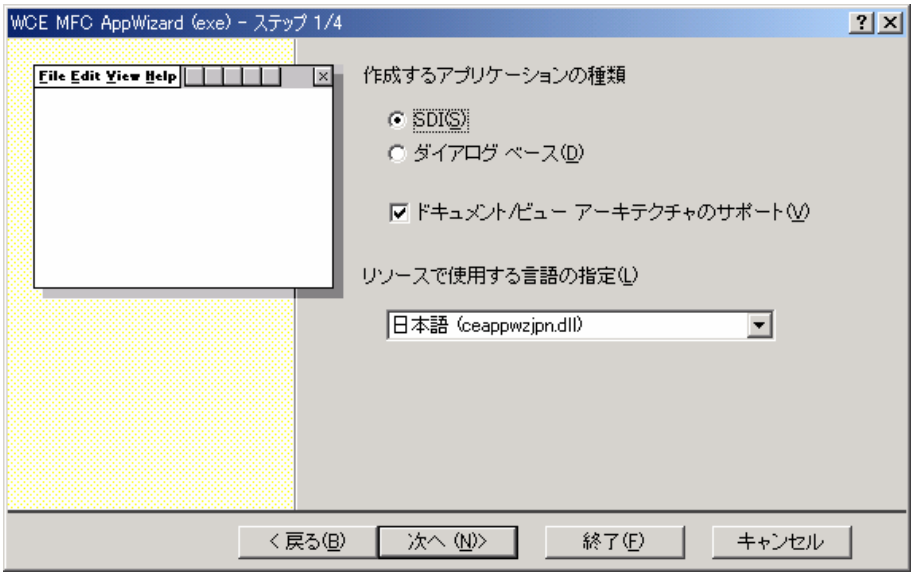


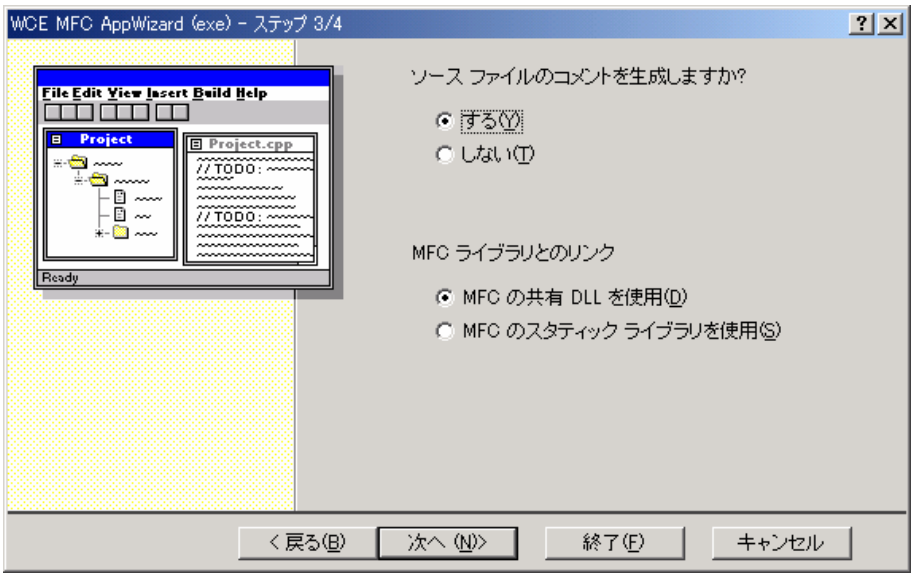
ダイアローグボックスが AP タイプ選択のためにポップアップします。Windows CE アプリケーションについては、選択可能ないくつかのプロジェクトタイプがあります。ユーザは“ WCE application ”を選択することによって SDK プログラムを作成するか、“ WCE MFC AppWizard (exe) ”を選択することによって MFC プログラムを作成、もしくは他の 7 オプションのいずれかを選択します。“ プロジェクト名 ”にこのプロジェクトの適当な名前を入力して、“ CPU ”の下で“ Win32 (WCE MIPS) ”を選択します。ここで次のステップにすすむために“ OK ”ボタンをクリックします。

