Falcon® 4210

SDK/Programming Manual

PSC Inc

959 Terry Street
Eugene, Oregon 97402
Telephone: (541) 683-5700
Fax: (541) 345-7140

Should future revisions of this manual be published, you can acquire printed versions by contacting PSC Customer Administration. Electronic versions may either be downloadable from the PSC web site (**www.pscnet.com**) or provided on appropriate media. If you visit our web site and would like to make comments or suggestions about this or other PSC publications, please let us know via the "Contact PSC" page.

Disclaimer

Reasonable measures have been taken to ensure that the information included in this manual is complete and accurate. However, PSC reserves the right to change any specification at any time without prior notice.

Chapter 1. Programming the Falcon 4210

1.1 Introduction

This manual will refer to a wide variety of development kits and tools available for Window CE and will provide overviews of the various programming interfaces. The Windows CE application program can be developed on standard Microsoft integrated development environment IDEs -- Embedded Visual Tools (EVT). EVT includes Embedded Visual C++ and Embedded Visual Basic for users to develop, emulate, and debug remotely, and is tailored for Windows CE devices. The tools support Win32, COM, ActiveX, MFC, and ATL.

The Win32 API set includes some redundancy of functions – i.e. several different functions can accomplish the same task. A small amount of redundant operating system code does not usually concern developers of desktop PC software, but small code size is crucial for Windows CE developers. The release of Windows CE offered a selected subset of Win32 APIs. (For details supporting APIs, please refer to the Release Notes for Embedded Visual Tools).

Microsoft's Component Object Model, or COM, offers a standard for creating robust components that can be queried at run time. Each COM object exposes interfaces, or collections of logically related methods. The base interface *IUnknown* allows COM objects to query a particular object about its supported interfaces. The COM model is language-independent, allowing components to be updated independently without requiring any changes to its caller.

An ActiveX control is a specific type of COM object. It represents the latest in a line of specifications for extensible controls, a line that started with Visual Basic extensions (VBX) and was followed by OLE controls (OCX). An ActiveX control usually presents a user interface and exposes properties, method, and events. The control interacts with a *control container*, such as Visual Basic, through a specified set of COM interfaces. By exposing its properties, method, and events, the ActiveX control can be driven by scripts.

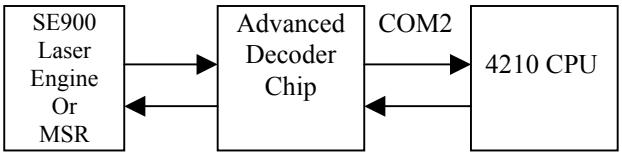
The Microsoft Foundation Classes (MFC) is a class library for developing Windows applications in C++. It exposes much of the same functionality as the Win32 API set but within a complete object-oriented application framework. The Active Template Library (ATL) is a C++ template library specially designed to create ActiveX controls and other COM components. By using template classes, ATL produces more efficient software than MFC, which uses only inheritance.

The key communication interface is Windows Sockets, or Winsock, which uses TCP/IP to communicate over a serial connection, Ethernet, RF, or infrared port. Windows CE also supports several other communication-related Win32 API sets: the Serial API, the Telephony API (TAPI), Remote Access Service (RAS), the WinINet API, which provides FTP and HTTP services, and the Windows networking API, which enumerates network resources and manages connections.

1.2 How to download data from scanner or MSR

1.2.1 Introduction

The major difference between the 4210 and a standard HPC/PalmPC is barcode input capability. The WinCE Reference Manual contains no information regarding barcode input. This section will introduce the programming structure of the barcode sub-system and the programming utility library for the 4210. Inside the 4210 there is an advanced decoding chip to control the SE900 laser engine and to handle barcode decoding. Below is system diagram for the 4210 barcode operation:



According to the above diagram, the 4210 communicates with the Decoder Chip by means of serial port COM2. Its communication parameter is fixed on 38400, N.8.1. Normally, the Decoder Chip is in sleep mode when COM2 is not activated. When COM2 is activated, the Decoder Chip will start working, and will decode the barcode "signal" from the laser engine when the trigger key is pressed. After decoding, barcode data and its symbology type will be sent directly to 4210.

Additionally, there are some pre-defined commands between the 4210 and the decoder chip for individual barcode symbology settings. Those commands will be described in section **1.2.2**.

Many programmers find it difficult to control the Decoder Chip via programming language alone, especially if they are not familiar with barcode and serial port controls. Because of this, PSC provides the following utility library and program for the user or application programmer to control the Decoder Chip:

1. Application program "BAR2KEY.EXE" is a useful application program that can read input data from the laser scanner and then directly input the data into the 4210's keyboard buffer. BAR2KEY.EXE makes barcode data input simple, and can be especially valuable to those programmers not familiar with COM port programming. The user program simply reads the barcode data from the keyboard. For barcode symbologies setting, run BARSETUP program to define all the supporting symbologies and the delimeters.
2. Utility library:
   Following are two DLL files for 4210 – "Scankey3.dll" and "Scanner3.dll".
   A. **Scanner3.dll**: This DLL provides function calls for users to get scanner data from the decoder chip. (Please refer to section **1.2.3** for a detailed function description.)
   B. **Scankey3.dll**: This DLL provides function calls for users to input scanned data into a standard keyboard buffer. (Please refer to section **1.2.4** for a detailed function description.)

1.2.2 Control commands for decoder chip

Below are all the control commands for the Decoder Chip. In the following command set, "Esc" is ASCII 0x1b, and BCC equals XOR for the command string from Esc through S1. ***Italic text*** indicates default settings.

1.2.2.1 Code39:

```
Esc,0,4,39,0,S1,BCC  where S1.Bit0=0 - disable decoding
                           S1.Bit0=1 - enable decoding
                           S1.Bit1=0 - full ASCII
                           S1.Bit1=1 - standard
                           S1.Bit3&2=0 - calculate and send check digit
                           S1.Bit3&2=1 - calculate but do not send check digit
                           S1.Bit3&2 =2 – do not calculate check digit
                           S1.Bit4=0 - send start and stop bit
                           S1.Bit4=1 – do not send start&stop bit
Esc,0,4,3a,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,3b,0,S1,BCC  where S1=0 - full ASCII
                           S1=1 - standard
Esc,0,4,3c,0,S1,BCC  where S1=0 - calculate and send check digit
                           S1=1 - calculate but do not send check digit
                           S1=2 – do not calculate check digit
Esc,0,4,3d,0,S1,BCC  where S1=0 - send start and stop bit
                           S1=1 – do not send start&stop bit
Esc,0,4,3e,0,S1,BCC  where S1=Mini. Length (0 ~ 48)
Esc,0,4,3f,0,S1,BCC  where S1=Maxi. Length (0 ~ 48)
```

1.2.2.2 I25:

```
Esc,0,4,40,0,S1,BCC  where S1.Bit0=0 - disable decoding
                           S1.Bit0=1 - enable decoding
                           S1.Bit1=0 - on fix length
                           S1.Bit1=1 - off fix length
                           S1.Bit32=0 - calculate and send Check Digit
                           S1.Bit32=1 - calculate but do not send check digit
                           S1.Bit32=2 – do not calculate check digit
                           S1.Bit54=0 - first digit suppressed
                           S1.Bit54=1 - last digit suppressed
                           S1.Bit54=2 - not suppressed
Esc,0,4,41,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,42,0,S1,BCC  where S1=0 - on fix length
                           S1=1 - off fix length
Esc,0,4,43,0,S1,BCC  where S1=0 - calculate and send check digit
                           S1=1 - calculate but do not send check digit
                           S1=2 – do not calculate check digit
Esc,0,4,44,0,S1,BCC  where S1=0 - first digit suppressed
                           S1=1 - last digit suppressed
                           S1=2 - not suppressed
Esc,0,4,45,0,S1,BCC  where S1=Mini. Length (2 ~ 64) Default=10
Esc,0,4,46,0,S1,BCC  where S1=Maxi. Length (2 ~ 64) Default=64
```

