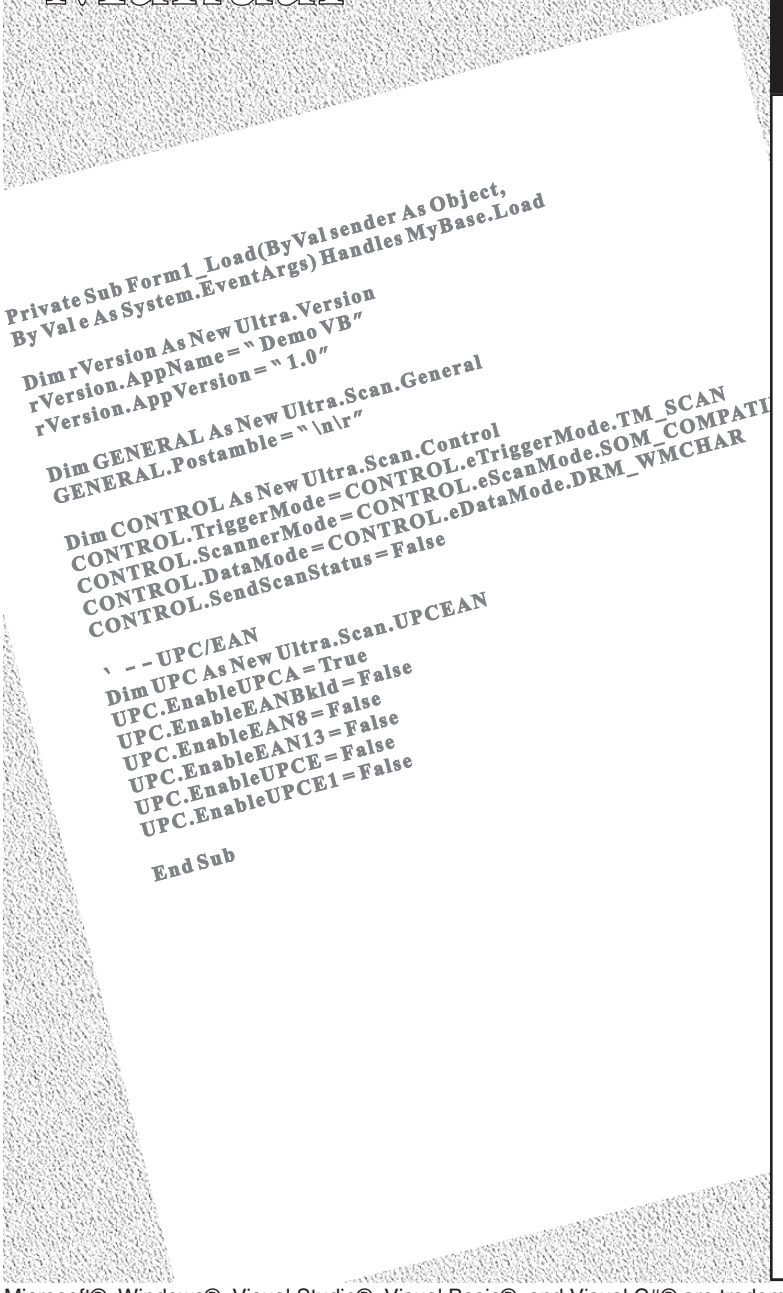
Programmer's Manual



```
Private Sub Form1_Load(ByVal sender As Object,
By Val e As System.EventArgs) Handles MyBase.Load

Dim rVersion As New Ultra.Version
rVersion.AppName = " Demo VB"
rVersion.AppVersion = " 1.0"

Dim GENERAL As New Ultra.Scan.General
GENERAL.Postamble = " \n\r"

Dim CONTROL As New Ultra.Scan.Control
CONTROL.TriggerMode = CONTROL.eTriggerMode.TM_SCAN
CONTROL.ScannerMode = CONTROL.eScanMode.SOM_COMPATI
CONTROL.DataMode = CONTROL.eDataMode.DRM_WMCHAR
CONTROL.SendScanStatus = False

' -- UPC/EAN
Dim UPC As New Ultra.Scan.UPCEAN
UPC.EnableUPCA = True
UPC.EnableEANBkld = False
UPC.EnableEAN8 = False
UPC.EnableEAN13 = False
UPC.EnableUPCE = False
UPC.EnableUPCE1 = False

End Sub
```

Monarch® Pathfinder® Ultra® Platinum 6039™ Printer

Microsoft® Visual Studio® .NET 2003 & 2005

- Microsoft® Visual Basic®
- Microsoft® Visual C#®

AVERY DENNISON

TC6039NETPM Rev. AE 1/08

Each product and program carries a respective written warranty, the only warranty on which the customer can rely. Paxar reserves the right to make changes in the product and the programs and their availability at any time and without notice. Although Paxar has made every effort to provide complete and accurate information in this manual, Paxar shall not be liable for any omissions or inaccuracies. Any update will be incorporated in a later edition of this manual.

Trademarks

Monarch®, Pathfinder®, Ultra®, MPCL, and 6039 are trademarks of Paxar Americas, Inc.
Paxar® is a trademark of Paxar Corporation.
Avery Dennison® is a trademark of Avery Dennison Corporation.
UFST, Monotype, the Monotype logo, and CG Triumvirate are trademarks of Monotype Imaging, Inc.

Avery Dennison Printer Systems Division
170 Monarch Lane
Miamisburg, OH 45342

Monotype Imaging

TABLE OF CONTENTS

Introduction 1-1
Audience 1-1
Using this Manual 1-1
System Requirements 1-2
Software Requirements 1-2
Minimum Hardware Requirements 1-2
SDK Contents 1-3
Related Documentation 1-3
About the Printer 1-4
Speaker 1-4
Memory 1-4
Display 1-4
Scanner 1-4
Keyboard 1-5
Fonts 1-6
Using Non-Resident Fonts 1-6
Developing Applications 2-1
Creating MPCL Packets 2-1
Writing Applications 2-4
Building Applications 2-4
Import Files 2-4
Scanner Function Overview 2-5
Using the Scanner 2-5
Transferring Files to the Printer 2-6
Programming Notes 2-7

Printing Functions 3-1
Stock 3-2
Calibrate 3-2
nStockType 3-3
Battery 3-4
IsBatteryOKToPrint 3-4
nBatteryLevel 3-5
Printing 3-7
Feed 3-7
File 3-8
Text 3-10
LastPrintStatus 3-12
TextDoubleByte 3-14
Byte 3-16
FileParse 3-18
Sensors 3-19
fBlackMark 3-19
fOnDemand 3-21
Misc 3-23
ClearError 3-23
LockCfgMenu 3-24
nStatus 3-25
ShiftMode 3-27

Scanning Functions....4-1
General Class....4-2
AimDuration....4-2
BdirRedundancy....4-3
GoodScanWav....4-4
LinearSecurity....4-5
NoReadWav....4-5
Preamble....4-7
Postamble....4-8
Timeout....4-9
Bar Code Classes....4-10
Codabar....4-10
Code128....4-12
Code39....4-14
Code93....4-17
D2of5....4-19
I2of5....4-21
MSI....4-24
RSS....4-26
UPCEAN....4-28

Control Class 4-34
CommitChanges 4-34
DataMode 4-35
EnableScanning 4-37
DisableAllCodes 4-38
DisableScanning 4-39
ScannerMode 4-40
SendScanStatus 4-42
Trigger 4-43
TriggerMode 4-44
SendScanStatus Codes 4-46
Sample Applications A-1
VB.NET Demo Sample A-1
C# Demo Sample A-7
VB.NET Scan/Print Sample A-14
C# Scan/Print Sample A-33

INTRODUCTION 1

The Monarch® Pathfinder® Ultra® Platinum 6039™ software development kit (SDK) helps developers write applications for the Monarch® Pathfinder® Ultra® Platinum 6039™ printer.

This manual includes the library for developers using

- Microsoft® Visual Studio® .NET 2003 (for Compact Framework 1.0).
- Microsoft® Visual Studio® .NET 2005 (for Compact Framework 2.0).

The .NET framework includes compilers for Microsoft® Visual Basic® and Microsoft® Visual C#®. This manual includes samples for both Visual Basic and Visual C#.

Information in this document supercedes information in previous versions. Check our Web site (**www.paxar.com**) for the latest documentation and release information.

Audience

This manual is written for experienced Microsoft® Visual Studio® .NET 2003 and 2005 programmers who write printer applications for the Microsoft® Windows® CE 5.0 platform. These programmers should also be familiar with the Monarch® Printer Control Language (MPCLII).

Using this Manual

Following is a summary of the contents of this manual:

Chapter		Contents
1	Introduction	Information you should know before using the SDK.
2	Developing Applications	Information about developing applications using the SDK.
3	Printing Functions	Contains syntax, definitions, and examples of each printing function.
4	Scanning Functions	Contains syntax, definitions, structures, and examples of each scanning function.
A	Sample Applications	Sample applications written using Microsoft® Visual Basic® and Microsoft® Visual C#®.

System Requirements

Following are the hardware and software requirements:

Software Requirements

- Windows® 2000 Professional Edition and Windows® 2000 Service Pack 3 or later; or Windows® XP Professional Edition and Windows® XP.

Note: The Microsoft® Windows Vista™ operating system is not currently supported.

- Microsoft® ActiveSync® synchronization software. The ActiveSync software is available on the Microsoft® Web site (Microsoft.com) by searching on ActiveSync.
- Microsoft® Visual Studio® .NET 2003 or 2005

Minimum Hardware Requirements

- Desktop computer with Pentium® II, 450Mhz (Pentium® III, 600Mhz recommended)
- Super VGA or higher monitor
- CD-ROM or DVD-ROM drive
- 96 MB (128 MB recommended) memory for Windows® 2000 Professional and 160 MB for Windows® XP Professional
- 900 MB of free hard disk space
- USB port
- USB cable (part number 125859).

SDK Contents

The SDK is located in the directory you specified during installation. It is divided into several sub-directories, as described below.

Sub-directory	Description
bin	Development tools
docs	Online documentation
include	Include files
lib	Library files
samples	Source code samples
utilities	Utility programs

Related Documentation

The following table describes other documentation for the printer:

Item	Description
Quick Reference	Includes basic start-up information such as supply loading, cleaning and minor troubleshooting.
Operator's Handbook	Includes information about using the printer, charging the battery, loading supplies, and more.
Packet Reference Manual	Includes syntax descriptions of the MPCL printer language to design a format.
System Administrator's Guide	Includes information about printer diagnostics, configuring the scanner, and using scanner diagnostics.

About the Printer

There are several printer features that you must understand before you write an application, such as the speaker, memory, display, and keyboard.

Speaker

Applications can make the printer's speaker beep for different lengths of time and frequencies or play a .wav file. For example, you might use the speaker to bring an error to the operator's attention or to indicate a good scan. Refer to the Microsoft® Visual Studio® .NET 2003 or 2005 documentation to use the speaker.

Memory

The printer contains 64 MB of Flash memory and 32 MB of RAM. The Flash memory contains the kernel and permanent storage used for user applications.

Display

The printer has a touch screen display (with a backlight) similar to a hand held computer. Refer to the Microsoft® Visual Studio® .NET 2003 or 2005 documentation to write messages to the display.

Scanner

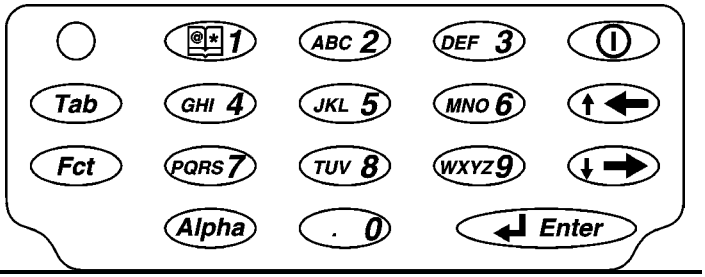
Each printer comes with the Symbol SE-955 bar code scanner.

See Chapter 4, "Scanning Functions" for functions and data structures for scanning.

Keyboard

The printer's keyboard appears to the right.

Refer to the Microsoft® Visual Studio® .NET 2003 or 2005 documentation to use the keyboard.



Key(s)	Description
Enter	Accepts data or a menu selection.
Tab	Tabs to the next tab stop or the next field. Pressing **Fct** + **Tab** backspaces a tab stop or returns to the previous field.
Fct	Performs an application-defined function when pressed with a single-digit number.
Alpha	Enters upper case or lower-case alphabetic mode.
Right Arrow	Moves the cursor to the right in a menu. Pressing **Fct** + **right arrow** scrolls the cursor down in a menu.
Left Arrow	Moves the cursor to the left in a menu. Pressing **Fct** + **left arrow** scrolls the cursor up in a menu. Backspaces in Alpha mode.
On/Off ⏻	Turns on and off the printer.
Numeric/ Alphabetic	Displays a numeric digit or letter.

Fonts

The printer has many resident fonts. You must load other fonts separately. Following is a list of these fonts and their IDs:

Standard (1), Reduced (2), Bold (3), OCRA (4), HR1 (5), and HR2 (6)	CG Triumvirate™ Typeface Bold 9 pt. (10)
EFF Swiss Bold (50)	CG Triumvirate™ Typeface 6 pt. (11)
CG Triumvirate™ Typeface Bold (Full Character Set) 6.5 pt. (1000)	CG Triumvirate™ Typeface Bold (Full Character Set) 8 pt. (1001)
CG Triumvirate™ Typeface Bold (Full Character Set) 10 pt. (1002)	CG Triumvirate™ Typeface Bold (Full Character Set) 12 pt. (1003)
CG Triumvirate™ Typeface Bold (Partial Character Set) 18 pt. (1004)	CG Triumvirate™ Typeface Bold (Partial Character Set) 22 pt. (1005)
CG Triumvirate™ Typeface Bold Condensed (Full Character Set) 6.5 pt. (1006)	CG Triumvirate™ Typeface Bold Condensed (Full Character Set) 8 pt. (1007)
CG Triumvirate™ Typeface Bold Condensed (Full Character Set) 10 pt. (1008)	CG Triumvirate™ Typeface Bold Condensed (Full Character Set) 12 pt. (1009)
CG Triumvirate™ Typeface Bold Condensed (Partial Character Set) 18 pt. (1010)	CG Triumvirate™ Typeface Bold Condensed (Partial Character Set) 22 pt. (1011)
Letter Gothic Bold (Full Character Set) 6 pt. (1012)	Letter Gothic Bold (Full Character Set) 9 pt. (1013)

Note: The partial character set fonts contain only numeric and special characters. With fonts 1012 and 1013, the space character is only 70% as wide as the other characters.

Using Non-Resident Fonts

Within your application, instantiate a new Print class such as rPrint and call a method such as Text or File to load the non-resident font or a font you have created with the MPCL® Toolbox Font Utility.

DEVELOPING APPLICATIONS 2

This chapter describes how to develop an application for the printer.

You will need to:

1. Create MPCL packets for your labels and tags, if needed.
2. Write the application.
3. Build the application.
4. Transfer or copy the application to the printer. See “Transferring Files” for more information.

Creating MPCL Packets

An application prints labels by submitting MPCL packets to the printer. Refer to the *6039 Packet Reference Manual* for more information.

Within your application, instantiate a new Print class such as rPrint and call a method such as Text when you need to print. For example,

VB.NET Sample

```
Imports Ultra                          ' Platinum Library

Dim fmtUPCA As String = "{F,1,A,R,E,200,200," & Chr(34) &
"UPCA"
& Chr(34) & "|" & "C,150,49,0,50,8,8,A,L,0,0,"
& Chr(34) & "Demo _ VB.NET" & Chr(34) & "|"
& "B,1,12,F,25,28,1,4,100,7,L,0|}"

Dim rPrint As New Print           ' Instantiate Print class
rPrint.ClearError()                   ' Clear any errors

If (Not rPrint.IsBatteryOKToPrint) Then    ' Check Battery
MsgBox("Low Battery", MsgBoxStyle.OKOnly, "Battery Check")

Else

  '--Print Format and then Batch Data

rPrint.Text = fmtUPCA
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
rPrint.Text = "1," & Chr(34) & txtUPC.Text & Chr(34) & "|}"

End If
```

This example prints the data stored in the fmtUPCA string and quotation marks (decimal value 34) around the data stored in the txtUPC.Text field. The **`rPrint.Text`** lines contain the batch data.