WCE Application を選択した場合、適当な項目をクリックあるいは選択して、“次へ”ボタンをクリックします。

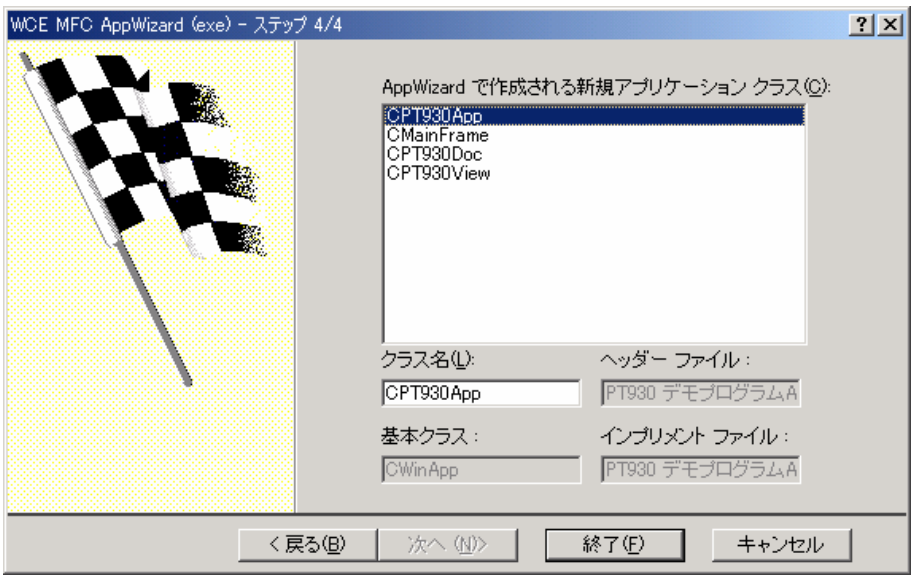


WCE MFC AppWizard(exe)を選択した場合、適当な項目をクリックあるいは選択して、“次へ”ボタンをクリックします。そして、すべての必要な MFC DLL は実行ファイルに組み込まれます。

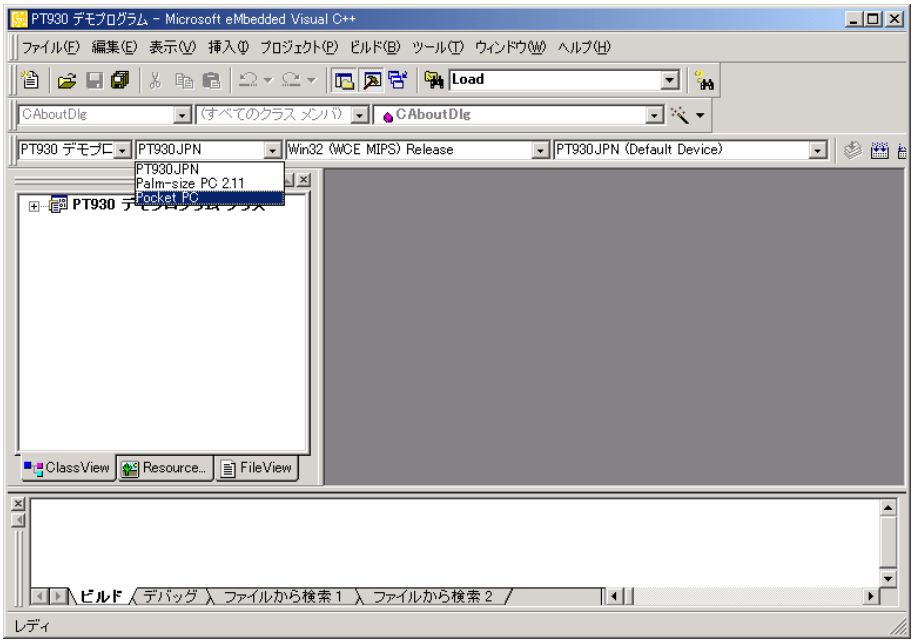




“終了”）ボタンを押し、そしてソースコードを生成するために”OK”）ボタンを押します。）



プロジェクトをビルドする前に、ターゲットプラットフォームとして"MR550 "または“ Palm-size PC 2.11 ”を選択し、そしてアクティブ構成として ”Win32 (CE MIPS) Release ” を選択して下さい。



これは皆様のアプリケーションプログラムに必要なソースコードを生成し、そして実行可能なプログラムファイルを生成するためにこのプロジェクトをビルドします。

このプログラムを MR550 に送信するために“ ActiveSync ”を使用することができます。.