1.2.2.3 S25&Toshiba:

```
Esc,0,4,47,0,S1,BCC  where S1.Bit0=0 - disable decoding
                           S1.Bit0=1 - enable decoding
                           S1.Bit1=0 - on fix length
                           S1.Bit1=1 - off fix length
                           S1.Bit32=0 - calculate and send check digit
                           S1.Bit32=1 - calculate but do not send check digit
                           S1.Bit32=2 – do not calculate check digit
Esc,0,4,48,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,49,0,S1,BCC  where S1=0 - on fix length
                           S1=1 - off fix length
Esc,0,4,4a,0,S1,BCC  where S1=0 - calculate and send check digit
                           S1=1 - calculate but do not send check digit
                           S1=2 – do not calculate check digit
Esc,0,4,4b,0,S1,BCC  where S1=Mini. Length (1 ~ 48) Default=4
Esc,0,4,4c,0,S1,BCC  where S1=Maxi. Length (1 ~ 48) Default=48
```

1.2.2.4 Code 32:

```
Esc,0,4,4d,0,S1,BCC  where S1.bit0=0 - disable decoding
                           S1.bit0=1 - enable decoding
                           S1.bit1=0 - send leading character
                           S1.bit1=1 – do not send leading character
                           S1.bit2=0 - send tailing character
                           S1.bit2=0 – do not send tailing character
Esc,0,4,4e,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,4f,0,S1,BCC  where S1=0 - send leading character
                           S1=1 – do not send leading character
Esc,0,4,50,0,S1,BCC  where S1=0 - send tailing character
                           S1=0 – do not send tailing character
```

1.2.2.5 Telpen:

```
Esc,0,4,51,0,S1,BCC  where S1.bit0=0 - disable decoding
                           S1.bit0=1 - enable decoding
                           S1.bit1=0 - character set=standard
                           S1.bit1=1 – character=numeric
Esc,0,4,52,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,53,0,S1,BCC  where S1=0 - character set=standard
                           S1=1 –character=numeric
```

1.2.2.6 EAN128/UCC:

```
Esc,0,4,54,0,S1,BCC  where S1.bit0=0 - disable decoding
                           S1.bit0=1 - enable decoding
                           S1.bit1=0 - Disable Code ID
                           S1.bit1=1 – Enable Code ID
Esc,0,4,55,0,S1,BCC  where S1=0 - disable decoding
                           S1=1 - enable decoding
Esc,0,4,56,0,S1,BCC  where S1=0 - Disable Code ID
                           S1=1 - Enable Code ID
Esc,0,4,57,0,S1,BCC  where S1=ASCII - Separator for double labels
```

1.2.2.7 Code128:

```
Esc,0,4,58,0,S1,BCC     where S1=0 - disable decoding
                              S1=1 - enable decoding
Esc,0,4,59,0,S1,BCC     where S1=Mini. Length (0 ~ 64) Default=1
Esc,0,4,5a,0,S1,BCC     where S1=Maxi. Length (0 ~ 64) Default=64
```

1.2.2.8 MSI/Flessey:

```
Esc,0,4,5b,0,S1,BCC   where   S1.Bit0=0 - disable decoding
                              S1.Bit0=1 - enable decoding
                              S1.Bit1=0 - send check digit
                              S1.Bit1=1 - do not send check digit
                              S1.Bit32=0 - check digit double module 10
                              S1.Bit32=1 - check digit module 11 plus 10
                              S1.Bit32=2 - check digit single module 10
Esc,0,4,5c,0,S1,BCC   where   S1=0 - disable decoding
                              S1=1 - enable decoding
Esc,0,4,5d,0,S1,BCC   where   S1=0 - send check digit
                              S1=1 – do not send check digit
Esc,0,4,5e,0,S1,BCC   where   S1=0 - check digit double module 10
                              S1=1 - check digit module 11 plus 10
                              S1=2 - check digit single module 10
Esc,0,4,5f,0,S1,BCC     where S1=Mini. Length (1 ~ 64) Default=1
Esc,0,4,60,0,S1,BCC     where S1=Maxi. Length (1 ~ 64) Default=64
```

1.2.2.9 Code 93:

```
Esc,0,4,61,0,S1,BCC     where S1=0 - disable decoding
                              S1=1 - enable decoding
Esc,0,4,62,0,S1,BCC     where S1=Mini. Length (1 ~ 48) Default=1
Esc,0,4,63,0,S1,BCC     where S1=Maxi. Length (1 ~ 48) Default=48
```

1.2.2.10 Code 11:

```
Esc,0,4,64,0,S1,BCC     where S1.bit0=0 -disable decoding
                              S1.bit0=1 - enable decoding
                              S1.bit1=0 - one check digit
                              S1.bit1=1 - two check digits
                              S1.bit2=0 - send check digit
                              S1.bit2=1 - do not send
Esc,0,4,65,0,S1,BCC     where S1=0 - disable decoding
                              S1=1 - enable decoding
Esc,0,4,66,0,S1,BCC     where S1=0 - one check digit
                              S1=1 - two check digits
Esc,0,4,67,0,S1,BCC     where S1=0 - send check digit
                              S1=1 – do not send
Esc,0,4,68,0,S1,BCC     where S1=Mini. Length (1 ~ 48) Default=1
Esc,0,4,69,0,S1,BCC     where S1=Maxi. Length (1 ~ 48) Default=48
```

1.2.2.11 CodaBar:

Esc,0,4,6a,0,S1,BCC where ***S1.bit0=0 - disable decoding***
S1.bit0=1 - enable decoding
S1.bit1=0 - send start&stop character
S1.bit1=1 – do not send
S1.bit32=0 - calculate and send check digit
S1.bit32=1 - calculate but do not send check digit
S1.bit32=2 – do not calculate check digit
S1.bit4=1 - CLSI format on
S1.bit4=2 – CLSI format off

Esc,0,4,6b,0,S1,BCC where ***S1=0 - disable decoding***
S1=1 - enable decoding

Esc,0,4,6c,0,S1,BCC where S1=0 - send start&stop character
S1=1 – do not send

Esc,0,4,6d,0,S1,BCC where S1=0 - calculate and send check digit
S1=1 - calculate but do not send check digit
S1=2 – do not calculate check digit

Esc,0,4,6e,0,S1,BCC where S1=0 - CLSI format on
S1=1 –CLSI format off

Esc,0,4,6f,0,S1,BCC where S1=Mini. Length (3 ~ 48) ***Default=3***

Esc,0,4,70,0,S1,BCC where S1=Maxi. Length (3 ~ 48) ***Default=48***

1.2.2.12 Label Code:

Esc,0,4,71,0,S1,BCC where ***S1.bit0=0 - disable decoding***
S1.bit0=1 - enable decoding
S1.bit1=0 - send check digit
S1.bit1=1 – do not send check digit

Esc,0,4,72,0,S1,BCC where ***S1=0 - disable decoding***
S1=1 - enable decoding

Esc,0,4,73,0,S1,BCC where ***S1=0 - send check digit***
S1=1 – do not send check digit

1.2.2.13 UPC-A:

Esc,0,4,74,0,S1,BCC where S1.bit0=0 - disable decoding
S1.bit0=1 - enable decoding
S1.bit1=0 - send leading character
S1.bit1=1 – do not send leading character
S1.bit2=0 - send check digit
S1.bit2=1 – do not send check digit

Esc,0,4,75,0,S1,BCC where S1=0 - disable decoding
S1=1 - enable decoding

Esc,0,4,76,0,S1,BCC where ***S1=0 - send leading character***
S1=1 – do not send leading character

Esc,0,4,77,0,S1,BCC where ***S1=0 - send check digit***
S1=1 – do not send check digit