C# Sample

```
using Ultra;                        // Platinum Library

string fmtUPCA = "{F,1,A,R,E,200,200,\"UPCA\"|"
+ "C,150,49,0,50,8,8,A,L,0,0,\"Demo C Sharp\",1|"
+ "B,1,12,F,25,28,1,4,100,7,L,0|}";

Ultra.Print rPrint = new Print();       // Instantiate Print
                                              // class
rPrint.ClearError();                    // Clear any errors

if (!rPrint.IsBatteryOKToPrint)         // Check Battery
MessageBox.Show("Low Battery", "Battery Check");
else
{

//--Print Format and Batch Data

rPrint.Text = fmtUPCA;
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|
rPrint.Text = 1,\"" + txtUPC.Text + "\"|}";
}
```

This example prints the data stored in the fmtUPCA string and the data stored in the txtUPC.Text field. The **`rPrint.Text`** lines contain the batch data.

Writing Applications

The SDK is designed to work with the Microsoft® Visual Studio® development system using the .NET 2003 and 2005 framework. This compiler can be downloaded free of charge from the Microsoft.com Web site.

Note: To lock access to functions, such as the display, video, control panel, refer to your standard Microsoft® Windows® documentation.

Building Applications

When you are finished writing the application, select **Build Application** to build and then, **Deploy Application** to download the application. The application and associated files can also be copied by using the procedure defined in "Transferring Files."

In addition, you can also Build a Cab File. Refer to your Microsoft® Visual Studio® .NET documentation for more information.

Import Files

You must always use Ultra.dll. This .dll file contains the scanning and printing information.

For Socket programming, refer to your standard Microsoft® Windows® documentation.

Scanner Function Overview

The scanner contains a buffer to hold the data from a scan. The application receives data from the system by one of two methods. The first method is by the standard keyboard input and second method is by a special Windows message. See Chapter 4, “Scanning Functions” for information about the two methods.

Using the Scanner

To use the scanner, the application must:

1. Include the code to instantiate the class. For example, using VB.NET:

```
Dim MyScanner As New Ultra.Scan.General
MyScanner.Postamble = "\n\r"
Dim Scanner As New Ultra.Scan.Control
Scanner.SendScanStatus = False
```

2. Configure the control scanning attributes and the attributes for each bar code. For example, using VB.NET:

```
Dim UPC As New Ultra.Scan.UPCEAN
UPC.EnableUPCA = True
UPC.EnableUPCE = False
```

3. Call the CommitChanges function to save the new settings. For example, using VB.NET

```
Scanner.CommitChanges()
```

Transferring Files to the Printer

To transfer files between the printer and a computer, you need to have Microsoft® ActiveSync® Synchronization Software installed on your computer.

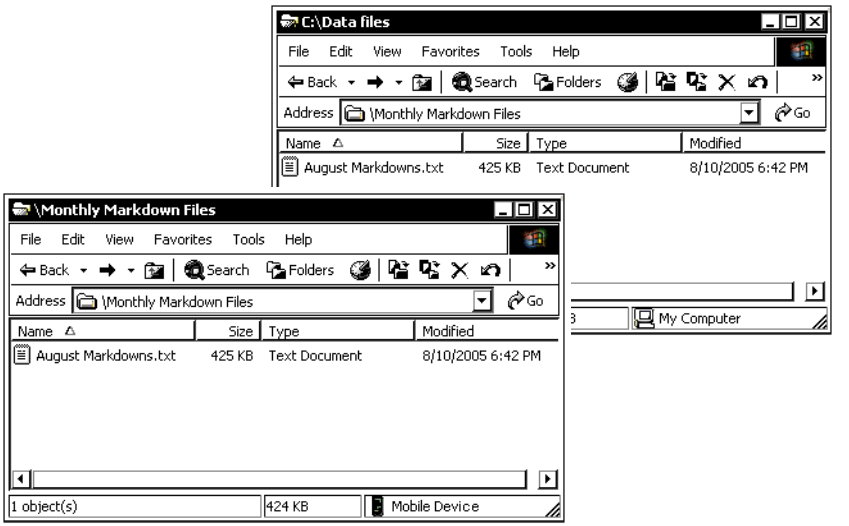
Download the ActiveSync synchronization software from the Microsoft.com Web site. Follow the download instructions on the site.

To transfer data files to the printer:

1. Open the folder with your data files on your computer.
2. Turn on the printer and wait thirty seconds for the desktop to load.

Note: For large file transfers, change the **Switch State to Suspend** setting on the printer to a longer delay time so the printer does not shut down during the transfer.

3. Attach the USB cable to your computer and printer.
4. Click **Explore** on the Microsoft ActiveSync utility after it connects to the printer. A new window appears called Mobile Device.
5. Open the destination folder for the data files on the Mobile Device (printer).



6. Drag the files from your computer to the Mobile Device folder.

Programming Notes

- Include the Ultra.dll import file in all your applications. Without it, you are not able to access any scanner or printing functions.
  For example,

```
Imports Ultra     ' for VB.NET
Using Ultra;              // for C#
```

- Test your application once you have loaded it into the printer.
- The libraries included in this SDK are designed to support Microsoft® Visual Basic® 2003 and 2005; therefore, some function names changed. Visual Basic and Visual Studio® 2005 are case insensitive. Visual C#® is case sensitive. Many developers use the same name for Type Defines and functions, just by varying the case. However, varying the case creates a new name and your application may not function as designed.
- Train the end users (operators) and/or their supervisor (system administrator) on the application. They also must know how to perform procedures (loading supplies, for example) that may vary from the generic descriptions in the *Operator's Handbook.*
- When a file is saved in RAM, it is lost when the charge in the main battery and backup battery has been depleted. A file saved in the Onboard Flash folder or SD (Secure Digital) memory card is saved even when both batteries have been depleted.
- For Socket programming, refer to your standard Microsoft® Windows® documentation.
- When you are finished writing the application, select **Build Application** to build and **Deploy Application** to download the application. In addition, you can also Build a cab file. Refer to your Microsoft® Visual Studio® .NET documentation for more information.
- To lock access to functions, such as the display, video, control panel, refer to your standard Microsoft® Windows® documentation.

PRINTING FUNCTIONS

3

The SDK contains a library of functions you can call in your application. The functions are divided into two categories: Printing Interface and Scanning Interface.

This chapter describes the printer management functions and data structures. See Chapter 4, “Scanning Functions” for scanning functions.

The function and data structure names are case-sensitive.

Note: Refer to the Microsoft® Visual Studio® .NET 2003 or 2005 documentation to configure the keyboard, sound, and display.

The libraries included in this SDK are designed to support Microsoft® Visual Basic® 2003 and 2005; therefore, some function names changed. Visual Basic and Visual Studio® 2005 are case insensitive. Visual C#® is case sensitive. Many developers use the same name for Type Defines and functions, just by varying the case. However, varying the case creates a new name and your application may not function as designed.

Type	Functions	
Stock	Calibrate nStockType	
Battery	nBatteryLevel	IsBatteryOKToPrint
Printing	Feed File Text LastPrintStatus	TextDoubleByte Byte FileParse
Sensors	fBlackMark	fOnDemand
Misc	ClearError nStatus	LockCfgMenu ShiftMode

Calibrate

Description

Calibrates the supplies in the printer and gives the supply information to the Print subsystem.

Operators can load supplies (as described in the *Operator's Handbook*) before running an application, but they cannot calibrate the supplies until the application calls this function. In general, you should display a prompt ("Load your supplies," for example) and require the operator to press a key, such as the trigger, to call this function.

Do not use this function when using fax paper because it has no black mark to detect.

Syntax

```
public void Calibrate()
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     // Instantiate class
rPrint.Calibrate();                   // Calibrate Supply
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print               ' Instantiate class
rPrint.Calibrate();                   ' Calibrate Supply
```

nStockType

Description

Sets the supply type in the printer.

```
public eSTOCK_TYPE nStockType
```

Parameters

STOCK_PAPER	Paper
STOCK_FAX	Fax
STOCK_SYNTHETIC	Synthetic

Return Values

STOCK_PAPER	Paper
STOCK_FAX	Fax
STOCK_SYNTHETIC	Synthetic

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     // Instantiate class
rPrint.nStockType = Ultra.Print.eSTOCK_TYPE.STOCK_PAPER;
                                      // Stock Type = Paper
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print               ' Instantiate class
rPrint.nStockType = rPrint.eSTOCK_TYPE.STOCK_PAPER;
                                  ' Set Stock Type to Paper
```

IsBatteryOKToPrint

Description

Checks if the printer's LiIon battery (located in the handle) is charged enough to allow printing. Check the battery level before any printing session.

Use this function immediately prior to printing, but not during printing. If you use it during printing, the return value is not accurate.

Syntax

```
public bool IsBatteryOKToPrint
```

Parameters

None

Return Values

FALSE The battery level is too low to allow printing.

TRUE The battery level is high enough to allow printing.

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();   //Instantiate class
rPrint.ClearError();                        // Clear any errors
      if (!rPrint.IsBatteryOKToPrint)       // Not OK to Print
            MessageBox.Show("Low Battery", "Battery Check");
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print                      ' Instantiate class
rPrint.ClearError()                            ' Clear any errors
      If (Not rPrint.IsBatteryOKToPrint) Then    ' Not OK to Print?
      MsgBox("Low Battery", MsgBoxStyle.OKOnly, "Battery Check")
```

nBatteryLevel

Description

Retrieves the LiIon battery's level. This battery is located in the printer's handle. This returns a value between 1 and 100, the percentage of charge left in the battery. Check the battery level before any processing.

Use this function immediately prior to printing, but not during printing. If you use this function during printing, the return value is not accurate.

Syntax

```
public int nBatteryLevel
```

Parameters

None

Return Values

0	You must charge the battery.
1 –100	The battery level is high enough to run the printer and print.
412	Could not retrieve the current battery level. Retry the function.

Example C#

```
using Ultra;
{
int nStat;
Ultra.Print rPrint = new Print();     //Instantiate class

nStat = rPrint.nBatteryLevel;
MessageBox.Show("Battery at " + nStat.ToString() + "
%","Battery
Check",MessageBoxButtons.OK,MessageBoxIcon.Asterisk,MessageBo
xDefaultButton.Button1);
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print                ' Instantiate class
Dim nStat As String

nStat = Convert.ToString(rPrint.nBatteryLevel)
MsgBox("Battery at " + nStat + " %", MsgBoxStyle.Exclamation,
"Battery Check")
```

Feed

Description

Feeds a label through the printer.

Syntax

```
public void Feed()
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     //Instantiate class
rPrint.Feed();                        // Feed a Label
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print               ' Instantiate class
rPrint.Feed()                         ' Feed a Label
```

File

Description

Writes MPCL packets to the Print subsystem.

You can send more than one packet at a time in a file.

A batch packet starts a print job, which makes an asynchronous call to the Print subsystem. After submitting a print job, the application should call LastPrintStatus in a loop, waiting until the printer becomes free.

Syntax

```
public string File
```

Parameters

FileName to be sent for printing

Return Values

None

Example C#

```
using Ultra;

string strScanData = "C39";

Ultra.Print rPrint = new Print();     //Instantiate class
rPrint.File = "\\Onboard Flash\\C39Fmt.txt";

rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|}";
```

Example VB.NET

```
Import Ultra

Dim rPrint As New Print                ' Instantiate class
Dim strScanData As String = "C39"

rPrint.File = "\Onboard Flash\C39Fmt.txt"
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
rPrint.Text = "1" & Chr(34) & strScanData & Chr(34) & "|"
rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|}"
```

Text

Description

Writes a string to the Print subsystem. MPCL packets can be sent as a string. After sending a completed MPCL Packet use the LastPrintStatus function to get a status of the MPCL Packet.

Syntax

```
public string Text
```

Parameters

String to be sent to print

Return Values

None

Example C#

```
using Ultra;

string fmtC39 = "{F,1,A,R,E,200,200,\"C39\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,20,V,23,2,4,12,100,8,C,0|}";

string strScanData = "C39";

Ultra.Print rPrint = new Print();     //Instantiate class

rPrint.Text = fmtC39;
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
rPrint.Text = "1,\"" + strScanData + "\"|2,\"" +
strScanData + "\"|}";
```

Example VB.NET

```
Import Ultra

Dim fmtC39 As String = "{F,1,A,R,E,200,200," & Chr(34) &
"C39" & Chr(34) & "|" _
& "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039" & Chr(34)
& "|" _
& "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
& "B,2,20,V,23,2,4,12,100,8,C,0|}"

Dim strScanData As String = "C39"
Dim rPrint As New Print              ' Instantiate class

rPrint.Text = fmtC39
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
rPrint.Text = "1" & Chr(34) & strScanData & Chr(34) & "|"
rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|}"
```

LastPrintStatus

Description

Retrieves status of the last MPCL packet processed by the Print Engine.

Syntax

```
public int LastPrintStatus
Parameters
```

None

Return Values

0	Successful.
703-793	A motion control error occurred. After the operator corrects the printer condition, the application must call ClearError to reset the Motion Control subsystem. Refer to the *6039 Packet Reference Manual* for more information.
Other non-zero	Error has occurred. Refer to the *6039 Packet Reference Manual* for more information.

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     //Instantiate class

if (rPrint.LastPrintStatus != 0)      // Print Status

MessageBox.Show(rPrint.LastPrintStatus.ToString(), "Print
Error", MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
```

Example VB.NET

```
Import Ultra
Dim rPrint As New Print                ' Instantiate class
Dim nStat As String

nStat = Convert.ToString(rPrint.LastPrintStatus())

If (rPrint.LastPrintStatus <> 0) Then' Print Status
MsgBox(nStat, MsgBoxStyle.Exclamation, "Print Error")
End If
```

TextDoubleByte

Description

Writes a string to the Print subsystem. MPCL packets can be sent as a string. After sending a completed MPCL Packet use the LastPrintStatus function to get a status of the MPCL Packet.

Syntax

```
public string TextDoubleByte
```

Parameters

String to be sent to print

Return Values

None

Example C#

```
using Ultra;
Ultra.Print rPrint = new Print(); //Instantiate class
rPrint.Text = "{F,4,A,R,E,200,200,\"Japan\"|";
rPrint.Text = "C,50,0,0,139,2,2,B,L,0,0,\"";
rPrint.TextDoubleByte = "\x82C6\x82C6\x82BD";
strScanData + "\"|}";
rPrint.Text = "{B,4,N,1|E,0,0,1,1,0,1|}";
```

Example VB.NET

```
Import Ultra
Dim rPrint As New Ultra.Print

rPrint.Text = "{F,4,A,R,E,200,200,""Japan""|"
rPrint.Text = "C,50,0,0,139,2,2,B,L,0,0,"""
rPrint.TextDoubleByte = ChrW(&H82C6) & ChrW(&H82C6) &
ChrW(&H82BD) & ChrW(&H82DE)
rPrint.Text = """|}"
rPrint.Text = "{B,4,N,1|E,0,0,1,1,0,1|}"
```

Byte

Description

Writes a byte array to the Print subsystem. MPCL packets can be sent as a byte array. After sending a completed MPCL Packet use the LastPrintStatus function to get a status of the MPCL Packet.

Syntax

```
public byte[] Byte
```

Parameters

Byte array to be sent to print

Return Values

None

Example C#

```
using Ultra;
string fmtC39 = "{F,1,A,R,E,200,200,\"C39\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,20,V,23,2,4,12,100,8,C,0|}";
string strScanData = "C39";
Ultra.Print rPrint = new Print(); //Instantiate class
rPrint.Text = fmtC39;
rPrint.Byte = "{B,1,N,1|E,0,0,1,1,0,1|";
rPrint.Byte = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|}";
```

Example VB.NET