6 Visual Basic によるプログラミング

6.1 インストール

VB プログラミングについては、Microsoft eMbedded Visual Basic と MR550 SDK for VB をインストールしなければなりません。

Microsoft eMbedded Visual Basic は Visual Basic 6.0 と CE toolkit for Visual Basic が組み合わされていますが、Windows CE プログラミングのみ使用可能です。

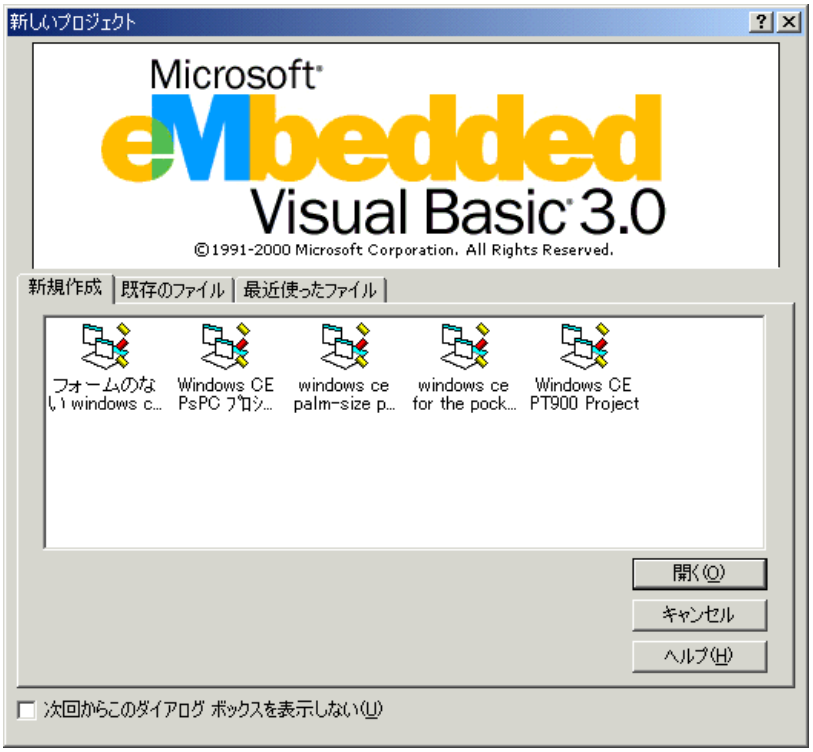
注意:本リリースに含まれている H/PC SDK を使用するために、VBCE 6.0 インストールの一部としてこれをインストールしなければなりません (完全またはカスタムのいずれか)。あるいは、これを最初にインストールして、そして VBCD 6.0 をインストールします。

より詳しい情報については、WinCE Toolkit for Visual Basic 6.0 のオンラインヘルプをご覧下さい。

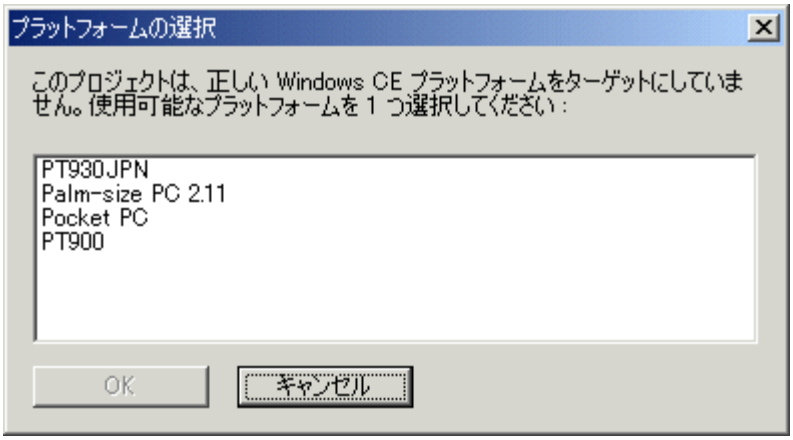
6.2 VB で Windows CE プログラムを作成する方法

Visual Basic のプログラムについては、VB プログラムを MR550 にインストールする時にインストール・プログラムと ActiveSync が必要になります。