1.2.2.14 UPC-E:

Esc,0,4,78,0,S1,BCC	where	S1.bit0=0 - disable decoding
		S1.bit0=1 - enable decoding
		S1.bit1=0 - send leading character
		S1.bit1=1 – do not send leading character
		S1.bit2=0 - send check digit
		S1.bit2=1 – do not send check digit
		S1.bit3=0 - zero expansion on
		S1.bit3=1 - zero expansion off
		S1.bit4=0 - disable NSC
		S1.bit4=1 - enable NSC
Esc,0,4,79,0,S1,BCC	where	S1=0 - disable decoding
		S1=1 - enable decoding
Esc,0,4,7a,0,S1,BCC	where	***S1=0 - send leading character***
		S1=1 – do not send leading character
Esc,0,4,7b,0,S1,BCC	where	S1=0 - send check digit
		S1=1 – do not send check digit
Esc,0,4,7c,0,S1,BCC	where	S1=0 - zero expansion on
		S1=1 - zero expansion off
Esc,0,4,7d,0,S1,BCC	where	***S1=0 - disable NSC***
		S1=1 - enable NSC

1.2.2.15 EAN13:

Esc,0,4,7e,0,S1,BCC	where	S1.bit0=0 - disable decoding
		S1.bit0=1 - enable decoding
		S1.bit1=0 - send leading character
		S1.bit1=1 – do not send leading character
		S1.bit2=0 - send check digit
		S1.bit2=1 – do not send check digit
		S1.bit3=0 - Enable Bookland
		S1.bit3=1 – Disable Bookland
Esc,0,4,7f,0,S1,BCC	where	S1=0 - disable decoding
		S1=1 - enable decoding
Esc,0,4,80,0,S1,BCC	where	***S1=0 - send leading character***
		S1=1 – do not send leading character
Esc,0,4,81,0,S1,BCC	where	***S1=0 - send check digit***
		S1=1 – don’t send check digit
Esc,0,4,82,0,S1,BCC	where	S1=0 - Enable Bookland
		S1=1 – Disable Bookland

1.2.2.16 EAN8:

Esc,0,4,83,0,S1,BCC	where	S1.bit0=0 - disable decoding
		S1.bit0=1 - enable decoding
		S1.bit1=0 - send leading character
		S1.bit1=1 – do not send leading character
		S1.bit2=0 - send check digit
		S1.bit2=1 – do not send check digit
Esc,0,4,84,0,S1,BCC	where	S1=0 - disable decoding
		S1=1 - enable decoding
Esc,0,4,85,0,S1,BCC	where	***S1=0 - send leading character***
		S1=1 – do not send leading character
Esc,0,4,86,0,S1,BCC	where	***S1=0 - send check digit***
		S1=1 – do not send check digit

1.2.2.17 Supplement:

Esc,0,4,87,0,S1,BCC	where	***S1.bit0=0 Off Supplement two*** s1.bit0=1 off supplement two ***S1.bit1=0 Off Supplement five*** s1.bit1=1 off supplement five ***S1.bit2=0 transmit if present,*** S1.bit2=1 must present
Esc,0,4,88,0,S1,BCC	where	***S1=0 Off Supplement two*** s1=1 off supplement two
Esc,0,4,89,0,S1,BCC	where	***S1=0 Off Supplement five*** s1=1 off supplement five
Esc,0,4,8a,0,S1,BCC	where	***S1=0 transmit if present,*** S1=1 must present
Esc,0,4,8b,0,S1,BCC	where	S1=0 Space Separator Inserted ***S1=1 Not Inserted***

1.2.2.18 Delta Code:

Esc,0,4,8c,0,S1,BCC	where	***S1.bit0=0 mean Disable decoding*** S1.bit0=1 mean Enable decoding ***S1.bit1=0 check digit calculate*** S1.bit1=1 not calculate S1.bit2=0 check digit send, *S1.bit2=1 not send*
Esc,0,4,8d,0,S1,BCC	where	***S1=0 mean Disable decoding*** S1=1 mean Enable decoding
Esc,0,4,8e,0,S1,BCC	where	***S1=0 check digit calculate*** S1=1 not calculate
Esc,0,4,8f,0,S1,BCC	where	S1=0 check digit send, *S1=1 not send*

1.2.2.19 MSR Track selection:

Esc,0,4,33,0,S1,BCC	where	***S1=0 All Track*** S1=1 Track 1 and Track 2 S1=2 Track 1 and Track 3 S1=3 Track 2 and Track 3 S1=4 Track 1 only S1=5 Track 2 only S1=6 Track 3 only

1.2.3 Function call list for “Scanner3.dll”

The Scanner3.dll file can be located in the Technical Binder CD in the \Programming\scanner folder. In this folder can also be found two additional files:

"Scanner3.lib" Used for VC programming
"Scanner3.h" Used for VC programming

1.2.3.1 Enable Decoder

Function Description: This function will open the COM2 port, create a thread to get any barcode input from the Decoder Chip, and then store input data in the system buffer. The application can use the function call **PT_GetBarcode()** to get input data from the system buffer.

Function call: INT PT_EnableBarcode(VOID);

Return code: =1 Create new thread fail
=2 Cannot re-enable
=3 Cannot open COM2
=4 Upload parameter from Hamster fail
=0 OK

1.2.3.2 Disable Decoder

Function Description: This function will close the COM2 port and then remove the thread which is created by **PT_EnableBarcode()**

Function call: VOID PT_DisableBarcode(VOID);

1.2.3.3 Check barcode input

Function Description: This function is used to check whether there is available barcode data in the system buffer that was successfully decoded by the decoder chip.

Function call: BOOL PT_CheckBarcode(VOID);

Return code: =TRUE There is input data in the system buffer.
=FALSE There is no data in the system buffer.

1.2.3.4 Read barcode data

Function Description: Get input barcode data and its type from system buffer.

Function call: BOOL PT_GetBarcode(TCHAR *szBarcodeBuffer,TCHAR *cType);

Parameter: (output)

szBarcodeBuffer : string buffer for storing input data

cType : Type of Input data
=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

Return code: TRUE = There is barcode input
FALSE = No Barcode Input

1.2.3.5 Check MSR input

Function Description: This function is used to check whether there is available MSR data in the system buffer that was successfully decoded by the decoder chip.
Function call: BOOL PT_CheckMSR**()**
Return code: TRUE = There is MSR input
FALSE = No MSR Input

1.2.3.6 Read MSR data

Function Description: Get Magnetic stripe reader (MSR) input.
Function call: BOOL PT_GetMSR**(**TCHAR *szTrack1,TCHAR *szTrack2,TCHAR *szTrack3, TCHAR *cType**)**
Parameter: (output)
szTrack1: string buffer for storing input data from track 1
szTrack2: string buffer for storing input data from track 2
szTrack3: string buffer for storing input data from track 3
cType : indictor for MSR input data
1 Track1 only
2 Track2 only
3 Track3 only
4 Track1 and Track2
5 Track2 and Track3
6 Track1, Track2 and Track3
Return code: TRUE = There is MSR input
FALSE = No MSR Input

1.2.3.7 Set Barcode symbology into decoder chip

Function description: This function is used to set individual barcode symbology.
Function call: BOOL PT_SetBarcodePara(unsigned char type, unsigned char status);
Return : TRUE --- Setting OK.
FALSE -- Setting Failure
Parameter: (input)

SYMBOLOGY	Type	Status	Action
Code39 all setting	0x39	Bit0 = 0	Disable
		Bit0 = 1	***Enable***
		Bit1 = 0	***Full Code39***
		Bit1 = 1	Standard Code39
		Bit3&2 = 0	Calculate and Send check digital
		Bit3&2 = 1	Calculate and Not Send check digital
		Bit3&2 = 2	***Not Calculate check digital***
		Bit4 = 0	Send Start and Stop bit
		Bit4 = 1	***Not Send Start and Stop bit***
Code39	0x3A	0	Disable
		1	***Enable***
Code39 Full ASCII	0x3B	***0***	***Enable***
		1	Disable
Code39 Check digital	0x3C	0	Calculate and Send check digital
		1	Calculate and Not Send check digital
		2	***Not Calculate check digital***
Code39 SS	0x3D	0	Send Start and Stop bit
		1	***Not Send Start and Stop bit***
Code39 mini. length	0x3E	***0*** ~ 48	**Default = 0**
Code39 maxi. length	0x3F	0 ~ ***48***	**Default = 48**