```
Import Ultra
Dim fmtC39 As String = "{F,1,A,R,E,200,200," & Chr(34) &
"C39" & Chr(34) & "|" _
& "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039" & Chr(34)
& "|" _
& "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
& "B,2,20,V,23,2,4,12,100,8,C,0|}"
Dim strScanData As String = "C39"
Dim rPrint As New Print ' Instantiate class
rPrint.Text = fmtC39
rPrint.Byte = "{B,1,N,1|E,0,0,1,1,0,1|"
rPrint.Byte = "1" & Chr(34) & strScanData & Chr(34) & "|"
rPrint.Byte = "2" & Chr(34) & strScanData & Chr(34) & "|}"
```

FileParse

Description

Writes MPCL packets to the Print subsystem. You can send more than one packet at a time in a file. A batch packet starts a print job, which makes an asynchronous call to the Print subsystem. After submitting a print job, the application should call LastPrintStatus in a loop, waiting until the printer becomes free. This command is a different from the File command by reading the passed file and sending the packets one by one checking the status after each packet has been sent. Any data that is in between the packets will not be sent to the printer.

Syntax

```
public string File
```

Parameters

FileName to be sent for printing

Return Values

None

Example C#

```
using Ultra;
string strScanData = "C39";
Ultra.Print rPrint = new Print(); //Instantiate class
rPrint.FileParse = "\\Onboard Flash\\AllFmts.txt";
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|}";
```

Example VB.NET

```
Import Ultra
Dim rPrint As New Print ' Instantiate class
Dim strScanData As String = "C39"
rPrint.FileParse = "\Onboard Flash\AllFmts.txt"
rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
rPrint.Text = "1" & Chr(34) & strScanData & Chr(34) & "|"
rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|}"
```

fBlackMark

Description

Retrieves the black mark sensor's latest state. This state is not necessarily the current state because it is updated only by the Print subsystem.

Syntax

```
public bool fBlackMark
```

Parameters

None

Return Values

TRUE	The supplies are aligned on the black mark.
FALSE	The supplies are not aligned on the black mark, or the Print subsystem is busy or uninitialized.

Example C#

```
using Ultra;
{
bool blnBlackMark;
Ultra.Print rPrint = new Print();     //Instantiate class

blnBlackMark = rPrint.fBlackMark;
if (blnBlackMark)                                  // if true
{
MessageBox.Show("Blocked", "Blk Mrk Sensor",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
     else                                          // if false
     {
MessageBox.Show("Not Blocked", "Blk Mrk Sensor",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print                ' Instantiate class
Dim blnOnDemand As Boolean

blnOnDemand = rPrint.fOnDemand
If (blnOnDemand) Then                          ' If True
MsgBox("Blocked", MsgBoxStyle.Information, "OnDemand
Sensor")
Else                                           ' If False
MsgBox("Not Blocked", MsgBoxStyle.Information, "OnDemand
Sensor")
End If
```

fOnDemand

Description

Determines the On-demand sensor's current state. This sensor is an option for the printer.

Syntax

```
Public bool OnDemand
```

Parameters

None

Return Values

TRUE	The sensor is blocked.
FALSE	The sensor is not blocked.

Example C#

```
using Ultra;
{
bool blnOnDemand;
Ultra.Print rPrint = new Print();     // Instantiate class

blnOnDemand = rPrint.fOnDemand;                  // If true
if (blnOnDemand)
{
MessageBox.Show("Blocked", "OnDemand Sensor",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
else                                          // if false
{
MessageBox.Show("Not Blocked", "OnDemand Sensor",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print                ' Instantiate class
Dim blnOnDemand As Boolean

blnOnDemand = rPrint.fOnDemand
If (blnOnDemand) Then                          ' If True
MsgBox("Blocked", MsgBoxStyle.Information, "OnDemand
Sensor")
Else                                           ' If False
MsgBox("Not Blocked", MsgBoxStyle.Information, "OnDemand
Sensor")
End If
```

Misc

ClearError

Description

Resets the Motion Control subsystem after an application receives a motion control error (703-793).

The operator must correct the printer condition (a supply jam, for example) before the application calls this function.

Syntax

```
public void ClearError()
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     //Instantiate class
rPrint.ClearError();                  //Clear Error
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print               ' Instantiate class
rPrint.ClearError();                  ' Enable Clear Error
```

LockCfgMenu

Description

Allows the application to control access to all configuration menus for the printer and scanner.

Syntax

```
public void LockCfgMenu(bool fLock)
```

Parameters

fLock	TRUE to lock the menu. FALSE to release the lock on the menu.

Return Values

None

Example C#

```
using Ultra;

{
Ultra.Keypad rKeypad = new Keypad();  //Instantiate class
rKeypad.LockCfgMenu(true);            //Lock Config Menu
}
```

Example VB.NET

```
Imports Ultra

Dim rKeypad As New Keypad             ' Instantiate class
rKeypad.LockCfgMenu(True)             ' Lock Config Menu
```

nStatus

Description

Retrieves the Print subsystem's status.

After submitting a print job, the application should call GetStatus in a loop, waiting until the printer becomes free.

Syntax

```
public int nStatus(void)
```

Parameters

None

Return Values

0	The Print subsystem is ready.
1	The Print subsystem is busy.
406	MPCL parser error.
412	Could not communicate with the print engine.
601	Batch packet error.
756	Out of supplies or other motion error. Load supplies. The application must call ClearError to reset the Motion Control subsystem.
762	Low battery. Printing is disabled. Replace with a fully charged battery. (Test labels can be printed.) The application must call ClearError to reset the Motion Control subsystem.
763	Waiting to dispense label. The on-demand sensor may be blocked. The application must call ClearError to reset the Motion Control subsystem.
791	Error pending. Operator intervention is required to clear the error. The application must call ClearError to reset the Motion Control subsystem.
703-793 not listed above	A motion control error occurred. After the operator corrects the printer condition, the application must call ClearError to reset the Motion Control subsystem. Refer to the *6039 Packet Reference Manual* for more information.

Example C#

```
using Ultra;
{
Ultra.Print rPrint = new Print();     //Instantiate class
if (rPrint.nStatus  > 1)         // Check Status
MessageBox.Show(rPrint.nStatus.ToString(), "Status",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
}
```

Example VB.NET

```
Imports Ultra
Dim rPrint As New Print                ' Instantiate class
Dim nStat As String

nStat = Convert.ToString(rPrint.nStatus())
      If (rPrint.nStatus > 1) Then     ' Check Status
      MsgBox(nStat, MsgBoxStyle.Exclamation, "Status")
      End If
```

ShiftMode

Description

Allows the application to determine the current shift mode of the keypad and also to set the shift mode.

```
public eKSM_LEVEL ShiftMode
```

Parameters

KSM_NORMAL	Normal Mode -Numeric (unshifted)
KSM_FUNCTION	Function Mode (F1-F10)
KSM_LOWERALPHA	Lower-case Alpha Mode
KSM_UPPERALPHA	Upper-case Alpha Mode

Return Values

KSM_NORMAL	Normal Mode -Numeric (unshifted)
KSM_FUNCTION	Function Mode (F1-F10)
KSM_LOWERALPHA	Lower-case Alpha Mode
KSM_UPPERALPHA	Upper-case Alpha Mode

Example C#

```
using Ultra;
{
Ultra.Keypad rKeypad = new Keypad();  //Instantiate Class
rKeypad.ShitfMode = Ultra.Keypad.eKSMLEVEL.KSM_NORMAL;
                                  // Force Keypad to numbers
}
```

Example VB.NET

```
Imports Ultra
Dim rKeypad As New Ultra.Keypad ' Instantiate class

rKeypad.ShiftMode = rKeypad.eKSM_LEVEL.KSM_NORMAL
                                  ' Force Keypad to numbers
```

SCANNING FUNCTIONS 4

The SDK contains a library of functions you can call in your application. The functions are divided into two categories: Printing Interface and Scanning Interface.

This chapter describes the scanning functions and data structures. Refer to Chapter 3, “Printing Functions” for printing.

The function and data structure names are case-sensitive.

Note: Refer to the Microsoft® Visual Studio® .NET 2003 or 2005 documentation to configure the keyboard, sound, and display.

The libraries included in this SDK are designed to support Microsoft® Visual Basic® 2003 and 2005; therefore, some function names changed. Visual Basic and Visual Studio® 2005 are case insensitive. Visual C#® is case sensitive. Many developers use the same name for Type Defines and functions, just by varying the case. However, varying the case creates a new name and your application may not function as designed.

Type	Functions	
General Class	AimDuration BdirRedundancy GoodScanWav LinearSecurity	NoReadWav Preamble Postamble Timeout
Bar Code Classes:	Codabar Code128 Code39 Code93 Code128	D2of5 I2of5 MSI RSS UPCEAN
Control Class	CommitChanges DataMode EnableScanning DisableScanning DisableAllCodes	ScannerMode SendScanStatus Trigger TriggerMode

General Class

AimDuration

Description

Sets the AIM Duration, which is the duration of the aiming beam when the scanner is activated.

Syntax

```
public int AimDuration
```

Parameters

AimDuration	00 - 99 tenths of a second 0 disables. **Default**

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        //Instantiate Class
GENERAL.AimDuration =  0;

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class

CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.AimDuration = 0
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

BdirRedundancy

Description

Enables Bi-Directional Redundancy, which specifies that good scans must occur in both directions (forward and reverse) for the scan to be complete.

Syntax

```
public bool BdirRedundancy
```

Parameters

BdirRedundancy	TRUE	Scans must occur in both directions
	FALSE	**Default**

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        //Instantiate Class
GENERAL.BdirRedundancy = false;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.BdirRedundancy = False
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

GoodScanWav

Description

Sets the file for a Good Scan. This sound is heard whenever a bar code is successfully scanned. To have no sound, clear this field.

Syntax

```
public string GoodScanWav
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL.GoodScanWav = "ding.wav";
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.GoodScanWav = "ding.wav"
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

LinearSecurity

Description

Sets the Linear Security, which is how many times to scan the same barcode to determine a successful read. Select a higher level for lower quality bar codes.

Syntax

```
public int LinearSecurity
```

Parameters

LinearSecurity 1 – 4 scans
Default: 1

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL. LinearSecurity = 1;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.LinearSecurity = 1
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

NoReadWav

Description

Sets the file for a No Scan. This sound is heard whenever a bar code is unsuccessfully scanned. To have no sound, clear this field.

Syntax

```
public string NoReadWav
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL.NoReadWav = "noscan.wav";
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.NoReadWav = "noscan.wav"
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Preamble

Description

Sets the Preamble, which specifies the characters to preface returned data from scanning. The Preamble can be up to twenty user-defined characters.

Syntax

```
public string Preamble
```

Parameters

Preamble	up to 20 characters **Default:** Empty

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL.Preamble = "[";
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.Preamble = "["
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Postamble

Description

Sets the Postamble, which is the data to be sent after each scanned barcode. The Postamble can be up to twenty user-defined characters.

Syntax

```
public string Postamble
```

Parameters

Postamble up to 20 characters
Default: Empty

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL.Postamble = "]";
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.Postamble = "]"
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Timeout

Description

Sets the scan Timeout in tenths of seconds, which is the amount of time the scanner beam is on before turning off when the trigger is pressed.

Syntax

```
public int Timeout
```

Parameters

Timeout	5 – 99 tenths of a second **Default:** 30

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                        // Instantiate class
GENERAL.Timeout = 30; ();       // Timeout after 30 seconds

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                        // Instantiate class
CONTROL.CommitChanges();
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.Timeout = 30
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Codabar

Description

Sets the scanner configuration values for CODABAR bar code.

Syntax

```
public bool Enable
public bool FixedLength
public int Length1
public int Length2
public bool CLSIEdit
public bool NOTISEdit
```

Field	Description
Enable	Enables/disables the ability to scan Codabar bar codes. **Default:** FALSE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** FALSE
Length1 *Length2*	Specifies lengths (including start and stop characters) for Codabar bar codes. Length 1: 0, 1-99, **Default:** 5 Length 2: 0, 1-99, **Default:** 55
CLSIEdit	Enables/disables the ability to strip the start and stop characters from 14-character Codabar bar codes and insert a space after the first, fifth, and tenth characters. **Default:** FALSE
NOTISEdit	Enables/disables the ability to strip the start and stop characters from Codabar bar codes. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Codabar CODABAR = new Ultra.Scan.Codabar();
                                  // Instantiate class
CODABAR.Enable = true;            // Enable Codabar Symbology
CODABAR.CLSIEdit = true;          // Enable CLSIEDit
CODABAR.NOTISEdit = true;         // Enable NOTISEdit
CODABAR.FixedLength = false;      // Variable Length Enabled
CODABAR.Length1 = 2;              // Variable Length1 = 2
CODABAR.Length2 = 55;             // Variable Length2  = 55

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra
Dim CODABAR As New Scan.Codabar          ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control    ' Instantiate class
CODABAR.Enable = True               ' Enable Codabar Symbology
CODABAR.CLSIEdit = True             ' Enable CLSIEDit
CODABAR.NOTISEdit = True            ' Enable NOTISEdit
CODABAR.FixedLength = False         ' Variable Length Enabled
CODABAR.Length1 = 2                 ' Variable Length1=2
CODABAR.Length2 = 55                ' Variable Length2=55
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Code128

Description

Sets the scanner configuration values for CODE128 bar code.

Syntax

```
public bool Enable
public bool UCCEAN128
public bool ISBT128
```

Field	Description
Enable	Enables/disables the ability to scan Code 128 bar codes. **Default:** TRUE
UCCEAN128	Enables/disables the ability to scan UCC/EAN-128 bar codes. **Default:** TRUE
ISBT128	Enables/disables the ability to scan ISBT 128 bar codes. **Default:** TRUE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Code128 C128 = new Ultra.Scan.Code128();
                                  // Instantiate class
C128.Enable = true;               // Enable Code 128 Symbology
C128.ISBT128 = true;              // Enable ISBT128
C128.UCCEAN128 = true;            // Enable UCCEAN128

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                                  // Configuration
```

Example VB.NET

```
Imports Ultra

Dim C128 As New Scan.Code128            ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

C128.Enable = True                ' Enable Code128 Symbology
C128.ISBT128 = True               ' Enable ISBT128
C128.UCCEAN128 = True             ' Enable UCCEAN128

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Code39

Description

Sets the scanner configuration values for Code39 bar code.

Syntax

```
public bool Enable;
public bool Trioptic;
public bool XlatetoCode32;
public bool Code32Prefix;
public bool FixedLength;
public int Length1;
public int Length2;
public bool VerifyCD;
public bool XmitCD;
public bool FullASCII;
```

Field	Description
Enable	Enables/disables the ability to scan Code 39 bar codes. **Default:** TRUE
Trioptic	Enables/disables the ability to scan Trioptic Code 39 bar codes. Do not enable *Trioptic* and *FullASCII* at the same time. **Default:** FALSE
XlatetoCode32	Enables/disables the ability to convert Code 39 bar codes to Code 32 bar codes. You must enable *Enable* when enabling this parameter. **Default:** FALSE
Code32Prefix	Enables/disables the ability to add "A" as a prefix to all Code 32 bar codes. You must enable *xlatetoCode32* when enabling this parameter. **Default:** FALSE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** FALSE
VerifyCD	Enables/disables the ability to check the integrity of Code 39 bar codes. When this parameter is enabled, only Code 39 symbols with a modulo 43 check digit are decoded. **Default:** FALSE

Field	Description
XmitCD	Enables/disables the ability to transmit check digits with the data. **Default:** FALSE
FullASCII	Enables/disables the ability to scan Full ASCII Code 39 bar codes. The scanner cannot distinguish Code 39 bar codes from Full ASCII Code 39 bar codes. Do not enable *Trioptic* and *FullASCII* at the same time. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Code39 C39 = new Ultra.Scan.Code39();
                                // Instantiate class
C39.Enable = true;              // Enable Code 39 Symbology
C39.Trioptic = false;           // Disable Trioptic
C39.XlateToCode32 = false;      // Disable Convert to Code 39
C39.FullASCII = false;          // Disable Full Ascii
C39.XmitCD = false;             // Disable Transmit C/D
C39.VerifyCD = false;           // Disable Verify C/D
C39.FixedLength = false;        // Variable Length Enabled
C39.Length1 = 2;                // Variable Length1 = 2
C39.Length2 = 55;               // Variable Length2 = 55

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                                // Configuration
```

Example VB.NET

```
Imports Ultra
Dim C39 As New Scan.Code39            ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control  ' Instantiate class

C39.Enable = True               ' Enable Code 39 Symbology
C39.Trioptic = False            ' Disable Trioptic
C39.XlateToCode32 = False       ' Disable Convert to Code 39
C39.Code32Prefix = False        ' Disable Code 39 Prefix
C39.FullASCII = False           ' Disable Full Ascii
C39.XmitCD = False              ' Disable Transmit C/D
C39.VerifyCD = False            ' Disable Verify C/D
C39.FixedLength = False         ' Variable Length Enabled
C39.Length1 = 2                 ' Variable Length1=2
C39.Length2 = 55                ' Variable Length2=55

CONTROL.CommitChanges()
               ' Save Changes to the Scanner Configuration
```

Code93

Description

Sets the scanner configuration values for CODE93 bar code.