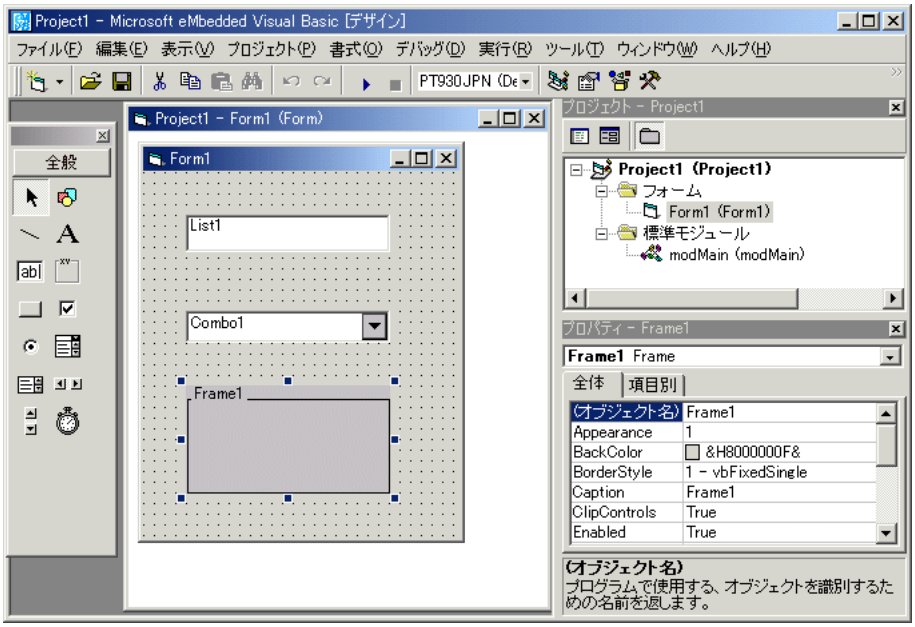
1. Visual Basic 6.0 を実行し、そして新しいプロジェクトを作成するために ”ファイル/新しいプロジェクト ” を選択します。



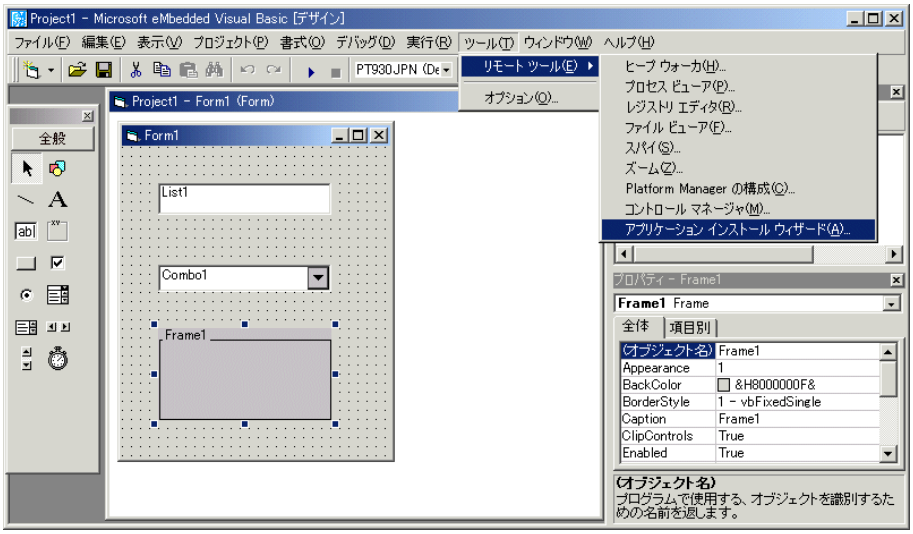
2.）“フォームのない Windows CE プロジェクト”のアイコンをダブルクリックし、ターゲット・プラットフォームとして”MR550”を選択して下さい。）