SYMBOLOGY	Type	Status	Action
I25	0x40	Bit0 = 0	Disable
		Bit0 = 1	***Enable***
		Bit1 = 0	Enable Fix Length
		Bit1 = 1	***Disable Fix Length***
		Bit2&3 = 0	Calculate and Send check digital
		Bit2&3 = 1	Calculate and Not Send check digital
		Bit2&3 = 2	***Not Calculate check digital***
		Bit4&5 = 0	First Digit suppressed
		Bit4&5 = 1	Last Digit suppressed
		Bit4&5 = 2	***Not suppressed***
I25	0x41	0	Disable
		1	***Enable***
I25 Fix Length	0x42	0	Enable Fix Length
		1	***Disable Fix Length***
I25 CD	0x43	0	Calculate and Send check digital
		1	Calculate and Not Send check digital
		2	***Not Calculate check digital***
I25 SS	0x44	0	First Digit suppressed
		1	Last Digit suppressed
		2	***Not suppressed***
I25 mini. length	0x45	2 ~ 64	**Default = 10**
I25 maxi. length	0x46	2 ~ 64	**Default = 64**
S25&Toshiba	0x47	***Bit0 = 0***	***Disable***
		Bit0 = 1	Enable
		Bit1 = 0	***Enable Fix Length***
		Bit1 = 1	Disable Fix Length
		Bit2&3 = 0	Calculate and Send Check Digit
		Bit2&3 = 1	Calculate and Not Send Check Digit
		Bit2&3 = 2	***Not Calculate Check Digit***
S25&Toshiba	0x48	***0***	***Disable***
		1	Enable
S25&Toshiba fix length	0x49	***0***	***Enable Fix Length***
		1	Disable Fix Length
S25&Toshiba CD	0x4A	0	Calculate and Send Check Digit
		1	Calculate and Not Send Check Digit
		2	***Not Calculate Check Digit***
S25&Toshiba mini length	0x4B	1 ~ 48	Default = 4
S25&Toshiba max length	0x4C	1 ~ 48	Default = 48

SYMBOLOGY	Type	Status	Action
Code32	0x4d	***Bit0 = 0***	***Disable***
		Bit0 = 1	Enable
		Bit1 = 0	***Send Leading Character***
		Bit1 = 1	Not Send Leading Character
		Bit2 = 0	***Send Tailing Character***
		Bit2 = 1	Not Send Tailing Character
Code32	0x4e	***0***	***Disable***
		1	Enable
Code32 Send leading char	0x4f	***0***	***Send Leading Character***
		1	Not Send Leading Character
Code32 Send tailing char	0x50	***0***	***Send Tailing Character***
		1	Not Send Tailing Character
Telpen	0x51	***Bit0=0***	***Disable***
		Bit0=1	Enable
		Bit1=0	***Character set = standard***
		Bit1=1	Character set = numeric
Telpen	0x52	***0***	***Disable***
		1	Enable
Telpen character set	0x53	***0***	***Standard***
		1	Numeric
EAN 128/UCC	0x54	0	Disable
		1	***Enable***
EAN 128/UCC	0x55	0	Disable
		1	***Disable***
EAN 128/UCC Code ID	0x56	***0***	***Disable***
		1	Disable
EAN 128/UCC separator for double labels	0x57	ASCII	ASCII character
Code 128	0x58	0	Disable
		1	***Enable***
Code 128 mini length	0x59	0 ~ 64	Default = 1
Code 128 maxi length	0x5a	0 ~ 64	Default = 64

SYMBOLOGY	Type	Status	Action
MSI/Plessey	0x5B	***bit0=0***	***Disable***
		bit0=1	Enable
		bit1=0	Send Check Digital
		bit1=1	***Not Send Check Digital***
		bit3&2=0	***Check Digital Double Module 10***
		bit3&2=1	Check Digital Module 11 plus 10
		bit3&2=2	Check Digital Module 10
MSI/Plessey	0x5C	***0***	***Disable***
		1	Enable
MSI/Plessey CD	0x5D	0	Send Check Digits
		1	***Not Send Check Digits***
MSI/Plessey CDMODE	0x5E	***0***	***Check Digital Double Module 10***
		1	Check Digital Module 11 plus 10
		2	Check Digital Module 10
MSI/Plessey mini length	0x5F	***1 ~ 64***	***Default = 1***
MSI/Plessey maxi	0x60	***1 ~ 64***	***Default = 64***
Code93	0x61	0	Disable
		1	***Enable***
Code93 mini length	0x62	1 ~ 48	Default = 1
Code93 maxi length	0x63	1 ~ 48	Default = 48
Code11	0x64	***bit0=0***	***Disable***
		bit0=1	Enable
		bit1=0	One Check Digits
		bit1=1	***Two Check Digits***
		bit2=0	Send Check Digits
		bit2=1	***Not Send Check Digits***
Code11	0x65	***0***	***Disable***
		1	Enable
Code11 check digit number	0x66	0	One Check Digits
		1	***Two Check Digits***
Code11 CD	0x67	0	Send Check Digits
		1	***Not Send Check Digits***
Code11 mini length	0x68	1 ~ 48	Default = 1
Code11 maxi length	0x69	1 ~ 48	Default = 48

SYMBOLOGY	Type	Status	Action
CodaBar	0x6a	***Bit0 = 0***	***Disable***
		Bit0 = 1	Enable
		Bit1 = 0	Send Start and Stop character
		Bit1 = 1	***Not Send Start and Stop character***
		Bit2&3 = 0	Calculate and Send Check Digit
		Bit2&3 = 1	Calculate and Not Send Check Digit
		Bit2&3 = 2	***Not Calculate Check Digit***
		Bit4 = 0	CLSI format on
		Bit4 = 1	***CLSI format off***
CodaBar	0x6b	***0***	***Disable***
		1	Enable
CodaBar SS	0x6c	0	Send Start and Stop character
		1	***Not Send Start and Stop character***
CodaBar CD	0x6d	0	Calculate and Send Check Digit
		1	Calculate and Not Send Check Digit
		2	***Not Calculate Check Digit***
Codabar CLSI format	0x6e	0	On
		1	***Off***
Codabar mini length	0x6f	3 ~ 48	Default = 3
Codabar maxi length	0x70	3 ~ 48	Default = 48
Label Code	0x71	***Bit0 = 0***	***Disable***
		Bit0 = 1	Enable
		Bit1 = 0	***Send Check Digit***
		Bit1 = 1	Not Send Check Digit
Label Code	0x72	***0***	***Disable***
		1	Enable
Label Code CD	0x73	***0***	***Send Check Digit***
		1	Not Send Check Digit
UPC-A	0x74	bit0=0	Disable
		bit0=1	***Enable***
		bit1=0	***Send Leading digit***
		bit1=1	Not Send Leading digit
		bit2=0	***Send Check digit***
		bit2=1	Not Send Check digit
UPC-A	0x75	0	Disable
		1	***Enable***
UPC-A LD	0x76	***0***	***Send Leading digit***
		1	Not Send Leading digit
UPC-A CD	0x77	***0***	***Send Check digit***
		1	Not Send Check digit

SYMBOLOGY	Type	Status	Action
UPC-E	0x78	bit0=0	Disable
		bit0=1	***Enable***
		bit1=0	***Send Leading digit***
		bit1=1	Not Send Leading digit
		bit2=0	Send Check digit
		bit2=1	***Not Send Check digit***
		bit3=0	Zero expansion On
		bit3=1	***Zero expansion Off***
		bit4=0	***NSC Enable***
		bit4=1	NSC Disable
UPC-E	0x79	0	Disable
		1	***Enable***
UPC-E LD	0x7a	***0***	***Send Leading digit***
		1	Not Send Leading digit
UPC-E CD	0x7b	0	Send Check digit
		1	***Not Send Check digit***
UPC-E Expansion	0x7c	0	Zero expansion On
		1	***Zero expansion Off***
UPC-E NSC	0x7d	***0***	***Disable***
		1	Enable
EAN13	0x7e	bit0=0	Disable
		bit0=1	***Enable***
		bit1=0	***Send Leading digit***
		bit1=1	Not Send Leading digit
		bit2=0	***Send Check digit***
		bit2=1	Not Send Check digit
		Bit3=0	Enable BookLand
		Bit3=1	***Disable BookLand***
EAN13	0x7F	0	Disable
		1	***Enable***
EAN13 LD	0x80	***0***	***Send Leading digit***
		1	Not Send Leading digit
EAN13 Check digit	0x81	***0***	***Send check digit***
		1	Not send
EAN13 bookland	0x82	***0***	***Enable***
		1	disable