Syntax

```
public bool Enable;
public bool FixedLength;
public int Length1;
public int Length2;
```

Field	Description
Enable	Enables/disables the ability to scan Code 93 bar codes. **Default:** FALSE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** FALSE
Length1 *Length2*	Specifies lengths (including start and stop characters) for Code 93 bar codes. Length 1: 0, 2-99, **Default:** 4 Length 2: 0, 2-99, **Default:** 55

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Code93 C93 = new Ultra.Scan.Code93();
                                  // Instantiate class
Code93.Enable = TRUE;             // Enable Code93 Symbology
Code93.FixedLength = FALSE;       // Variable Length Enabled
Code93.Length1 = 4;               // Variable Lenght1 = 4
Code93.Length2 = 20;              // Variable Lenght1 = 20
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim C93 As New Scan.Code93           ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control' Instantiate class

Code93.Enable = TRUE;             ' Enable Code93 Symbology
Code93.FixedLength = FALSE;       ' Variable Length Enabled
Code93.Length1 = 4;               ' Variable Lenght1 = 4
Code93.Length2 = 20;              ' Variable Lenght1 = 20

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

D2of5

Description

Sets the scanner configuration values for D2of5 bar code.

Syntax

```
public bool Enable;
public bool FixedLength;
public int Length1;
public int Length2;
```

Field	Description
Enable	Enables/disables the ability to scan D2of5 bar codes. **Default:** FALSE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** TRUE
Length1 *Length2*	Specifies lengths (including start and stop characters) for D2of5 bar codes. Length 1: 0, 2-99, **Default:** 12 Length 2: 0, 2-99, **Default:** 0

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.D2OF5 D2OF5 = new Ultra.Scan.D2OF5 ();
                                  // Instantiate class
D2OF5.Enable = TRUE;              // Enable D2OF5 Symbology
D2OF5.FixedLength = FALSE;        // Variable Length Enabled
D2OF5.Length1 = 12;               // Variable Lenght1 = 12
D2OF5.Length2 = 0;                // Variable Lenght1 = 0
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim D2OF5 As New Scan.D2OF5          ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control ' Instantiate class

D2OF5.Enable = TRUE;              ' Enable D2OF5 Symbology
D2OF5.FixedLength = FALSE;        ' Variable Length Enabled
D2OF5.Length1 = 12;               ' Variable Lenght1 = 12
D2OF5.Length2 = 0;                ' Variable Lenght1 = 0

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

I2of5

Description

Sets the scanner configuration values for I2of5 bar code.

Syntax

```
public bool Enable;
public bool FixedLength;
public int Length1;
public int Length2;
public bool VerifyCD;
public bool XmitCD;
public bool XlatetoEAN13;
```

Field	Description
Enable	Enables/disables the ability to scan I2of5 bar codes. **Default:** TRUE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** TRUE
Length1 *Length2*	Specifies lengths (including start and stop characters) for I2of5 bar codes. Length 1: 0, 2-99, **Default:** 14 Length 2: 0, 2-99, **Default:** 0
VerifyCD	Specifies whether the scanner should check the integrity of 2of5 bar codes to ensure they comply with either the Uniform Symbology Specification (USS) or Optical Product Code Council (OPCC) algorithms. **Default:** FALSE
XmitCD	Enables/disables the requirement to transmit check digits with the data. **Default:** FALSE
XlatetoEAN13	Enables/disables the requirement to convert 14-character I2of5 bar codes into EAN13 bar codes, and transmit them as EAN13 bar codes. To use this parameter, enable Enable, set the length to 14, and ensure the data has a leading zero and a valid EAN 13 check digit. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.I2OF5 I2OF5 = new Ultra.Scan.I2OF5 ();
                                  // Instantiate class
I2OF5.Enable = True;              // Enable I2OF5 Symbology
I2OF5.FixedLength = True;         // Variable Length Enabled
I2OF5.Length1 = 14;               // Variable Lenght1 = 14
I2OF5.Length2 = 0;                // Variable Lenght1 = 0
I2of5.VerifyCD = True;            // Enable Verify C/D
I2of5.XmitCD = True;              // Enable Transmit C/D
I2of5.XlatetoEAN13= False;        // Disable Convert to EAN13

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim I2OF5 As New Scan.I2OF5             ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

I2OF5.Enable = True;              ' Enable I2OF5 Symbology
I2OF5.FixedLength = FALSE;        ' Variable Length Enabled
I2OF5.Length1 = 14;               ' Variable Lenght1 = 14
I2OF5.Length2 = 0;                ' Variable Lenght1 = 0
I2of5.VerifyCD = True;            ' Enable Verify C/D
I2of5.XmitCD = True;              ' Enable Transmit C/D
I2of5.XlatetoEAN13= False;        ' Disable Convert to EAN13

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

MSI

Description

Sets the scanner configuration values for MSI bar code.

Syntax

```
public bool Enable;
public bool FixedLength;
public int Length1;
public int Length2;
public bool Use2CDs;
public bool XmitCD;
public bool UseMod10Mod11CDAlg;
```

Field	Description
Enable	Enables/disables the ability to scan MSI bar codes. **Default:** FALSE
FixedLength	If FixedLength is TRUE, lengths 1 and 2 are fixed; if FixedLength is FALSE, length 1 is the minimum and length 2 is the maximum. **Default:** FALSE
Length1 *Length2*	Specifies lengths (including start and stop characters) for MSI bar codes. Length 1: 0, 1-99, **Default:** 6 Length 2: 0, 1-99, **Default:** 55
Use2CDs	Indicates the bar code contains two check digits instead of one. If True, enable UseMod10M0d11CDAlg. **Default:** FALSE
XmitCD	Enables/disables the requirement to transmit data with the check digit. **Default:** FALSE
UseMod10Mod11CDAlg	Uses the Mod10/Mod11 check digit algorithm instead of Mod10/Mod10 algorithm. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.MSI MSI = new Ultra.Scan.MSI ();
                                  // Instantiate class
MSI.Enable = True;                // Enable MSI Symbology
MSI.FixedLength = True;           // Variable Length Enabled
MSI.Length1 = 6;                  // Variable Lenght1 = 6
MSI.Length2 = 55;                 // Variable Lenght1 = 55
MSI.Use2CDs = False;              // Disable two check digits
MSI.XmitCD = False;               // Disable Transmit C/D
MSI.UseMod10Mod11CDAl= False;     // Enable Mod10/Mod10

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim MSI As New Scan.MSI                 ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

MSI.Enable = True;                ' Enable MSI Symbology
MSI.FixedLength = True;           ' Variable Length Enabled
MSI.Length1 = 6;                  ' Variable Lenght1 = 6
MSI.Length2 = 55;                 ' Variable Lenght1 = 55
MSI.Use2CDs = False;              ' Disable two check digits
MSI.XmitCD = False;               ' Disable Transmit C/D
MSI.UseMod10Mod11CDAl= False;     ' Enable Mod10/Mod10

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

RSS

Description

Sets the scanner configuration values for RSS bar code.

Syntax

```
public bool Enable14;
public bool EnableLimited;
public bool EnableExpanded;
public bool ConvertUPCEAN;
```

Field	Description
Enable14	Enables/disables the ability to scan RSS bar codes. **Default:** FALSE
EnableLimited	Enables/disables the ability to scan RSS Limited bar codes. **Default:** FALSE
EnableExpanded	Enables/disables the ability to scan RSS Expanded bar codes. **Default:** FALSE
ConvertUPCEAN	Enables/disables the ability to convert to UPC/EAN bar codes. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.RSS RSS = new Ultra.Scan.RSS ();
                           // Instantiate class
RSS.Enable14 = False;      // Disable RSS Symbology
RSS.EnableLimited = FALSE; // Disable RSS Limited
RSS.EnableExpanded = FALSE;// Disable RSS Expanded
RSS.ConvertUPCEAN = FALSE; // Disable Converions to UPC/EAN

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim RSS As New Scan.RSS                 ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

RSS.Enable14 = False;       ' Disable RSS Symbology
RSS.EnableLimited = FALSE;  ' Disable RSS Limited
RSS.EnableExpanded = FALSE; ' Disable RSS Expanded
RSS.ConvertUPCEAN = FALSE;  ' Disable Conversion to UPC/EAN

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

UPCEAN

Description

Sets the scanner configuration values for UPC and EAN bar codes.

Syntax

```
public bool EnableUPCA;
public bool EnableUPCE;
public bool EnableUPCE1;
public bool EnableEAN8;
public bool EnableEAN13;
public bool EnableBklnd;
public eSUPPLEMENTAL Supplement;
public int  SupRedundancy;
public bool XmitUPCACD;
public bool XmitUPCECD;
public bool XmitUPCE1CD;
public eXMITUPCPREAMBLE XmitUPCAPre;
public eXMITUPCPREAMBLE XmitUPCEPre;
public eXMITUPCPREAMBLE XmitUPCE1Pre;
public bool XlateEToA;
public bool XlateE1toA;
public bool EANZeroExtend;
public bool Xlate8to13;
public int  Security;
public bool CouponCode;
```

Field	Description
EnableUPCA	Enables/disables the ability to scan UPCA bar codes. **Default:** TRUE
EnableUPCE	Enables/disables the ability to scan UPCE0 bar codes. **Default:** TRUE
EnableUPCE1	Enables/disables the ability to scan UPCE1 bar codes. **Default:** FALSE
EnableEAN8	Enables/disables the ability to scan EAN8 bar codes. **Default:** TRUE
EnableEAN13	Enables/disables the ability to scan EAN13 bar codes. **Default:** TRUE
EnableBkInd	Enables/disables the ability to scan Bookland EAN bar codes. **Default:** FALSE
Supplement	Specifies how to treat UPC and EAN bar codes with supplemental characters (UPCA+2, for example). Values are: USM_REQUIRED The scanner scans bar codes with supplemental characters only. For example, it scans a UPCA+2 bar code, but not UPCA. USM_IGNORE The scanner ignores supplemental characters. For example, it scans a UPCA+2 bar code as a UPCA. USM_AUTO Uses scanning information as specified in SupRedundancy. **Default:** USM_IGNORE
SupRedundancy	If using USM_AUTO for Supplement, this sets the number of times a symbol without supplemental information is decoded. Values are 2-20. **Default:** 7
XmitUPCACD	Enables/disables the requirement to transmit UPCA bar codes with the check digit. **Default:** TRUE
XmitUPCECD	Enables/disables the requirement to transmit UPCE0 bar codes with the check digit. **Default:** TRUE
XmitUPCE1CD	Enables/disables the requirement to transmit UPCE1 bar codes with the check digit. **Default:** TRUE

Field	Description
XmitUPCAPre	Specifies how to transmit UPCA Preamble characters. Values are: 1 = XUP_NONE Do not transmit. 2 = XUP_SYSCHAR Transmit system character. 3 = XUP_SYSCOUNT Transmit system character and country code. **Default:** XUP_SYSCHAR
XmitUPCEPre	Specifies how to transmit UPCE Preamble characters. Values are: 1 = XUP_NONE Do not transmit. 2 = XUP_SYSCHAR Transmit system character. 3 = XUP_SYSCOUNT Transmit system character and country code. **Default:** XUP_SYSCHAR
XmitUPCE1Pre	Specifies how to transmit UPCE1 Preamble characters. Values are: 1 = XUP_NONE Do not transmit. 2 = XUP_SYSCHAR Transmit system character. 3 = XUP_SYSCOUNT Transmit system character and country code. **Default:** XUP_SYSCHAR
XlateEToA	Translates a UPCE bar codes to a UPCA bar code. **Default:** FALSE
XlateE1toA	Translates a UPCE1 bar codes to a UPCA bar code. **Default:** FALSE
EANZeroExtend	Adds five leading zeros to scanned EAN8 bar codes to make them compatible with EAN13 bar codes. **Default:** FALSE
Xlate8TO13	Translates an EAN8 bar code to an EAN13 bar code. **Default:** FALSE
Security	Sets the scan security level, which is how many times to scan the same bar code to determine a successful read. Choose the minimum-security level needed. Values are 0 Use when most scans are successful. 1 Use when the unsuccessful scans are related to characters 1, 2, 7, and 8. 2 Use when the unsuccessful scans are not limited to characters 1, 2, 7, and 8. 3 Choose 4 if unsuccessful scans still occur at level 2. **Default:** 0

Field	Description
CouponCode	Enables the ability to scan UPCA/EAN Coupon Code bar codes. **Default:** FALSE

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.UPC UPCEAN = new Ultra.Scan.UPCEAN ();
                                  // Instantiate class
UPCEAN.EnableUPCA = TRUE;         // Enable UPCA Symbology
UPCEAN.EnableUPCE = FALSE;        // Disable UPCA Symbology
UPCEAN.EnableUPCE1 = FALSE;       // Disable UPCE1 Symbology
UPCEAN.EnableEAN8 = FALSE;        // Disable EAN8 Symbology
UPCEAN.EnableEAN13 = FALSE;       // Disable EAN13 Symbology
UPCEAN.EnableBklnd = FALSE;       // Disable Bklnd Symbology
UPCEAN.Supplement =
Ultra.Scan.UPCEAN.eSUPLEMENTAL.USM_IGNORE;
                             // Ignore Supplemental characters
UPCEAN.nSupRedundancy = 7;        // Number of times
                                  // Supplemental is decoded
UPCEAN.fXmitUPCACD = TRUE;        // Enable Transmit UPCA C/D
UPCEAN.fXmitUPCECD = FALSE;       // Disable Transmit UPCE C/D
UPCEAN.fXmitUPCE1CD = FALSE;      // Disable Transmit UPCE1 C/D
UPCEAN.nXmitUPCAPre =
     Ultra.Scan.UPCEAN.eXMITUPCPREAMPLE.XUP_SYSCHAR;
                                  // Transmit UPCA Preamble
UPCEAN.nXmitUPCEPre =
     Ultra.Scan.UPCEAN.eXMITUPCPREAMPLE.XUP_SYSCHAR;
                                  // Transmit UPCE Preamble
UPCEAN.nXmitUPCE1Pre =
     Ultra.Scan.UPCEAN.eXMITUPCPREAMPLE.XUP_SYSCHAR;
                                  // Transmit UPCE1 Preamble
UPCEAN.fxlateEToA = FALSE;   // Disable Convert UPCE to UPCA
UPCEAN.fxlateE1toA = FALSE;  // Disable Convert UPCE1 to UPCA
UPCEAN.fEANZeroExtend = FALSE;  // Disable adding zeros
' for EAN13
UPCEAN.fXlate8TO13= FALSE;        // Disable Convert to
                                  // EAN8 to EAN13
UPCEAN.nSecurity = 1;             // Enable Scan Security level
UPCEAN.fCouponCode = TRUE;        // Enable Coupon Scanning

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim UPC As New Scan.UPCEAN             ' Instantiate class
Dim CONTROL As New Ultra.Scan.Control  ' Instantiate class

UPCEAN.EnableUPCA = TRUE;         ' Enable UPCA Symbology
UPCEAN.EnableUPCE = FALSE;        ' Disable UPCA Symbology
UPCEAN.EnableUPCE1 = FALSE;       ' Disable UPCE1 Symbology
UPCEAN.EnableEAN8 = FALSE;        ' Disable EAN8 Symbology
UPCEAN.EnableEAN13 = FALSE;       ' Disable EAN13 Symbology
UPCEAN.EnableBklnd = FALSE;       ' Disable Bklnd Symbology
UPCEAN.Supplement = UPC.eSUPPLEMENTAL.USM_IGNORE;
                           ' Ignore Supplemental characters
UPCEAN.nSupRedundancy = 7;        ' Number of times
                                  ' Supplemental is decoded
UPCEAN.fXmitUPCACD = TRUE;        ' Enable Transmit UPCA C/D
UPCEAN.fXmitUPCECD = FALSE;       ' Disable Transmit UPCE C/D
UPCEAN.fXmitUPCE1CD = FALSE;      ' Disable Transmit UPCE1 C/D
UPCEAN.nXmitUPCAPre = UPC.eXMITUPCPREAMBLE.XUP_SYSCHAR;
                                  ' Transmit UPCA Preamble
UPCEAN.nXmitUPCEPre = UPC.eXMITUPCPREAMBLE.XUP_SYSCHAR;
                                  ' Transmit UPCE Preamble
UPCEAN.nXmitUPCE1Pre = UPC.eXMITUPCPREAMBLE.XUP_SYSCHAR;
                                  ' Transmit UPCE1 Preamble
UPCEAN.fxlateEToA = FALSE;  ' Disable Convert UPCE to UPCA
UPCEAN.fxlateE1toA = FALSE; ' Disable Convert UPCE1 to UPCA
UPCEAN.fEANZeroExtend = FALSE;  ' Disable adding zeros
                                ' for EAN13
UPCEAN.fXlate8TO13= FALSE;      ' Disable Convert to
                                ' EAN8 to EAN13
UPCEAN.nSecurity = 1;           ' Enable Scan Security level
UPCEAN.fCouponCode = TRUE;      ' Enable Coupon Scanning

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

CommitChanges

Description

Retrieves the scanner configuration data from the application and saves the reconfigured settings to the scan engine.