3.）そしてアプリケーションに従って皆様の Visual Basic プログラムを編集して下さい。）



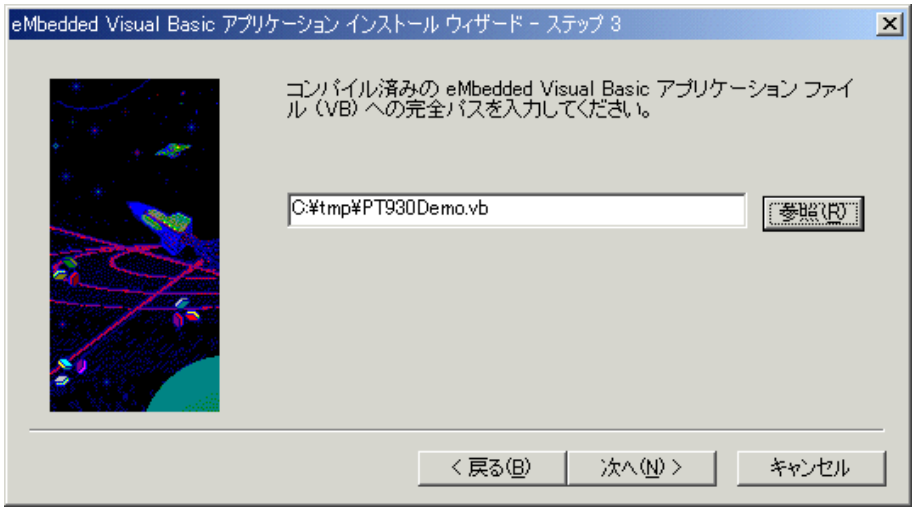
4. Visual Basic プログラムの編集が終わった後でこのプロジェクトを保存し(これは“ *.vbp ”または"*.ebp"としてディスクに保存します)、そして“ *.vb ”をディスクに保存するために“ ファイル ”から” プロジェクトの作成 ”を選択します。

5. 

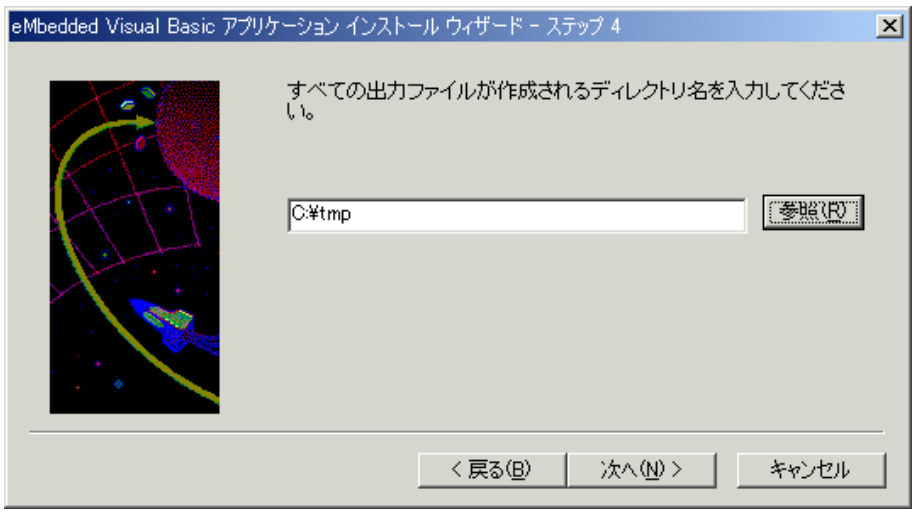
5. "ツール - リモート・ツール"から“ アプリケーション・インストール・ウイザード ”を選択し、そして” ステップ１”ダイアログボックスの“ 次へ ”ボタンをクリックして下さい。