SYMBOLOGY	Type	Status	Action
EAN8	0x83	Bit0=0	Disable
		Bit0=1	***Enable***
		Bit1=0	***Send Leading digit***
		Bit1=1	Not Send Leading digit
		Bit2=0	***Send Check digit***
		Bit2=1	Not Send Check digit
EAN8	0x84	0	Disable
		1	***Enable***
EAN8 LD	0x85	***0***	***Send Leading digit***
		1	Not Send Leading digit
EAN8 CD	0x86	***0***	***Send Check digit***
		1	Not Send Check digit
Supplement Two/Five	0x87	***bit0=0***	***Disable Supplement Two***
		bit0=1	Enable Supplement Two
		Bit1=0	***Disable Supplement Five***
		Bit1=1	Enable Supplement Five
		Bit2=0	***Transmit if Present***
		Bit2=1	Must Present
Supplement Two	0x88	***Bit0=0***	***Disable***
		Bit0=1	Enable
Supplement Five	0x89	***Bit0=0***	***Disable***
		Bit0=1	Enable
Supplement MH	0x8A	***Bit0=0***	***Transmit if Present***
		Bit0=1	Must Present
Supplement SSI	0x8B	Bit0=0	Space Separator Inserted
		Bit0=1	***No Insert***
Delta Code	0x8C	***Bit0=0***	***Disable***
		Bit0=1	Enable
		Bit1=0	***Calculate Check Digit***
		Bit1=1	Not Calculate Check Digit
		Bit2=0	Send Check Digit
		Bit2=1	***Not Send Check Digit***
Delta Code	0x8D	***Bit0=0***	***Disable***
		Bit0=1	Enable
Delta Code CDC	0x8E	***Bit0=0***	***Calculate Check Digit***
		Bit0=1	Not Calculate Check Digit
Delta Code CDD	0x8F	Bit0=0	Send Check Digit
		Bit0=1	***Not Send Check Digit***
MSR	6	***0***	***All Tracks***
		1	Track 1 and Track 2
		2	Track 1 and Track 3
		3	Track 2 and Track 3
		4	Track 1 only
		5	Track 3 only
		6	Track 3 only

1.2.3.8 Get Barcode symbology setting from decoder chip

Function description: This function is used to get individual barcode symbology from the decoder chip.
Function call: BOOL PT_GetBarcodePara(unsigned char type, int *status);
Input/Output: Please refer to table on section **1.2.3.7**
Return : TRUE --- OK.
FALSE – Failure

1.2.3.9 Get DLL version no

Function description: This function is used to get DLL version no.
Function call: INT PT_DllVersion(void);
Return : Integer

1.2.3.10 Reset all symbologies to default

Function Description: This function call will reset decoder chip's symbologies setting to system default value

Function call for VC: int PT_ SetToDefault (VOID)
Function call for VB: PT_ SetToDefault

1.2.4 Function call list for "Scankey3.dll"

The Scankey3.dll file is located in the Technical Binder CD in folder \Programming\scankey. Also in this folder are 3 extra files:

"Scankey3.lib" Used for VC programming
"Scankey3.h" Used for VC programming
"Scankey3.bas" Used for VB programming

1.2.4.1 Enable Decoder

Function Description: This function will open the COM2 port, create a thread to get any barcode input from the Decoder Chip, and then send scanner data to the keyboard buffer. The user application can get input data just like standard keyboard input.

Function call for VC: int PT_EnableBarToKey(VOID)
Function call for VB: PT_EnableBarToKey() as Long
Return code: =1 Create new thread fail
=2 Cannot re-enable
=3 Cannot open COM2
=4 Upload parameter from Hamster fail
=0 OK

1.2.4.2 Disable Decoder

Function Description: This function will close the COM2 port and then remove the thread that was created by **PT_EnableBarToKey()**
Function call for VC: VOID PT_DisableBarToKey (VOID)
Function call for VB: PT_DisableBarToKey ()

1.2.4.3 Set Barcode symbology into decoder chip

Function description: This function is used to set individual barcode symbology.
Function call for VC: BOOL PT_SetBarToKeyPara(unsigned char ,unsigned char)
Function call for VB: PT_SetBarToKeyPara (ByVal strType As String, ByVal strStatus As String) As Boolean
Return: TRUE --- Setting OK.
FALSE -- Setting Failure
Parameter: (input) the same with section 1.2.3.7

1.2.4.4 Get Barcode symbology setting from decoder chip

Function description: This function is used to get individual barcode symbology from the decoder chip.
Function call VC: BOOL PT_GetBarToKeyPara(unsigned char cType,int* iStatus)
Function call VB: No function call available for VB
Input/Output: Please refer to table on section **1.2.3.7**
Return: TRUE --- OK.
FALSE – Failure

1.2.4.5 Get DLL version no

Function description: This function is used to get the DLL version number.
Function call for VC: INT PT_Version (void);
Function call for VB: PT_Version As Long
Return: Integer

1.2.4.6 Set Pre-amble string

Function description: This function forces "Scankey3.dll" to send a Pre-amble string to the keyboard buffer before scanned data. The default Pre-amble string is NULL and its maximum length is 48 characters.
Function call for VC: void PT_SetPreamble(TCHAR *sPreambleStr)
Function call for VB: void PT_SetPreamble(ByVal strPreamble As String)
Input **:** String buffer for Pre-amble string
Return: None

1.2.4.7 Get Pre-amble string

Function description: Get the current Pre-amble string.
Function call for VC: void PT_GetPreamble(TCHAR * sPreambleStr)
Function call VB: No function call available for VB
Output **:** String buffer for Pre-amble string
Return: None

1.2.4.8 Set Post-amble string

Function description: This function forces "Scankey3.dll" to send a post-amble string to the keyboard buffer after scanned data. The default Post-amble string is NULL and its maximum length is 48 characters.
Function call for VC: void PT_SetPostamble(TCHAR *sPostambleStr)
Function call for VB: void PT_SetPostamble(ByVal strPostamble As String)
Input **:** String buffer for the post-amble string
Return: None

1.2.4.9 Get Post-amble string

Function description: Get the current Post-amble string.
Function call: void PT_GetPostamble (TCHAR * sPostambleStr)
Function call VB: No function call available for VB
Output **:** String buffer for the post-amble string
Return: None

1.2.4.10 Set delimiter string

Function description: This function forces "Scankey3.dll" to append a delimiter string to the keyboard buffer behind the post-amble. The default delimiter string is NULL and its maximum length is 10 characters.
Function call for VC: void PT_SetDelimiter(TCHAR *sDelimiterStr)
Function call for VB: PT_SetPostamble (ByVal strPostamble As String)
Input **:** String buffer for delimiter string
Return: None

1.2.4.11 Get delimiter string

Function description: Get the current delimiter string.
Function call for VC: void WINAPI PT_GetDelimiter(TCHAR *sDelimiterStr)
Function call for VB: No function call available for VB
Output **:** String buffer for delimiter string
Return: None

1.2.4.12 Disable laser trigger key

Function Description: This function disables the laser beam trigger key and leaves the COM2 port open. This function call is useful when some fields allow only keyboard input.
Function call for VC: int PT_StopScan (VOID)
Function call for VB: PT_StopScan

1.2.4.13 Enable laser trigger key

Function Description: This function activates the laser beam trigger key and leaves the COM2 port open. This function call is useful when scanner input is now allowed.

Function call for VC: int PT_StartScan (VOID)
Function call for VB: PT_StartScan

1.2.4.14 Reset all symbologies to default

Function Description: This function call will reset decoder chip's symbologies setting to system default value.