Syntax

```
public void CommitChanges();
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

DataMode

Description

Retrieves the Data Receive Mode from the application and saves the reconfigured settings to the scan engine.

Syntax

```
public eDATAMODE DataMode;
```

Parameters

DataMode	The supply type. Values are	
	DRM_WMCHAR	The application receives data by the keyboard buffer. **Default**
	DRM_SMSCANCHAR	The application receives data by the scanner through a special message.

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

Scanner.DataMode = Ultra.Scan.Control.eDATAMODE.DRM_WMCHAR;
                           // Data from Keyboard buffer

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim Control As New Ultra.Scan.Control   ' Instantiate class

Control.DataMode = Control.eDATAMODE.DRM_SMSCANCHAR
                                  ' Data from the scanner

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

EnableScanning

Description

Allows the scanner to scan the defined bar codes.

Syntax

```
public void EnableScanning();
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

CONTROL.EnableScanning();  // Enable Scanning

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra
Dim CONTROL As New Ultra.Scan.Control  ' Instantiate class

CONTROL.EnableScanning()               ' Enable Scanning

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

DisableAllCodes

Description

Disables all bar codes. This function will cause all barcodes to be disabled from being scanned. Use this function when you want to enable only a few selected codes.

Syntax

```
public void DisableAllCodes();
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.DisableAllCodes();       // Disable all bar codes

// Enable needed bar codes here

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra
Dim CONTROL As New Ultra.Scan.Control ' Instantiate class

CONTROL.DisableAllCodes()             ' Disable all bar codes

' Enable needed bar codes here

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

DisableScanning

Description

Disables the scanner from scanning the defined bar codes.

Syntax

```
public void DisableScanning();
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.DisableScanning();       // Disable Scanning

CONTROL.CommitChanges();       //--Save Changes to the Scanner
                               // Configuration
```

Example VB.NET

```
Imports Ultra
Dim CONTROL As New Ultra.Scan.Control ' Instantiate class

CONTROL.DisableScanning()              ' Disable Scanning

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

ScannerMode

Description

Retrieves the current scanner mode data containing the default values and saves the reconfigured settings to the scan engine.

Syntax

```
public eSCANMODE ScannerMode;
```

Parameters

None

Return Values

SOM_MOMENTARY	The scanner is on when the trigger is pressed and goes off when the trigger is released.
SOM_CONTINUOUS	The scanner is always on. A good scan causes the scanner to reset and continue scanning.
SOM_COMPATIBLE	The scanner operates in Monarch® 6037™ compatible mode, which means the scanner is on when the trigger is pressed and goes off after a successful scan or a predetermined timeout period. **Default**

Example C#

```
using Ultra;

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                           // Instantiate class

CONTROL.ScannerMode =
     Ultra.Scan.Control.eSCANMODE.SOM_COMPATIBLE;
                           // Enable Compatibility with 6037

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           //  Configuration
```

Example VB.NET

```
Imports Ultra

Dim CONTROL As New Ultra.Scan.Control  ' Instantiate class

CONTROL.ScannerMode = CONTROL.eSCANMODE.SOM_COMPATIBLE
                           ' Enable Compatibility with 6037

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

SendScanStatus

Description

Enable Send Scan Status to return the data after any scan. This data precedes the bar code and includes the length of data and bar code type.

Syntax

```
public bool SendScanStatus
```

Parameters

SendScanStatus	TRUE	Enable Send Scan Status
	FALSE	**Default**

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.General GENERAL = new Ultra.Scan.General();
                                    // Instantiate Class
GENERAL.SendScanStatus = false;
                                    // Send Scan Status
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                    // Instantiate class
CONTROL.CommitChanges();
                //--Save Changes to the Scanner Configuration
```

Example VB.NET

```
Imports Ultra
Dim GENERAL As New Scan.General       ' Instantiate class
Dim CONTROL As New Scan.Control       ' Instantiate class
GENERAL.SendScanStatus = False
                                      ' Send Scan Status
CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

Trigger

Description

Initiates a scan, placing the scanned data in the scanner buffer. If the LED on the keypad turns green, the scan was successful. This function works with each scanner.

Syntax

```
public void Trigger();
```

Parameters

None

Return Values

None

Example C#

```
using Ultra;
Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();
                                  // Instantiate class

CONTROL.Trigger();                // Start Scanning

CONTROL.CommitChanges();   //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra
Dim CONTROL As New Ultra.Scan.Control ' Instantiate class

CONTROL.Trigger()                      ' Start Scanning

CONTROL.CommitChanges()
                 ' Save Changes to the Scanner Configuration
```

TriggerMode

Description

Sets the trigger mode configuration values the application set.

Syntax

```
public eTRIGGERMODE TriggerMode;
```

Parameters

TriggerMode The supply type. Values are

TM_SCAN	Pressing the trigger turns on the scanner.
TM_DROP	The printer ignores the trigger press and does not turn on the scanner.
TM_FORWARD	The printer passes the trigger press to the application as an F11, allowing more control of the application. You can code a custom application to perform a special function whenever it receives an F11. **Default**

Return Values

None

Example C#

```
using Ultra;

Ultra.Scan.Control CONTROL = new Ultra.Scan.Control;
                           // Instantiate class

CONTROL.TriggerMode =
Ultra.Scan.Control.eTRIGGERMODE.TM_SCAN;
                           // Enable scanning by trigger

CONTROL.CommitChanges      //--Save Changes to the Scanner
                           // Configuration
```

Example VB.NET

```
Imports Ultra

Dim CONTROL As New Ultra.Scan.Control   ' Instantiate class

CONTROL.TriggerMode = CONTROL.eTRIGGERMODE.TM_SCAN
                ' Enable scanning by trigger

CONTROL.CommitChanges()
                ' Save Changes to the Scanner Configuration
```

SendScanStatus Codes

If you have enabled SendScanStatus in your General structures definitions, use the following table to interpret the data returned from every scan. This data precedes the bar code and includes the length of data and bar code type. See “General” in this chapter for information about enabling SendScanStatus. See Appendix A, “Sample Applications” for information on receiving the scan status and extracting the length and barcode type.

Value	Bar Code Type	Value	Bar Code Type
0	No Scan	22	Bookland EAN
1	Code 39	23	Coupon Code
2	Codabar	48	RSS 14
3	Code 128	49	RSS Limited
4	Discrete 2of5	50	RSS Expanded
5	IATA 2of5	72	UPC A with 2 Supplements
6	Interleaved 2of5	73	UPC E with 2 Supplements
7	Code 93	74	EAN 8 with 2 Supplements
8	UPC A	75	EAN 13 with 2 Supplements
9	UPC E	80	UPC E1 with 2 Supplements
10	EAN 8	136	UPC A with 5 Supplements
11	EAN 13	137	UPC E with 5 Supplements
14	MSI Plessey	138	EAN 8 with 5 Supplements
15	EAN 128	139	EAN 13 with 5 Supplements
16	UPC E1	144	UPC E1 with 5 Supplements
21	Trioptic Code 39		

SAMPLE APPLICATIONS

This chapter contains two sample applications written in both VB.NET and C#. The first application can be used as a demo. The second application shows how to use the scanning and printing functions.

VB.NET Demo Sample

```
'*************************************************************
'* Name: 6039 Demo Application (Demo VB)                     *
'* Version:     1.0.0                                        *
'* Date:        5/1/2006                                     *
'* Development                                               *
'* Environment: Microsoft Visual Studio .NET 2003            *
'* (Visual Basic.NET)                                        *
'* Company:     Paxar Americas, Inc                          *
'*              Copyright 2006                               *
'* Supply Size  2020 (2.0" x 2.0")                           *
'* Description:                                              *
'* Setup Barcode Symbologies for only UPCA in the            *
'* Form1_Load Event                                          *
'* Scan/Enter UPC and Print Label with UPCA barcode          *
'* References                                                *
'* Add Reference to Ultra.dll                                *
'*************************************************************
Imports System              ' For Strings
Imports System.Drawing
' provides access to GDI+ basic graphics functionality
Imports System.Collections
' collections of objects, ArrayList
Imports System.Windows.Forms    ' Forms
Imports Ultra               ' Needed for Scanning & Printing

Public Class Form1
    Inherits System.Windows.Forms.Form
    Friend WithEvents btnClear As System.Windows.Forms.Button
    Friend WithEvents btnEnter As System.Windows.Forms.Button
    Friend WithEvents txtUPC As System.Windows.Forms.TextBox
    Friend WithEvents lblScanEnterData As
System.Windows.Forms.Label
```

```
#Region " Windows Form Designer generated code "
   Public Sub New()
      MyBase.New()
' This call is required by the Windows Form Designer.
      InitializeComponent()
' Add any initialization after the InitializeComponent() call
   End Sub

'Form overrides dispose to clean up the component list.
   Protected Overloads Overrides Sub Dispose(ByVal disposing
As Boolean)
      MyBase.Dispose(disposing)
   End Sub

' NOTE: The following procedure is required by the Windows
' Form Designer
' It can be modified using the Windows Form Designer.
' Do not modify it using the code editor.
   Private Sub InitializeComponent()
      Me.btnClear = New System.Windows.Forms.Button
      Me.btnEnter = New System.Windows.Forms.Button
      Me.txtUPC = New System.Windows.Forms.TextBox
      Me.lblScanEnterData = New System.Windows.Forms.Label

' btnClear
    Me.btnClear.Location = New System.Drawing.Point(136, 120)
    Me.btnClear.Text = "Clear"

' btnEnter
    Me.btnEnter.Location = New System.Drawing.Point(40, 120)
    Me.btnEnter.Text = "Enter"

' txtUPC
    Me.txtUPC.Location = New System.Drawing.Point(32, 40)
    Me.txtUPC.MaxLength = 12
    Me.txtUPC.Size = New System.Drawing.Size(168, 22)
    Me.txtUPC.Text = ""

' lblScanEnterData
    Me.lblScanEnterData.Location = New
System.Drawing.Point(24, 16)
    Me.lblScanEnterData.Size = New System.Drawing.Size(136,
24)
    Me.lblScanEnterData.Text = "Scan/Enter Barcode:"
```

```
' Form1
    Me.ClientSize = New System.Drawing.Size(242, 272)
    Me.Controls.Add(Me.btnClear)
    Me.Controls.Add(Me.btnEnter)
    Me.Controls.Add(Me.txtUPC)
    Me.Controls.Add(Me.lblScanEnterData)
    Me.Text = "Demo"

   End Sub

   Public Shared Sub Main()
      Application.Run(New Form1)
   End Sub
#End Region
'*************************************************************
'* Form1_Load Event
'* Description:
'* Set App Name & Version
'* Enable/Disable Barcode Symbologies & Commit Changes(save
'* to ultra.cfg)
'*************************************************************
   Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
' ---Set Application Name & Version Info
      Dim rVersion As New Ultra.Version
' Instantiate Version class
      rVersion.AppName = "Demo VB"
' Application Name
      rVersion.AppVersion = "1.0"
' Application Version
' Setup Scanner Configuration Programatically
      Dim GENERAL As New Ultra.Scan.General
      GENERAL.Postamble = "\n\r"
' Append CR -- Auto-Enter

' CONTROL
      Dim CONTROL As New Ultra.Scan.Control

' Set TriggerMode (Activate scanner on trigger press/release)
      CONTROL.TriggerMode = CONTROL.eTriggerMode.TM_SCAN
```

```
' Set ScannerMode (Scanner is on until T/O or successful
' scan)
      CONTROL.ScannerMode = CONTROL.eSCANMODE.SOM_COMPATIBLE

' Set DataMode (application receives data is by the WM_CHAR
' standard method.)
      CONTROL.DataMode = CONTROL.eDataMode.DRM_WMCHAR

      CONTROL.SendScanStatus = False
' Don't Send Scan Status Message

      CONTROL.DisableAllCodes()
' Disable All barcodes
'*************************************************************
'* Enable UPCA Barcodes only
'*************************************************************
      Dim UPC As New Ultra.Scan.UPCEAN
' Instantiate UPCEAN class
      UPC.EnableUPCA = True     ' Enable UPCA
      UPC.XmitUPCACD = True     ' Xmit UPCA C/D
      UPC.XmitUPCAPre =
Ultra.Scan.UPCEAN.eXMITUPCPREAMBLE.XUP_SYSCHAR

' Ignores UPCA + 2, UPCA +5, EAN8 + 2, EAN8 +5, EAN13 +2 &
' EAN13 + 5 Symbologies
      UPC.Supplemental = UPC.eSupplemental.USM_IGNORE

' --Save Changes to the Scanner Configuration which builds a
' new Ultra.cfg
      CONTROL.CommitChanges()

' --Set focus to txtUPC text box
      txtUPC.Focus()
   End Sub
```

```
'*************************************************************
'* txtUPC_KeyDown Event
'* Description:
'* Handle the KeyDown event to determine the
'* type of character entered into the control.
'* if Enter is Pressed, execute the btnEnter_Click Event
'*************************************************************
   Private Sub txtUPC_KeyDown(ByVal sender As Object, ByVal e
As System.Windows.Forms.KeyEventArgs) Handles txtUPC.KeyDown
      If (e.KeyCode.ToString() = "Return") Then
         btnEnter_Click(sender, e)
      End If
   End Sub
'*************************************************************
'* btnEnter_Click Event
'* Description:
'* If length of data is 12 print label if not
'* display error message
'*************************************************************
   Private Sub btnEnter_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEnter.Click
      Dim fmtUPCA As String = "{F,1,A,R,E,200,200," & Chr(34)
& "UPCA" & Chr(34) & "|"
          & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "Demo
VB.NET" & Chr(34) & "|"
          & "B,1,12,F,25,28,1,4,100,7,L,0|}"

      Dim rPrint As New Print
' Instantiate Print class

      If (txtUPC.Text.Length <> 12) Then
' Make sure Length is 12

         MsgBox("Invalid Length", MsgBoxStyle.OKOnly, "Entry
error")

         txtUPC.Text = ""
' Reset content to nothing
         txtUPC.Focus()
' Focus on the text box
      Else
         rPrint.ClearError()
' Clear any errors
         If (Not rPrint.IsBatteryOKToPrint) Then
```

```
' Check Battery
   MsgBox("Low Battery", MsgBoxStyle.OKOnly, "Battery Check")
    Else
' --Print Format and then Batch Data
      rPrint.Text = fmtUPCA
      rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
      rPrint.Text = "1," & Chr(34) & txtUPC.Text & Chr(34) &
"|}"