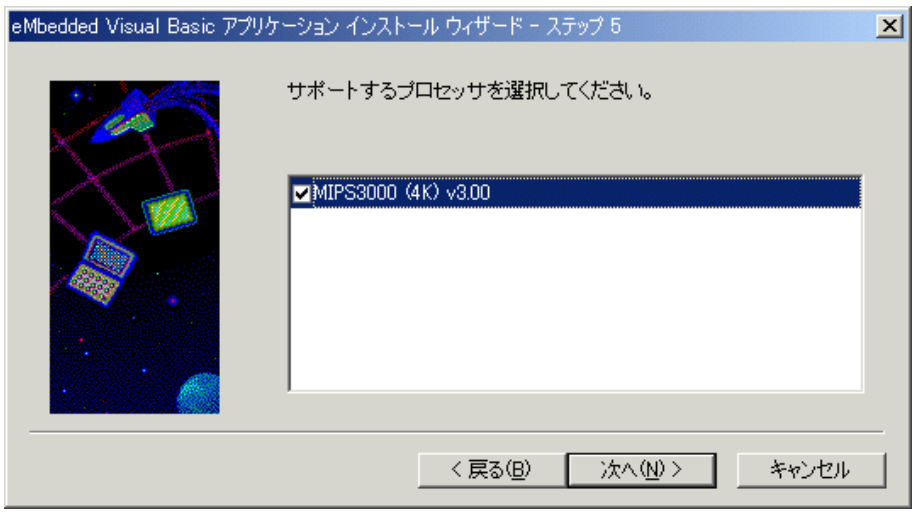
)

6.) “ ステップ) 2 ”) と) “ ステップ) 3 ”) のダイアローグボックスで、ユーザはステップ 4 で生成される) “ *.ebp ”) と) “ *.vb ”) ファイルを表示しなければなりません。)



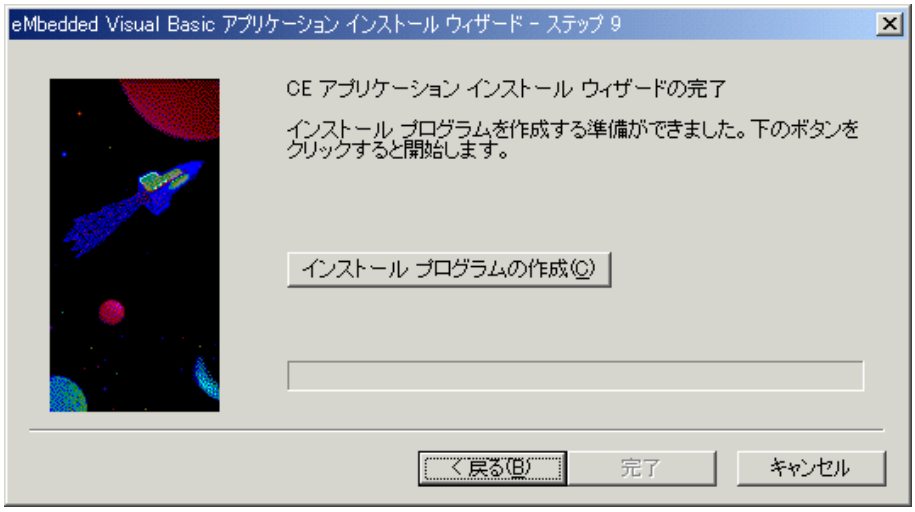
》“ステップ４”》ダイアローグボックスで出力ファィルのディレクトリを表示します。そして”ステップ５”のダイアローグボックスでプロセッサとして》“MIPS》3000”》をクリックします。》





”ステップ 6”のダイアログボックスで必要な ActiveX コントロールをクリックします。そして、”ステップ 7”のダイアログボックスで MR550 に送るすべての必要なファイルを選択します。

⟫インストールディスクの適当な“ディレクトリ”⟫と”ステップ 8”ダイアローグボックスの要求に従った他の情報を入力します。そして”ステップ 9”ダイアローグボックスで“インストール・プログラムの作成”⟫ボタンをクリックします。⟫



インストールディスクはディレクトリ”CD1”に保存されました。⟫CD-ROM⟫またはディスケットにコピーして下さい。⟫

⟫MR550 に VB プログラムをインストールしようとする PC で ActiveSync⟫が実行していることを確認して、⟫“setup.exe”を実行して下さい。⟫