Function call for VC: int PT_ SetToDefault (VOID)
Function call for VB: PT_ SetToDefault

1.3 Programming by Visual C/C++ with SDK or MFC

1.3.3 System Requirements:

HW	**CPU**	A Pentium 90-MHz or higher processor is recommended.
	Main memory	32 MB for Windows NT Workstation 4.0 (48 MB recommended); 24 MB for Windows 95/98 (48 MB recommended).
	CDROM drive	Compatible with multimedia desktop computer specification.
	Display	VGA or higher resolution monitor required. A Super VGA monitor is recommended.
	Mouse	Mouse device or compatible pointing device.
SW	**OS**	Microsoft® Windows NT® Workstation 4.0, Microsoft® Windows® 95, or Microsoft® Windows® 98.
	Develop tools	Microsoft eMbedded Visual Tools and 4210 SDK for VC/VB

Note: Microsoft Windows NT Workstation 4.0 is the recommended debug host for your development environment. The toolkit can be installed on Windows 95/98, and you can build your application from there, *however, emulation will not work on Windows 95/98. Instead we recommend you use Windows NT as your host machine*.

1.3.4 Installation

Use Microsoft eMbedded Visual Tools (EVT) and 4210 SDK for VC++ to develop 4210 application programs.

Microsoft® eMbedded Visual C++® 3.0 enables you to develop Windows CE-based applications using an integrated development environment (IDE) similar to that used in developing desktop Visual C++ applications. This IDE, however, contains Windows CE-specific versions of many of the standard development tools you use to create, test, and refine your applications. It also includes a variety of tools that help you develop new software uniquely appropriate for Windows CE platforms and devices.

Custom-built for developing Windows CE applications, the eMbedded Visual C++ IDE is easy to learn and will be familiar to programmers who have experience with other members of the Visual C++ family. Applications can be created with eMbedded Visual C++ to run on the Handheld PC Pro (H/PC Pro), Palm-size PC 1.2, Pocket PC and 4210 platforms. You can also use eMbedded Visual C++ to create applications that run on custom Windows CE-based platforms, or within a desktop emulator that simulates a Windows-CE based platform.

After the installation of Microsoft eMbedded Visual Tools is complete, a dialog box will prompt you to install the SDKs for the H/PC and the Palm-size PC. Choose “Yes” to install these SDKs at this time.

This will launch from Microsoft eMbedded Visual Tools setup program, which in turn launches the setup program for the Windows CE SDKs for the H/PC, Palm-size PC and Pocket PC. All three setup programs must be completed to ensure a properly working installation. You must install these SDKs if you want to use MFC for Windows CE.

Additionally, PSC also provides our own SDK for the 4210 platform. Run the installation program in folder \SDK\VC in the Technical Binder CD. The PSC SDK includes the Palm-Size SDK and a special scanner library, so its not necessary to copy "scanner3.lib", "scanner3.h", "scankey3.lib" and "scankey3.h" into your development folder. Find them in the standard \lib and \include folders.

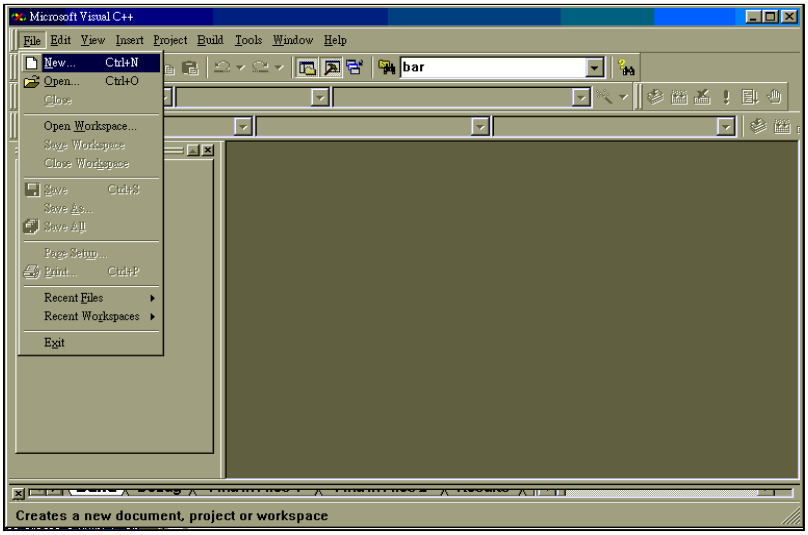
Note: You must install at least one Windows CE SDK in order for the Windows CE Toolkit for Visual C++ to function correctly.

For more detailed information, please refer to release document of eMbedded Visual C++.

1.3.5 How to create a WinCE application

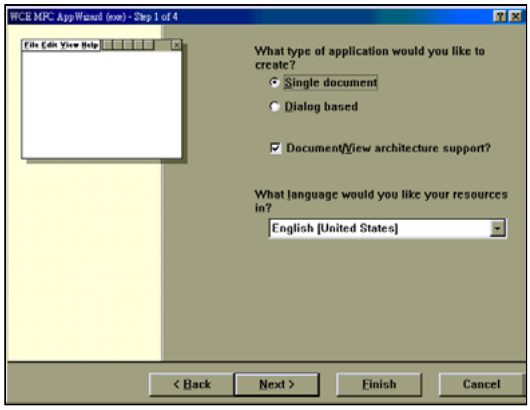
Follow the steps below to create an application:

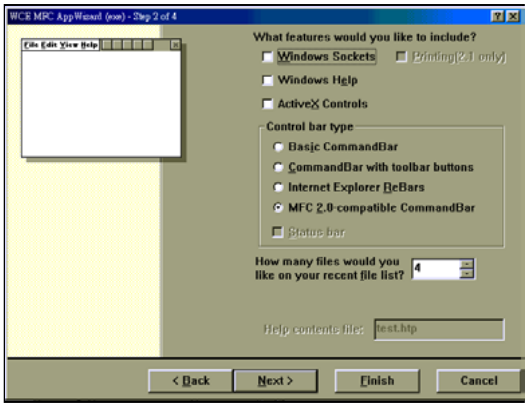
1. Execute eMbedded Visual C++.
2. Select “File / New” from “Main Menu”.



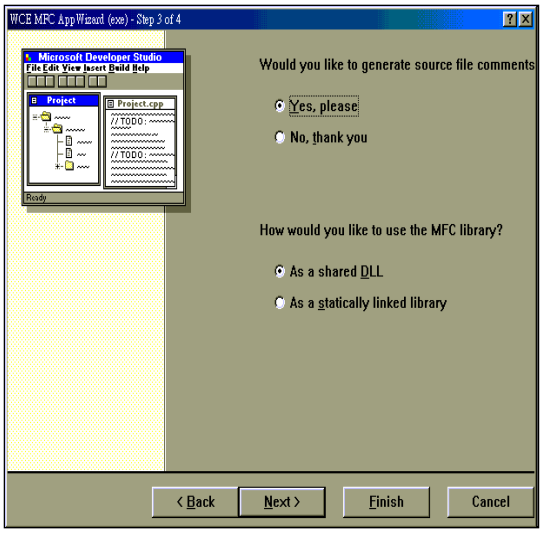
3. A dialog box will pop up for AP type selecting. For WinCE applications there are 7 project types to choose from. The user can create an SDK program by selecting “WCE application”, or create an MFC program by selecting “WCE MFC AppWizard (exe)”, or select any of the other five options. Input an appropriate name for this project in “Project Name”, and then select “Win32 (WCE MIPS)” under “Platforms”. Now click the “OK” button for the next step.