' --Clear UPC and Set Focus
      txtUPC.Text = ""          ' Reset content to nothing
      txtUPC.Focus()            ' Focus on the text box
         End If
      End If

   End Sub
'*************************************************************
'* btnClear_Click Event
'* Description:
'* Erase data &
'* Set focus back to txtUPC
'*************************************************************
   Private Sub btnClear_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnClear.Click
      txtUPC.Text = ""                                       '
Erase any data
      txtUPC.Focus()                                         '
Put focus back on the txt box
   End Sub

End Class
```

C# Demo Sample

```
//'************************************************************
//*  Name: 6039 Demo Application (Demo CSharp)               *
//'* Version:     1.0.0                                      *
//'* Date:        5/1/2006                                   *
//'* Development                                             *
//'* Environment: Microsoft Visual Studio .NET 2003          *
//'* (C Sharp.NET)                                          *
//'* Company:     Paxar Americas, Inc                        *
//'*              Copyright 2006                             *
//'* Supply Size  2020 (2.0" x 2.0")                         *
//*  Description:                                           *
//*  Setup Barcode Symbologies in the Form1_Load Event      *
//*  Scan/Enter UPC and Print Label with UPCA barcode       *
//*  References                                             *
//*  Add Reference to Ultra.dll                             *
//************************************************************
using System;              // For Strings
using System.Drawing;
// provides access to GDI+ basic graphics
using System.Collections;
// collections of objects, ArrayList
using System.Windows.Forms;     // Forms
using Ultra;               // Needed for Scanning & Printing

namespace Demo_CSharp
{
/// <summary>
/// Summary description for Form1.
/// </summary>
   public class Form1 : System.Windows.Forms.Form
   {
      private System.Windows.Forms.Label lblScanEnterData;
      private System.Windows.Forms.Button btnEnter;
      private System.Windows.Forms.TextBox txtUPC;
      private System.Windows.Forms.Button btnClear;
```

```
      public Form1()
      {
// Required for Windows Form Designer support
         InitializeComponent();
// TODO: Add any constructor code after InitializeComponent
// call
      }
/// <summary>
/// Clean up any resources being used.
/// </summary>
      protected override void Dispose( bool disposing )
      {
         base.Dispose( disposing );
      }
      #region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
   private void InitializeComponent()
    {
    this.lblScanEnterData = new System.Windows.Forms.Label();
    this.txtUPC = new System.Windows.Forms.TextBox();
    this.btnEnter = new System.Windows.Forms.Button();
    this.btnClear = new System.Windows.Forms.Button();
// lblScanEnterData

   this.lblScanEnterData.Location = new
System.Drawing.Point(16, 16);
   this.lblScanEnterData.Size = new System.Drawing.Size(136,
24);
   this.lblScanEnterData.Text = "Scan/Enter Barcode:";

// txtUPC
   this.txtUPC.Location = new System.Drawing.Point(24, 40);
   this.txtUPC.MaxLength = 12;
   this.txtUPC.Size = new System.Drawing.Size(168, 22);
   this.txtUPC.Text = "";
   this.txtUPC.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.txtData_KeyDown);
```

```
// btnEnter
   this.btnEnter.Font = new System.Drawing.Font("Tahoma",
9.75F, System.Drawing.FontStyle.Regular);
   this.btnEnter.Location = new System.Drawing.Point(32, 126);
   this.btnEnter.Text = "Enter";
   this.btnEnter.Click += new
System.EventHandler(this.btnEnter_Click);

// btnClear
   this.btnClear.Font = new System.Drawing.Font("Tahoma",
9.75F, System.Drawing.FontStyle.Regular);
   this.btnClear.Location = new System.Drawing.Point(128,
128);
   this.btnClear.Text = "Clear";
   this.btnClear.Click += new
System.EventHandler(this.btnClear_Click);

// Form1
   this.ClientSize = new System.Drawing.Size(242, 272);
   this.Controls.Add(this.btnClear);
   this.Controls.Add(this.btnEnter);
   this.Controls.Add(this.txtUPC);
   this.Controls.Add(this.lblScanEnterData);
   this.MaximizeBox = false;
   this.MinimizeBox = false;
   this.Text = "Demo";
   this.WindowState =
System.Windows.Forms.FormWindowState.Maximized;
   this.Load += new System.EventHandler(this.Form1_Load);
    }
    #endregion

/// <summary>
/// The main entry point for the application.
/// </summary>

   static void Main()
    {
      Application.Run(new Form1());
    }
```

```
//************************************************************
//* Form1_Load Event
//* Description:
//*   Set App Name & Version
//* Enable/Disable Barcode Symbologies & Commit Changes(save
//* to ultra.cfg)
//************************************************************
   private void Form1_Load(object sender, System.EventArgs e)
      {
// ---Set Application Name & Version Info
         Ultra.Version rVersion = new Ultra.Version();
// Instantiate Version class
         rVersion.AppName = "Demo C Sharp";
// Application Name
         rVersion.AppVersion = "1.0";
// Application Version
// Setup Scanner Configuration Programatically
// ---Scanner Settings
      Ultra.Scan.General GENERAL = new Ultra.Scan.General();
      GENERAL.Postamble = "\\n\\r";
// Append CR -- Auto-Enter
      Ultra.Scan.Control CONTROL = new Ultra.Scan.Control();

// Set TriggerMode (Activate scanner on trigger press/release)
      CONTROL.TriggerMode =
Ultra.Scan.Control.eTRIGGERMODE.TM_SCAN;

// Set ScannerMode (Scanner is on until T/O or successful
// scan)
      CONTROL.ScannerMode =
Ultra.Scan.Control.eSCANMODE.SOM_COMPATIBLE;

// Set DataMode (application receives data is by the WM_CHAR
// standard method.)
      CONTROL.DataMode =
Ultra.Scan.Control.eDATAMODE.DRM_WMCHAR;

      CONTROL.SendScanStatus = false;
// Don't Send Scan Status Msg
      CONTROL.DisableAllCodes();
// Disable All barcodes
```

```
//*************************************************************
//* Enable UPCA Barcodes only
//*************************************************************
       Ultra.Scan.UPCEAN UPC = new Ultra.Scan.UPCEAN();
// Instantiate UPCEAN class
       UPC.EnableUPCA = true;   // Enable UPCA
       UPC.XmitUPCACD = true;   // Xmit UPCA C/D
       UPC.XmitUPCAPre =
Ultra.Scan.UPCEAN.eXMITUPCPREAMBLE.XUP_SYSCHAR;

// Ignores UPCA + 2, UPCA +5, EAN8 + 2, EAN8 +5, EAN13 +2 &
// EAN13 + 5 Symbologies
         UPC.Supplemental =
Ultra.Scan.UPCEAN.eSUPPLEMENTAL.USM_IGNORE;

// --Save Changes to the Scanner Configuration which builds a
// new Ultra.cfg
         CONTROL.CommitChanges();

// --Set focus to txtUPC text box
         txtUPC.Focus();
      }
//*************************************************************
//* txtData_KeyDown Event
//* Description:
//* Handle the KeyDown event to determine the
//* type of character entered into the control.
//* if Enter is Pressed, execute the btnEnter_Click Event
//*************************************************************
    private void txtData_KeyDown(object sender,
System.Windows.Forms.KeyEventArgs e)
    {
       if (e.KeyCode.ToString() == "Return")
          btnEnter_Click(sender, e);
    }
```

```
//*************************************************************
//* btnEnter_Click Event
//* Description:
//* If length of data is 12 print label if not
//* display error message
//*************************************************************
    private void btnEnter_Click(object sender,
System.EventArgs e)
    {
       string fmtUPCA = "{F,1,A,R,E,200,200,\"UPCA\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"Demo C Sharp\",1|" +
"B,1,12,F,25,28,1,4,100,7,L,0|}";

       if (txtUPC.Text.Length != 12)
// Make sure Length is 12
       {
         MessageBox.Show("Invalid Length", "Entry error",
MessageBoxButtons.OK,
            MessageBoxIcon.Hand,
MessageBoxDefaultButton.Button1);
         txtUPC.Text = "";      // Reset content to nothing
         txtUPC.Focus();        // Focus on the text box
      }
      else
      {
        Ultra.Print rPrint = new Print();
// Instantiate Print class
        rPrint.ClearError();    // Clear any errors
        if (!rPrint.IsBatteryOKToPrint)    // Check Battery
          MessageBox.Show("Low Battery", "Battery Check");

//--Print Format and then Batch Data
          rPrint.Text = fmtUPCA;
          rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" +
txtUPC.Text + "\"|}";

// --Clear UPC and Set Focus
          txtUPC.Text = "";     // Reset content to nothing
          txtUPC.Focus();       // Focus on the text box
         }
      }
```

```
//*************************************************************
//* btnClear_Click Event
//* Description:
//* Erase data &
//* Set focus back to txtUPC
//*************************************************************
    private void btnClear_Click(object sender,
System.EventArgs e)
    {
      txtUPC.Text = "";    // Erase any data
         txtUPC.Focus();   // Put focus back on the txt box
      }
   }
}
```

VB.NET Scan/Print Sample

```
'*************************************************************
'* Project:       Platinum Scan-N-Print Sample (VB.NET)      *
'* Version:       1.0.0                                      *
'* Date:          5/1/2006                                   *
'* Development                                               *
'* Environment: Microsoft Visual Studio .NET 2003            *
'* Company:           Paxar Americas, Inc                    *
'*                         Copyright 2006
*
'* Description:                                              *
'* Scan and Print Demo for the M6039 Ultra Platinum.         *
'* Files will be loaded into the \paxar folder on the        *
'* Platinum                                                  *
'* Notes:  Need to add References to:                        *
'*                        Microsoft.WindowsCE.Form           *
'*                             Ultra.dll                     *
'*************************************************************
Imports System.Drawing
Imports System.Collections
Imports System.Windows.Forms
Imports System.Runtime.InteropServices
Imports System.Data
Imports Ultra
```

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Public WithEvents label3 As System.Windows.Forms.Label
    Public WithEvents txtScanLength As
System.Windows.Forms.TextBox
    Public WithEvents txtScanType As
System.Windows.Forms.TextBox
    Public WithEvents label2 As System.Windows.Forms.Label
    Public WithEvents label1 As System.Windows.Forms.Label
    Public WithEvents txtScanData As
System.Windows.Forms.TextBox

' -- MyMessageWindow
    Private objMsgWnd As MyMessageWindow

#Region " Windows Form Designer generated code "
   Public Sub New()
     MyBase.New()
' This call is required by the Windows Form Designer.
     InitializeComponent()
' Add any initialization after the InitializeComponent() call
' --Add any initialization after the InitializeComponent()
' --call
' objMsgWnd = new MyMessageWindow(this);
    objMsgWnd = New MyMessageWindow
    objMsgWnd.SetForm(Me)
    End Sub

' Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing
As Boolean)
    MyBase.Dispose(disposing)
    End Sub
' NOTE: The following procedure is required by the Windows
' Form Designer
' It can be modified using the Windows Form Designer.
' Do not modify it using the code editor.
    Public WithEvents lblStatus As System.Windows.Forms.Label
    Private Sub InitializeComponent()
        Me.label3 = New System.Windows.Forms.Label
        Me.txtScanLength = New System.Windows.Forms.TextBox
        Me.txtScanType = New System.Windows.Forms.TextBox
        Me.label2 = New System.Windows.Forms.Label
        Me.label1 = New System.Windows.Forms.Label
        Me.txtScanData = New System.Windows.Forms.TextBox
        Me.lblStatus = New System.Windows.Forms.Label
```

```
'label3
  Me.label3.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
  Me.label3.Location = New System.Drawing.Point(8, 136)
  Me.label3.Size = New System.Drawing.Size(56, 16)
  Me.label3.Text = "Length"

' txtScanLength
  Me.txtScanLength.Location = New System.Drawing.Point(8, 152)
  Me.txtScanLength.Size = New System.Drawing.Size(88, 22)
  Me.txtScanLength.Text = ""

'txtScanType
  Me.txtScanType.Location = New System.Drawing.Point(8, 96)
  Me.txtScanType.Size = New System.Drawing.Size(208, 22)
  Me.txtScanType.Text = ""

' label2
  Me.label2.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
  Me.label2.Location = New System.Drawing.Point(8, 80)
  Me.label2.Size = New System.Drawing.Size(100, 16)
  Me.label2.Text = "Type"

' label1
  Me.label1.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
  Me.label1.Location = New System.Drawing.Point(8, 24)
  Me.label1.Size = New System.Drawing.Size(100, 16)
  Me.label1.Text = "Scan Barcode"

' txtScanData
  Me.txtScanData.Location = New System.Drawing.Point(8, 40)
  Me.txtScanData.Multiline = True
  Me.txtScanData.Size = New System.Drawing.Size(208, 20)
  Me.txtScanData.Text = ""

' lblStatus
  Me.lblStatus.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Regular)
  Me.lblStatus.Location = New System.Drawing.Point(8, 216)
  Me.lblStatus.Size = New System.Drawing.Size(224, 24)
  Me.lblStatus.TextAlign =
System.Drawing.ContentAlignment.TopCenter
```

```
' Form1
   Me.ClientSize = New System.Drawing.Size(242, 272)
   Me.Controls.Add(Me.lblStatus)
   Me.Controls.Add(Me.label3)
   Me.Controls.Add(Me.txtScanLength)
   Me.Controls.Add(Me.txtScanType)
   Me.Controls.Add(Me.label2)
   Me.Controls.Add(Me.label1)
   Me.Controls.Add(Me.txtScanData)
   Me.Text = "Scan & Print Sample"

    End Sub
    Public Shared Sub Main()
        Application.Run(New Form1)
    End Sub
#End Region

'************************************************************
'* Form1_Load Event
'* Form Load Event - Enable Barcode Symbologies
'*      Initialize Text Fields & Set Focus
'************************************************************
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load

' --Display Status to user
      lblStatus.Text = "Please Wait..."

' --Change Cursor to Hour Glass
      Cursor.Current = Cursors.WaitCursor

' --Refresh the Screen
      Me.Refresh()
'************************************************************
' Setup Scanner Configuration Programatically
'************************************************************
' CONTROL
    Dim Scanner As New Ultra.Scan.Control

' Set ScannerMode (Scanner is on until T/O or successful scan)
    Scanner.ScannerMode = Scanner.eSCANMODE.SOM_COMPATIBLE
'Set DataMode (Application receives data from the scanner)
    Scanner.DataMode = Scanner.eDataMode.DRM_SMSCANCHAR
```

```
' Set TriggerMode (Activate scanner on trigger press/release)
    Scanner.TriggerMode = Scanner.eTriggerMode.TM_SCAN

' --Enable Send Scan Status
    Scanner.SendScanStatus = True
' Enable Send Scan Status
'*************************************************************
'* UPCA Barcodes
'*************************************************************
    Dim UPC As New Ultra.Scan.UPCEAN
    UPC.EnableUPCA = True       ' Enable UPCA
    UPC.XmitUPCACD = True       ' Xmit UPCA C/D
    UPC.XmitUPCAPre = UPC.XmitUPCAPre.XUP_SYSCHAR

' To Scan/Print UPCE Barcodes, UPCE Preamble must be set to
' NONE
      UPC.EnableUPCE = True     ' Enable UPCE
      UPC.XmitUPCECD = True     ' Xmit UPCE C/D
      UPC.XmitUPCEPre = UPC.XmitUPCEPre.XUP_NONE

      UPC.EnableUPCE1 = True    ' Enable UPCE1
      UPC.XmitUPCE1CD = True    ' Xmit UPCE1 C/D
      UPC.XmitUPCE1Pre = UPC.XmitUPCE1Pre.XUP_SYSCHAR

      UPC.EnableEAN8 = True     ' Enable EAN8
      UPC.EnableEAN13 = True    ' Enable EAN13
      UPC.CouponCode = True     ' Enable Coupon Code
      UPC.EnableEANBkld = True  ' Enable Bookland
      UPC.Xlate8To13 = False    ' Don't Translate 8 to 13
      UPC.XlateE1ToA = False
' Don't Translate UPCE1 to UPCA
      UPC.XlateEToA = False
' Don't Translate UPCE to UPCA
      UPC.EANZeroExtend = False
' Don't use EAN Zero Extend
      UPC.SupRedundancy = 7
' Set Supplemental Redundancy = 7

' Enables UPCA + 2, UPCA +5, EAN8 + 2, EAN8 +5, EAN13 +2 &
' EAN13 + 5 Symbologies
      UPC.Supplemental = UPC.eSupplemental.USM_AUTO
```

```
'*************************************************************
'* Code 39 Barcodes
'*************************************************************
    Dim C39 As New Ultra.Scan.Code39
    C39.Enable = True           ' Enable Code 39
    C39.FullASCII = True        ' Enable Full Ascii
    C39.XmitCD = True           ' Xmit C/D
    C39.VerifyCD = False        ' Don't Verify C/Ds
    C39.Trioptic = False   ' Disable Triopic Code 39
    C39.Code32Prefix = False    ' No CODE32 prefix
    C39.XlateToCode32 = False   ' Don't Convert to CODE32
    C39.FixedLength = False     ' Variable Length Enabled
    C39.Length1 = 2             ' Variable Length1 = 2
    C39.Length2 = 55       ' Variable Length2 = 55
'*************************************************************
'* Code 128 Barcodes
'*************************************************************
    Dim C128 As New Ultra.Scan.Code128
    C128.Enable = True          ' Enable Code 128
    C128.UCCEAN128 = True       ' Enable UCCEAN128
    C128.ISBT128 = True         ' Enable ISBT128
'*************************************************************
'* Interleaved 2 of 5 Barcodes
'*************************************************************
    Dim I2OF5 As New Ultra.Scan.I2of5
    I2OF5.Enable = True         ' Enable I 2of5 Symbology
    I2OF5.VerifyCD = False      ' Don't Verify C/Ds
    I2OF5.XmitCD = False        ' Don't Xmit C/Ds
    I2OF5.FixedLength = False   ' Variable Length Enabled
    I2OF5.Length1 = 2           ' Variable Length1 = 2
    I2OF5.Length2 = 22          ' Variable Length2 = 22
'*************************************************************
'* Discrete 2 of 5 Barcodes
'*************************************************************
    Dim D2OF5 As New Ultra.Scan.D2of5
    D2OF5.Enable = True         ' Enable D 2of 5
    D2OF5.FixedLength = False   ' Variable Length Enabled
    D2OF5.Length1 = 2           ' Variable Length1 = 2
    D2OF5.Length2 = 22          ' Variable Length2 = 22
```

```
'*************************************************************
'* MSI Plessey Barcodes
'*************************************************************
  Dim MSI As New Ultra.Scan.MSI
  MSI.Enable = True                    ' Enable MSI Barcodes
  MSI.Use2CDs = False                  ' Don't use 2 C/Ds
  MSI.UseMod10Mod11CDAlg = False       ' Don't use Mod 10
  MSI.XmitCD = True               ' Xmit C/D
  MSI.FixedLength = False         ' Variable Length Enabled
  MSI.Length1 = 1                 ' Variable Length1 = 1
  MSI.Length2 = 55                ' Variable Length2 = 55

'*************************************************************
'* Code 93 Barcodes
'*************************************************************
  Dim C93 As New Ultra.Scan.Code93
  C93.Enable = True               ' Enable Code 93 Symbology
  C93.FixedLength = False         ' Variable Length Enabled
  C93.Length1 = 2                 ' Variable Length1 = 2
  C93.Length2 = 55                ' Variable Length2 = 55
'*************************************************************
'* Codabar Barcodes
'*************************************************************
  Dim CODABAR As New Ultra.Scan.Codabar
  CODABAR.Enable = True           ' Enable Codabar Symbology
  CODABAR.FixedLength = False     ' Variable Length Enabled
  CODABAR.Length1 = 1             ' Variable Length1 = 1
  CODABAR.Length2 = 55            ' Variable Length2 = 55
'*************************************************************
'* General Settings
'*************************************************************
  Dim GENERAL As New Ultra.Scan.General
  GENERAL.Preamble = ""           ' Clear any preambles
  GENERAL.Postamble = ""          ' Clear any postambles
  GENERAL.Timeout = 30            ' No Scan T/O = 30tenths/sec
  GENERAL.AimDuration = 0         ' Aim Duration
  GENERAL.BdirRedundancy = False
' Disable BiDirection Redundancy
  GENERAL.LinearSecurity = 1
' Set Linear Security = 1
  GENERAL.GoodScanWav = "\Windows\goodscan.wav"
' Set GoodScan WAV File
  GENERAL.NoReadWav = "\Windows\noread.wav"
' Set NoREAD WAV File
```

```
' Save Changes to the Scanner Configuration which builds a
' new Ultra.cfg
  Scanner.CommitChanges()

  lblStatus.Text = ""      ' Initialize Status Field
  txtScanData.Text = ""    ' Initialize Barcode Data field
  txtScanType.Text = ""    ' Initialize Barcode Type Field
  txtScanLength.Text = ""  ' Initialize Barcode Length field
  txtScanData.Focus()      ' Set focus to Scan Data field
  Cursor.Current = Cursors.Default
' Change Cursor Back to Default
  lblStatus.Text = "Ready" ' Inform user it is ready
  Me.Refresh()             ' Refresh the Screen
 End Sub
End Class

Public Class MyMessageWindow
    Inherits Microsoft.WindowsCE.Forms.MessageWindow

' Message IDs received from scanner

   Public Const SM_SCANCHAR As Integer = &H3000
' incomming scanner char message
   Public Const SM_SCANSTATUS As Integer = &H3001
' incomming scanner status message
   Public count As Integer = 0       ' initialize count

   Public Const STR_SCANMSG As String = "Paxar Scanner
Interface"

   Public strScanData As String
' Class based variables
   Public unScanMsgID As Integer
' Message ID to use for broadcast
    Public objForm As Form1

    <DllImport("coredll", SetLastError:=True)> _
    Public Shared Function RegisterWindowMessage(ByVal
lpMessage As String) As Integer
    End Function
     Public Sub SetForm(ByVal myForm As Form1)
        objForm = myForm
        unScanMsgID = RegisterWindowMessage(STR_SCANMSG)
    End Sub
```

```
'*************************************************************
'* WndProc
'* Description
'*************************************************************
  Protected Overrides Sub WndProc( _
  ByRef msg As Microsoft.WindowsCE.Forms.Message)

      Dim nBarType As Integer = 0

      If (msg.Msg = unScanMsgID) Then
          If (msg.WParam.ToInt32() = SM_SCANCHAR) Then
              objForm.txtScanData.Text +=
Convert.ToChar(msg.LParam.ToInt32())
              count = count - 1

              If (count = 0) Then
                  strScanData = objForm.txtScanData.Text

                  '--Print a Label
                  PrintLabel()
              End If 'If (count = 0)
          End If 'If (msg.WParam.ToInt32

          If (msg.WParam.ToInt32() = SM_SCANSTATUS) Or
(msg.Msg = 12288) Then
              nBarType = (msg.LParam.ToInt32() >> 16) '&
&HFFFF
              count = ((msg.LParam.ToInt32() << 16) >> 16)
' clear out the high 2 bytes by shifting right then left

              objForm.txtScanLength.Text = "" &
count.ToString()

              If (objForm.txtScanLength.Text = "-165535") Then
                  objForm.txtScanLength.Text = ""
                  objForm.txtScanType.Text = "No Scan"
              End If 'If (objForm.txtScanLength.Text = "-
165535")
```

```
Select Case nBarType
  Case -1
    objForm.txtScanType.Text = "No Scan"
    objForm.txtScanLength.Text = "0"
  Case 0
    objForm.txtScanType.Text = "Unknown"
  Case 1
    objForm.txtScanType.Text = "Code 39"
  Case 2
    objForm.txtScanType.Text = "Codabar"
  Case 3
    objForm.txtScanType.Text = "Code 128"
  Case 4
    objForm.txtScanType.Text = "Discrete 2 of 5"
  Case 5
    objForm.txtScanType.Text = "IATA 2 of 5"
  Case 6
    objForm.txtScanType.Text = "Interleaved 2 of 5"
  Case 7
    objForm.txtScanType.Text = "Code 93"
  Case 8
    objForm.txtScanType.Text = "UPC A"
  Case 9
    objForm.txtScanType.Text = "UPC E"
  Case 10
    objForm.txtScanType.Text = "EAN 8"
  Case 11
    objForm.txtScanType.Text = "EAN 13"
  Case 14
    objForm.txtScanType.Text = "MSI Plessey"
  Case 15
    objForm.txtScanType.Text = "EAN 128"
  Case 16
    objForm.txtScanType.Text = "UPC E1"
  Case 21
    objForm.txtScanType.Text = "Trioptic Code 39"
  Case 22
    objForm.txtScanType.Text = "Bookland EAN"
  Case 23
    objForm.txtScanType.Text = "Coupon Code"
  Case 48
    objForm.txtScanType.Text = "RSS-14"
  Case 49
    objForm.txtScanType.Text = "RSS-Limited"
  Case 50
  objForm.txtScanType.Text = "RSS-Expanded"
```

```
          Case 72
            objForm.txtScanType.Text = "UPC A + 2"
          Case 73
            objForm.txtScanType.Text = "UPC E + 2"
          Case 74
            objForm.txtScanType.Text = "EAN 8 + 2"
          Case 75
            objForm.txtScanType.Text = "EAN 13 + 2"
          Case 80
            objForm.txtScanType.Text = "UPC E1 + 2"
          Case 136
            objForm.txtScanType.Text = "UPC A + 5"
          Case 137
            objForm.txtScanType.Text = "UPC E + 5"
          Case 138
            objForm.txtScanType.Text = "EAN 8 + 5"
          Case 139
            objForm.txtScanType.Text = "EAN 13 + 5"
          Case 144
            objForm.txtScanType.Text = "UPC E1 + 5"
          End Select

            objForm.txtScanData.Text = ""

            End If 'If (msg.WParam.ToInt32() = SM_SCANSTATUS)


      End If 'If (msg.Msg = unScanMsgID) Then
' -- Call the base class WndProc for default message handling
' base.WndProc(ref msg)
      MyBase.WndProc(msg)
   End Sub
'*************************************************************
'* PrintLabel
'* Description
'* After a good scan, Print Label
'*************************************************************
    Public Sub PrintLabel()
      Dim strFCN1 As String = "~201"
' Code 128 Function Code F1
      Dim strType As String
```

```
' Barcode Type

      Dim rPrint As New Ultra.Print

      Const fmtC39 As String = "{F,1,A,R,E,200,200," & Chr(34)
& "C39" & Chr(34) & "|" _
                & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) &
"M6039 Platinum" & Chr(34) & "|" _
                & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
                & "B,2,20,V,23,2,4,12,100,8,C,0|" _
                & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtCodabar As String = "{F,1,A,R,E,200,200," &
Chr(34) & "CODABAR" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,18,V,23,100,5,8,100,8,B,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtC128 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "C128" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,22,V,23,100,8,8,100,8,B,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtI2OF5 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "I2OF5" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,22,V,23,11,3,13,100,8,C,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtC93 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "C93" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,18,V,23,100,23,10,100,8,B,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"
```

```
        Const fmtUPCA As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCA" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,12,F,25,28,1,4,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtUPCA2 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCA2" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,14,F,25,38,10,2,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtUPCA5 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCA5" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,17,F,25,28,11,2,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtUPCE As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCE" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,7,V,25,56,2,4,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN8 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN8" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,8,V,25,42,6,4,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN82 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN82" & Chr(34) & "|"
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|"
            & "B,1,10,V,25,25,14,4,100,7,L,0|"
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"
```

```
        Const fmtEAN85 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN85" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,13,V,25,40,15,2,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN13 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN13" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,13,V,25,24,7,4,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN132 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN132" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,15,V,25,40,16,2,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN135 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN135" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,18,V,25,25,17,2,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtMSI As String = "{F,1,A,R,E,200,200," &
Chr(34) & "MSI" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,11,V,25,100,9,7,100,8,B,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtEAN128 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "EAN128" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "T,1,22,V,6,1,0,1,1,1,O,C,0,0|" _
            & "B,2,22,V,23,100,8,8,100,8,B,0|" _
            & "T,3,22,V,133,1,0,1,1,1,O,C,0,0|}"
```

```
        Const fmtUPCE2 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCE2" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,9,V,25,35,12,4,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        Const fmtUPCE5 As String = "{F,1,A,R,E,200,200," &
Chr(34) & "UPCE5" & Chr(34) & "|" _
            & "C,150,49,0,50,8,8,A,L,0,0," & Chr(34) & "M6039
Platinum" & Chr(34) & "|" _
            & "B,1,12,V,25,20,13,4,100,7,L,0|" _
            & "T,2,22,V,133,1,0,1,1,1,O,C,0,0|}"

        strType = objForm.txtScanType.Text

' --Clear any errors
      rPrint.ClearError()

' Print Format & Batch
  Select Case (strType)
  Case "Code 39"
    rPrint.Text = fmtC39
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "Codabar"
    rPrint.Text = fmtCodabar
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "Code 128"
    rPrint.Text = fmtC128
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
```

```
  Case "Discrete 2 of 5"
    objForm.lblStatus.Text = "D2of5 not available to print"

  Case "IATA 2 of 5"
    objForm.lblStatus.Text = "IATA 2of5 not available to
print"

  Case "Interleaved 2 of 5"
    rPrint.Text = fmtI2OF5
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "Code 93"
    rPrint.Text = fmtC93
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "UPC A"
    rPrint.Text = fmtUPCA
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "UPC E"
    rPrint.Text = fmtUPCE
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "EAN 8"
    rPrint.Text = fmtEAN8
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
```

```
  Case "EAN 13"
    rPrint.Text = fmtEAN13
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "MSI Plessey"
    rPrint.Text = fmtMSI
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
  Case "EAN 128"
    rPrint.Text = fmtEAN128
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strFCN1 & strScanData &
Chr(34) & "|"
    rPrint.Text = "3" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
  Case "UPC E1"
    objForm.lblStatus.Text = "UPC E1 not available to print"

  Case "Trioptic Code 39"
    objForm.lblStatus.Text = "Trioptic Code 39 not available
to print"
  Case "Bookland EAN"
    objForm.lblStatus.Text = "Bookland EAN not available to
print"

  Case "Coupon Code"
    objForm.lblStatus.Text = "Coupon Code not available to
print"

  Case "RSS-14"
    objForm.lblStatus.Text = "RSS-14 not available to print"

  Case "RSS-Limited"
    objForm.lblStatus.Text = "RSS-Limited not available to
print"

  Case "RSS-Expanded"
    objForm.lblStatus.Text = "RSS-Expanded not available to
print"
```

```
  Case "UPC A + 2"
    rPrint.Text = fmtUPCA2
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "UPC E + 2"
    rPrint.Text = fmtUPCE2
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "EAN 8 + 2"
    rPrint.Text = fmtEAN82
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "EAN 13 + 2"
    rPrint.Text = fmtEAN132
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
  Case "UPC E1 + 2"
    objForm.lblStatus.Text = "UPC E1 + 2 not available to
print"

  Case "UPC A + 5"
    rPrint.Text = fmtUPCA5
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""
```

```
  Case "UPC E + 5"
    rPrint.Text = fmtUPCE5
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "EAN 8 + 5"
    rPrint.Text = fmtEAN85
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "EAN 13 + 5"
    rPrint.Text = fmtEAN135
    rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|"
    rPrint.Text = "1," & Chr(34) & strScanData & Chr(34) & "|"
    rPrint.Text = "2" & Chr(34) & strType & Chr(34) & "|}"
    objForm.lblStatus.Text = ""

  Case "UPC E1 + 5"
    objForm.lblStatus.Text = "UPC E1 + 5 not available to
print"

  Case Else
    End Select

   End Sub
End Class
```