4. Click or select proper items and then click the “Next” button.

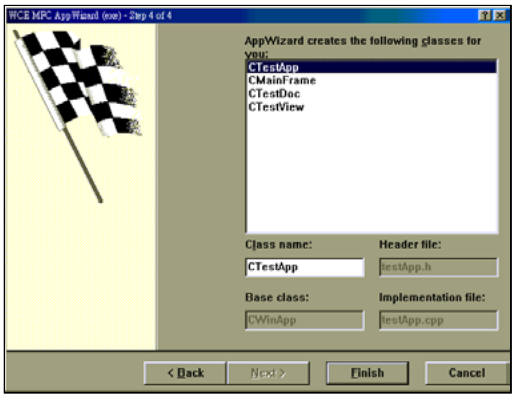




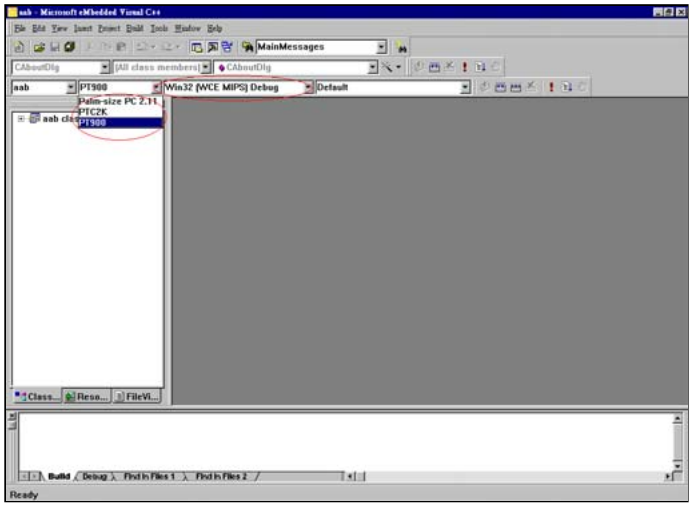
5. Click or select the appropriate items and then click the “Next” button. Because there is no MFC support on 4210 with 8 MB FlashRom, you should click item “As a statically linked library” instead of “As shared DLL”. Then all the necessary MFC DLL will be bound into the execution file. This will result in an increase in the size of the application program.



6. Click the “Finish button” and then the “OK” button to generate source code.



7. Before building the project, please select "4210" or “Palm-size PC” as the target platform and “Win32 (CE MIPS) Release” as the active configuration.



8. This will generate the necessary source code for your application program, and then build this project to generate an executable program file**.**

9. Use “ActiveSync” to send this program file to 4210. For detailed communication procedures, please refer to the 4210 Communication Manual.

1.4 Programming by Visual Basic

1.4.1 Installation

For VB programming, install Microsoft eMbedded Visual Basic and 4210 SDK for VB

Microsoft eMbedded Visual Basic is combined from Visual Basic 6.0 and CE toolkit for Visual Basic, but it is only available for WinCE programming. The user can still develop their Windows application program for Windows 95/98/NT when they choose the item 1 combination.

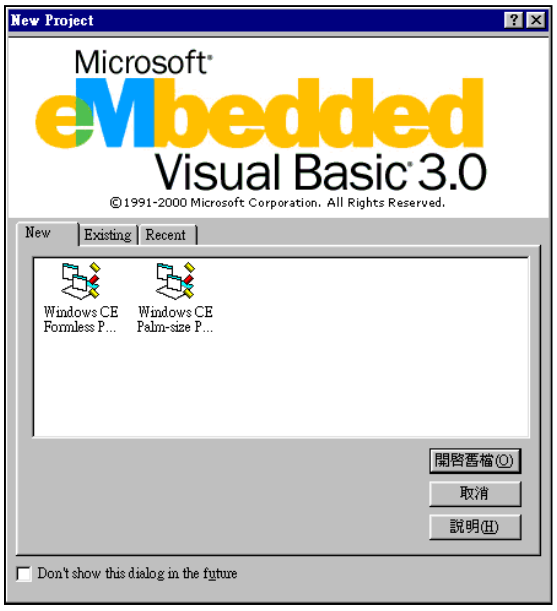
NOTE: In order to use the H/PC SDK included with this release, you must install it as part of the VBCE 6.0 install (either Complete or Custom). Or, you can install it first and then install VBCE 6.0.

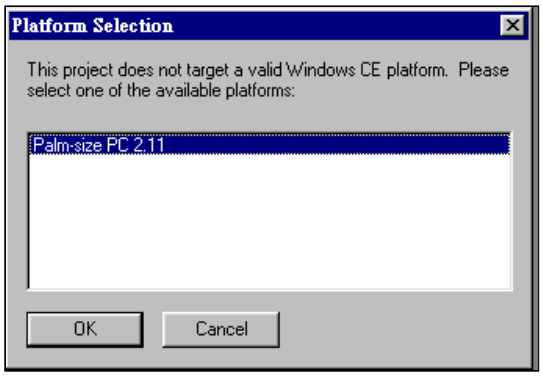
For more detailed information, please refer to the release document in WinCE Toolkit for Visual Basic 6.0

1.4.2 How to create a Visual Basic program

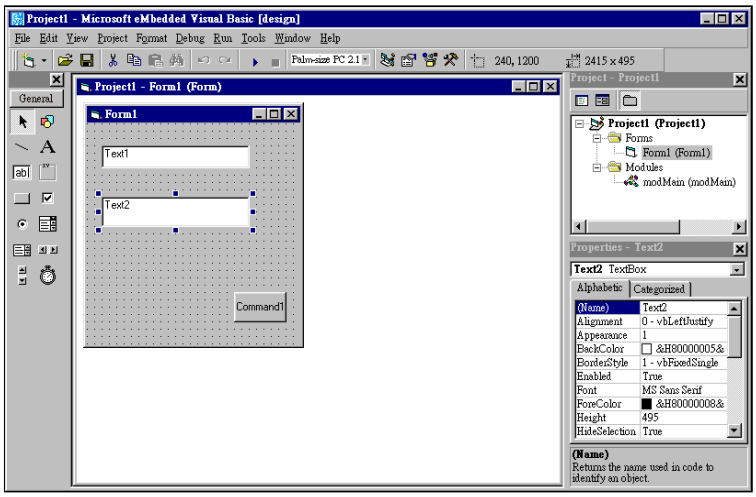
Visual Basic programming will require making an installation program and disk. CE Service is required when installing a VB program into the 4210. CESync cannot be used to install the program into the 4210. Besides these requirements, please follow the steps below to create an application

1. Execute Visual Basic 6.0, then Select "File / New" from the main menu to create a new project.
2. Double click the icon "Windows CE Formless Project", and then select "Palm-size PC 2.10" as the target platform.



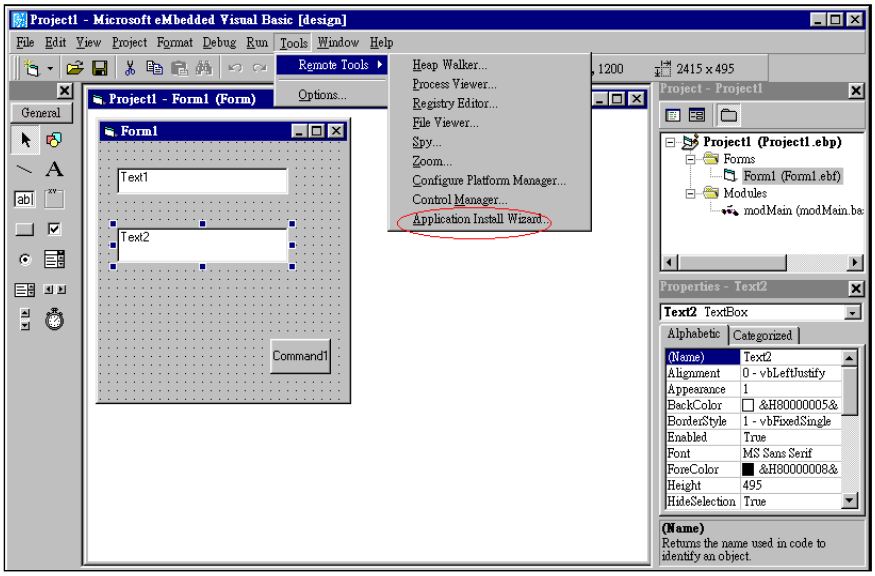


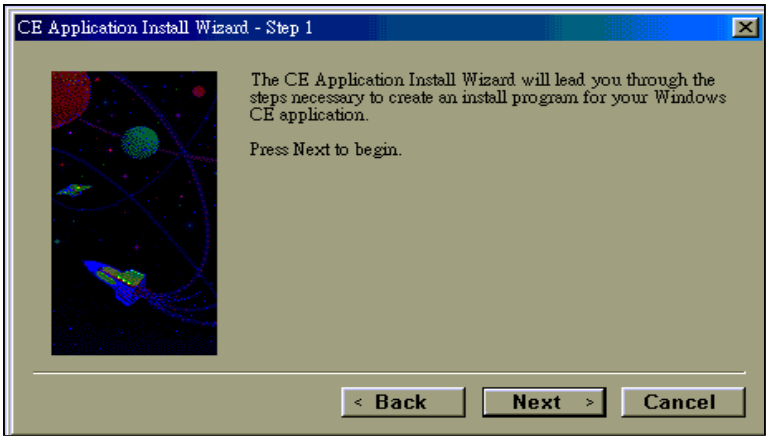
3. Then edit your Visual Basic program according your application.



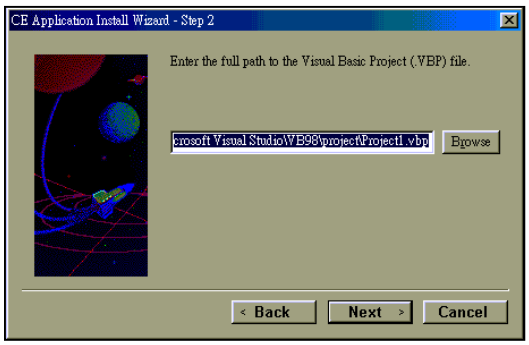
4. Save this project after finishing the Visual Basic program (this action will save “*.vbp” or "*.ebp" to disk), and then select “Make Project” from “File” items to save “*.vb” to disk.

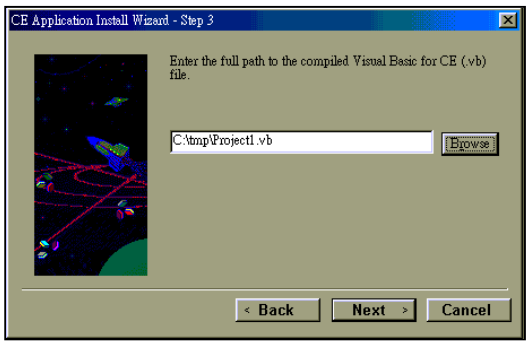
5. Select “Application Install Wizard” from “Windows CE” or "Tools\Remote Tools" item, and then click the “Next” button in the “Step1” dialog box.



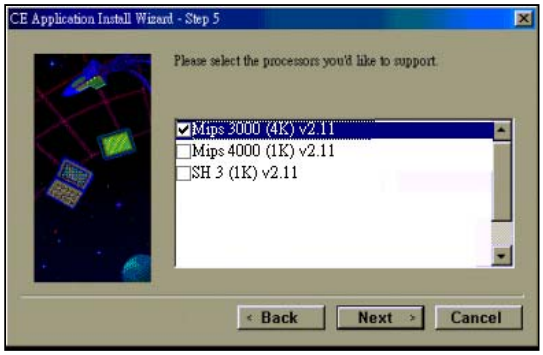


6. In the “Step 2” and “Step 3” dialog boxes, browse the “*.vbp” and “*.vb” files which will be generated in step 4.

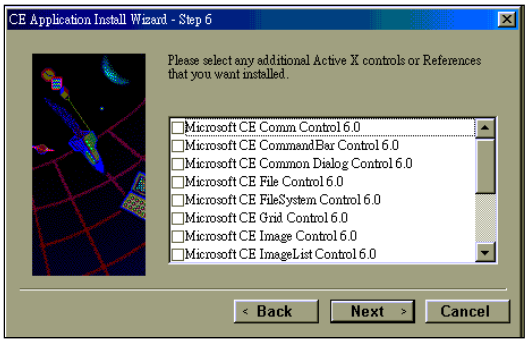


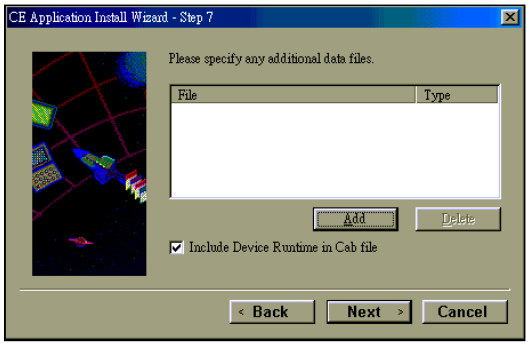


7. Browse the directory for output files in the “Step 4” dialog box. Then click “MIPS 3000” as processor in the “Step 5” dialog box.

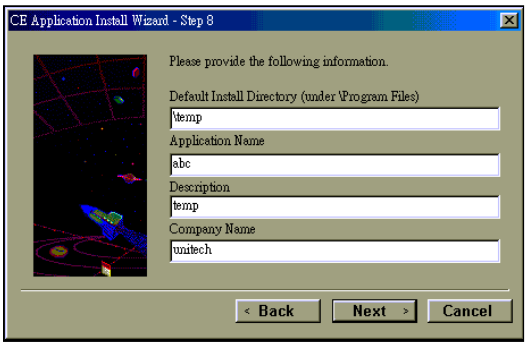


8. Click the necessary ActiveX control in the ”step 6” dialog box. Then select all the necessary files that will be sent to 4210 in the “Step 7” dialog box.





9. Input the proper “directory” in which the installation disk and other information will be stored according to the prompt in the “Step 8” dialog box. Then click the “Create Install” button in the “Step 9” dialog box.



10. The installation disk is NOW stored on directory “CD1” (which is specified in the “Step 8” dialog box). Please copy to a CD-ROM or diskette.

11. Please make sure ActiveSync is running on the PC on which you intend to install your VB program for the 4210, and then execute “setup.exe”.

Filename:	PSC 4210 Prog Man.doc
Directory:	C:\Documents and Settings\jamaitla\Local Settings\Temporary Internet Files\OLK29
Template:	C:\Documents and Settings\jamaitla\Application Data\Microsoft\Templates\Normal.dot
Title:	
Subject:	
Author:	ANTHONY HSU
Keywords:	
Comments:	
Creation Date:	4/3/2002 10:21 AM
Change Number:	7
Last Saved On:	4/5/2002 3:41 PM
Last Saved By:	Jan Hellsund
Total Editing Time:	241 Minutes
Last Printed On:	4/9/2002 12:23 PM
As of Last Complete Printing	
Number of Pages:	30
Number of Words:	5,885 (approx.)
Number of Characters:	33,549 (approx.)

Asia Pacific
PSC Hong Kong
Hong Kong
Telephone: [852]-2-584-6210
Fax: [852]-2-521-0291

Australia
PSC Asia Pacific Pty Ltd.
North Ryde, Australia
Telephone: [61] 0 (2) 9878 8999
Fax: [61] 0 (2) 9878 8688

France
PSC S.A.R.L.
LES ULIS Cedex, France
Telephone: [33].01.64.86.71.00
Fax: [33].01.64 46.72.44

Germany
PSC GmbH
Darmstadt, Germany
Telephone: 49 (0) 61 51/93 58-0
Fax: 49 (0) 61 51/93 58 58

Italy
PSC S.p.A.
Vimercate (MI), Italy
Telephone: [39] (0) 39/62903.1
Fax: [39] (0) 39/6859496

Japan
PSC Japan K.K.
Shinagawa-ku, Tokyo, Japan
Telephone: 81 (0)3 3491 6761
Fax: 81 (0)3 3491 6656

Latin America
PSC S.A., INC.
Miami, Florida, USA
Telephone: (305) 539-0111
Fax: (305) 539-0206

United Kingdom
PSC Bar Code Ltd.
Watford, England
Telephone: 44 (0) 1923 809500
Fax: 44 (0) 1923 809 505

www.psc.com

PSC Inc.
959 Terry Street
Eugene, OR
Telephone: (541) 683-5700
Fax: (541) 345-7140

R44-2313 (Rev B)

9/04