C# Scan/Print Sample

```
//'************************************************************
//'* Project:     Platinum Scan-N-Print Sample (CSharp.NET) *
//'* Version:     1.0.0                                      *
//'* Date:        5/1/2006                                   *
//'* Development                                             *
//'* Environment: Microsoft Visual Studio .NET 2003          *
//'* (C Sharp.NET)                                          *
//'* Company:     Paxar Americas, Inc                        *
//'*              Copyright 2006                             *
//'* Supply Size  2020 (2.0" x 2.0")                         *
//'* Description:                                           *
//'* Scan and Print Demo for the M6039 Ultra Platinum.       *
//'* Files will be loaded into the \paxar folder on the      *
//'* Platinum                                               *
//'* Notes: Need to add References to:                      *
//'*            Microsoft.WindowsCE.Form                    *
//'*            Ultra.dll                                   *
//'************************************************************
using System;                    // For Strings
using System.Drawing;
// provides access to GDI+ basic graphics
using System.Collections;
// collections of objects, ArrayList
using System.Windows.Forms;      // Forms
using System.Data;
using System.Runtime.InteropServices;
// Needed for RegisterWindowMsg
using Ultra;
// Needed for Scanning & Printing
using System.IO;

namespace ScanNPrintCS
```

```
{
/// <summary>
/// Summary description for Form1.
/// </summary>
   public class Form1 : System.Windows.Forms.Form
   {
      public System.Windows.Forms.Label lblLength;
      public System.Windows.Forms.TextBox txtScanLength;
      public System.Windows.Forms.TextBox txtScanType;
      public System.Windows.Forms.Label lblType;
      public System.Windows.Forms.Label lblScanBarcode;
      public System.Windows.Forms.TextBox txtScanData;
      public System.Windows.Forms.Label lblStatus;

      private MyMessageWindow objMsgWnd;

      public Form1()
      {
// Required for Windows Form Designer support
         InitializeComponent();

// TODO: Add any constructor code after InitializeComponent
// call
         objMsgWnd = new MyMessageWindow(this);
      }
/// <summary>
/// Clean up any resources being used.
/// </summary>
      protected override void Dispose( bool disposing )
      {
         base.Dispose( disposing );
      }
```

```
      #region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
    private void InitializeComponent()
    {
      this.lblLength = new System.Windows.Forms.Label();
      this.txtScanLength = new System.Windows.Forms.TextBox();
      this.txtScanType = new System.Windows.Forms.TextBox();
      this.lblType = new System.Windows.Forms.Label();
      this.lblScanBarcode = new System.Windows.Forms.Label();
      this.txtScanData = new System.Windows.Forms.TextBox();
      this.lblStatus = new System.Windows.Forms.Label();

// lblLength
      this.lblLength.Font = new System.Drawing.Font("Tahoma",
9F, System.Drawing.FontStyle.Bold);
      this.lblLength.Location = new System.Drawing.Point(8,
136);
      this.lblLength.Size = new System.Drawing.Size(56, 16);
      this.lblLength.Text = "Length";

// txtScanLength
      this.txtScanLength.Location = new
System.Drawing.Point(8, 152);
      this.txtScanLength.Size = new System.Drawing.Size(88,
22);
     this.txtScanLength.Text = "";

// txtScanType
     this.txtScanType.Location = new System.Drawing.Point(8,
96);
     this.txtScanType.Size = new System.Drawing.Size(208, 22);
     this.txtScanType.Text = "";

// lblType
     this.lblType.Font = new System.Drawing.Font("Tahoma", 9F,
System.Drawing.FontStyle.Bold);
     this.lblType.Location = new System.Drawing.Point(8, 80);
     this.lblType.Size = new System.Drawing.Size(100, 16);
     this.lblType.Text = "Type";
```

```
// lblScanBarcode
     this.lblScanBarcode.Font = new
System.Drawing.Font("Tahoma", 9F,
System.Drawing.FontStyle.Bold);
     this.lblScanBarcode.Location = new
System.Drawing.Point(8, 24);
     this.lblScanBarcode.Size = new System.Drawing.Size(100,
16);
     this.lblScanBarcode.Text = "Scan Barcode";

// txtScanData
     this.txtScanData.Location = new System.Drawing.Point(8,
40);
     this.txtScanData.Multiline = true;
     this.txtScanData.Size = new System.Drawing.Size(208, 20);
     this.txtScanData.Text = "";

// lblStatus
     this.lblStatus.Location = new System.Drawing.Point(8,
216);
     this.lblStatus.Size = new System.Drawing.Size(224, 24);
     this.lblStatus.TextAlign =
System.Drawing.ContentAlignment.TopCenter;

// Form1
     this.ClientSize = new System.Drawing.Size(242, 272);
     this.Controls.Add(this.lblStatus);
     this.Controls.Add(this.lblLength);
     this.Controls.Add(this.txtScanLength);
     this.Controls.Add(this.txtScanType);
     this.Controls.Add(this.lblType);
     this.Controls.Add(this.lblScanBarcode);
     this.Controls.Add(this.txtScanData);
     this.MaximizeBox = false;
     this.MinimizeBox = false;
     this.Text = "Scan & Print Sample";
     this.WindowState =
System.Windows.Forms.FormWindowState.Maximized;
     this.Load += new System.EventHandler(this.Form1_Load);
    }
      #endregion
```

```
/// <summary>
/// The main entry point for the application.
/// </summary>

   static void Main()
   {
     Application.Run(new Form1());
   }

//*************************************************************
//* Form1_Load Event
//* Description
//*      Form Load Event - Enable Barcode Symbologies
//*      Initialize Text Fields & Set Focus
//*************************************************************
   private void Form1_Load(object sender, System.EventArgs e)
   {
// --Display Status to user
     lblStatus.Text = "Please Wait...";

// --Change Cursor to Hour Glass
      Cursor.Current = Cursors.WaitCursor;
//--Refresh the Screen
      this.Refresh();
//*************************************************************
// Setup Scanner Configuration Programatically
//*************************************************************
// Instantiate Control class
Ultra.Scan.Control Scanner = new Ultra.Scan.Control();

// Set ScannerMode (Scanner is on until T/O or successful
// scan)
   Scanner.ScannerMode =
Ultra.Scan.Control.eSCANMODE.SOM_COMPATIBLE;

// Set DataMode (Application receives data from the scanner)
   Scanner.DataMode =
Ultra.Scan.Control.eDATAMODE.DRM_SMSCANCHAR;

// Set TriggerMode (Activate scanner on trigger press/release)
   Scanner.TriggerMode =
Ultra.Scan.Control.eTRIGGERMODE.TM_SCAN;

// Send Scan Status messages to the application
   Scanner.SendScanStatus = true;
// Enable Send Scan Status
```

```
//************************************************************
//* UPCA Barcodes
//************************************************************
    Ultra.Scan.UPCEAN UPC = new Ultra.Scan.UPCEAN();
// Instantiate UPCEAN class
    UPC.EnableUPCA = true;      // Enable UPCA
    UPC.XmitUPCACD = true;      // Xmit UPCA C/D
    UPC.XmitUPCAPre =
Ultra.Scan.UPCEAN.eXMITUPCPREAMBLE.XUP_SYSCHAR;

// --To Scan/Print UPCE Barcodes, UPCE Preamble must be set to
// NONE
    UPC.EnableUPCE = true;      // Enable UPCE
    UPC.XmitUPCECD = true;      // Xmit UPCE C/D
    UPC.XmitUPCEPre =
Ultra.Scan.UPCEAN.eXMITUPCPREAMBLE.XUP_NONE;
    UPC.EnableUPCE1 = true;     // Enable UPCE1
    UPC.XmitUPCE1CD = true;     // Xmit UPCE1 C/D
    UPC.XmitUPCE1Pre =
Ultra.Scan.UPCEAN.eXMITUPCPREAMBLE.XUP_SYSCHAR;
    UPC.EnableEAN8 = true;      // Enable EAN8
    UPC.EnableEAN13 = true;     // Enable EAN13
    UPC.CouponCode = true;      // Enable Coupon Code
    UPC.EnableEANBkld = true;   // Enable Bookland
    UPC.Xlate8To13 = false;     // Don't Translate 8 to 13
    UPC.XlateE1ToA = false;  // Don't Translate UPCE1 to UPCA
    UPC.XlateEToA = false;   // Don't Translate UPCE to UPCA
    UPC.EANZeroExtend = false;
// Don't use EAN Zero Extend
    UPC.SupRedundancy = 7;
// Set Supplemental Redundancy to 7
// Enables UPCA + 2, UPCA +5, EAN8 + 2, EAN8 +5, EAN13 +2 &
// EAN13 + 5 Symbologies
   UPC.Supplemental =
Ultra.Scan.UPCEAN.eSUPPLEMENTAL.USM_AUTO;
//************************************************************
//* Code 39 Barcodes
//************************************************************
   Ultra.Scan.Code39 C39 = new Ultra.Scan.Code39();
// Instantiate Code39 class
   C39.Enable = true;
// Enable Code 39
   C39.FullASCII = true;
// Enable Full Ascii
   C39.XmitCD = true;          // Xmit C/D
```

```
   C39.VerifyCD = false;         // Don't Verify C/Ds
   C39.Trioptic = false;         // Disable Triopic Code 39
   C39.Code32Prefix = false;     // No CODE32 prefix
   C39.XlateToCode32 = false;    // Don't Convert to CODE32
   C39.FixedLength = false;      // Variable Length Enabled
   C39.Length1 = 2;              // Variable Length1 = 2
   C39.Length2 = 55;             // Variable Length2 = 55
//************************************************************
//* Code 128 Barcodes
//************************************************************
   Ultra.Scan.Code128 C128 = new Ultra.Scan.Code128();
// Instantiate Code128 class
   C128.Enable = true;           // Enable Code 128
   C128.UCCEAN128 = true;        // Enable UCCEAN128
   C128.ISBT128 = true;          // Enable ISBT128
//************************************************************
//* Interleaved 2 of 5 Barcodes
//************************************************************
   Ultra.Scan.I2of5 I2OF5 = new Ultra.Scan.I2of5();
// Instantiate I2of5 class
   I2OF5.Enable = true;          // Enable I 2of5 Symbology
   I2OF5.VerifyCD = false;       // Don't Verify C/Ds
   I2OF5.XmitCD = false;         // Don't Xmit C/Ds
   I2OF5.FixedLength = false;    // Variable Length Enabled
   I2OF5.Length1 = 2;            // Variable Length1 = 2
   I2OF5.Length2 = 22;           // Variable Length2 = 22
//************************************************************
//* Discrete 2 of 5 Barcodes
//************************************************************
   Ultra.Scan.D2of5 D2OF5 = new  Ultra.Scan.D2of5();
// Instantiate I2of5 class
   D2OF5.Enable = true;          // Enable D 2of 5
   D2OF5.FixedLength = false;    // Variable Length Enabled
   D2OF5.Length1 = 2;            // Variable Length1 = 2
   D2OF5.Length2 = 22;           // Variable Length2 = 22
//************************************************************
//* MSI Plessey Barcodes
//************************************************************
   Ultra.Scan.MSI MSI = new Ultra.Scan.MSI();
// Instantiate MSI class
   MSI.Enable = true;            // Enable MSI Barcodes
   MSI.Use2CDs = false;          // Don't use 2 C/Ds
   MSI.UseMod10Mod11CDAlg = false;    // Don't use Mod 10
   MSI.XmitCD = true;                 // Xmit C/D
   MSI.FixedLength = false;      // Variable Length Enabled
```

```
   MSI.Length1 = 1;              // Variable Length1 = 1
   MSI.Length2 = 55;             // Variable Length2 = 55
//************************************************************
//* Code 93 Barcodes
//************************************************************
   Ultra.Scan.Code93 C93 = new Ultra.Scan.Code93();
// Instantiate Code 93 class
   C93.Enable = true;            // Enable Code 93 Symbology
   C93.FixedLength = false;      // Variable Length Enabled
   C93.Length1 = 2;              // Variable Length1 = 2
   C93.Length2 = 55;             // Variable Length2 = 55
//************************************************************
//* Codabar Barcodes
//************************************************************
   Ultra.Scan.Codabar CODABAR = new Ultra.Scan.Codabar();
// Instantiate Codabar class
   CODABAR.Enable = true;        // Enable Codabar Symbology
   CODABAR.FixedLength = false; // Variable Length Enabled
   CODABAR.Length1 = 1;          // Variable Length1 = 1
   CODABAR.Length2 = 55;         // Variable Length2 = 55
//************************************************************
//* General Settings
//************************************************************
   Ultra.Scan.General GENERAL = new Ultra.Scan.General();
// Instantiate General class
   GENERAL.Preamble = "";        // Clear any preambles
   GENERAL.Postamble = "";       // Clear any postambles
   GENERAL.Timeout = 30;         // No Scan T/O = 30tenths/sec
   GENERAL.AimDuration = 0;      // Aim Duration
   GENERAL.BdirRedundancy = false;
// Disable BiDirection Redundancy
   GENERAL.LinearSecurity = 1;  // Set Linear Security = 1
   GENERAL.GoodScanWav = "\\Windows\\goodscan.wav";
// Set GoodScan WAV File
   GENERAL.NoReadWav = "\\Windows\\noread.wav";
// Set NoREAD WAV File

// Save Changes to the Scanner Configuration which builds a
// new Ultra.cfg
   Scanner.CommitChanges();
```

```
   lblStatus.Text = "";    // Initialize Status Field
   txtScanData.Text = "";
// Initialize Barcode Data field
   txtScanType.Text = "";
// Initialize Barcode Type Field
   txtScanLength.Text = "";
// Initialize Barcode Length field
   txtScanData.Focus();    // Set focus to Scan Data field
   Cursor.Current = Cursors.Default;
// Change Cursor Back to Default
   lblStatus.Text = "Ready";    // Inform user it is ready
   this.Refresh();         // Refresh the Screen
      }
   } // public class Form1 : System.Windows.Forms.Form

   public class MyMessageWindow :
Microsoft.WindowsCE.Forms.MessageWindow
   {

// Message IDs received from scanner

   public const UInt32 SM_SCANCHAR     = 0x3000;
// incomming scanner char msg
   public const UInt32 SM_SCANSTATUS   = 0x3001;
// incomming scanner status msg
   public int count = 0;
// initialize count

   public const string STR_SCANMSG     = "Paxar Scanner
Interface";

   public string strScanData;

// Class based variables

   private UInt32 unScanMsgID;
// Message ID to use for broadcast
   private Form1  objForm;

   [DllImport("coredll.dll", SetLastError = true)]
   private static extern int RegisterWindowMessage(string
strMessage);
```

```
//*************************************************************
//* MyMessageWindow
//* Description
//*************************************************************
   public MyMessageWindow(Form1 myForm)
    {
      objForm = myForm;
      unScanMsgID =
(UInt32)RegisterWindowMessage(STR_SCANMSG);
    }
//*************************************************************
//* WndProc
//* Description
//*************************************************************
   protected override void WndProc(ref
Microsoft.WindowsCE.Forms.Message msg)
   {
   int   nBarType = 0;

   if (msg.Msg == unScanMsgID)
    {
     if (msg.WParam.ToInt32() == SM_SCANCHAR)
      {
       objForm.txtScanData.Text +=
Convert.ToChar(msg.LParam.ToInt32());
       count--;
         if (count == 0)
            {
            strScanData = objForm.txtScanData.Text;
            PrintLabel();
            }
          }
         if (msg.WParam.ToInt32() == SM_SCANSTATUS)
        {
         nBarType = (msg.LParam.ToInt32() >> 16) & 0xFFFF;
         count = msg.LParam.ToInt32() & 0xFFFF;
         objForm.txtScanLength.Text = ""+count;
      if(objForm.txtScanLength.Text == "65535")
        {
        objForm.txtScanLength.Text = "";
        objForm.txtScanType.Text = "No Scan";
        }
```

```
//--Barcode Type
    switch (nBarType)
    {
      case -1:
        objForm.txtScanType.Text = "No Scan";
        objForm.txtScanLength.Text = "0";
        break;
      case 0:
        objForm.txtScanType.Text = "Unknown";
        break;
        case 1:
        objForm.txtScanType.Text = "Code 39";
        break;
      case 2:
        objForm.txtScanType.Text = "Codabar";
        break;
      case 3:
        objForm.txtScanType.Text = "Code 128";
        break;
      case 4:
        objForm.txtScanType.Text = "Discrete 2 of 5";
        break;
      case 5:
        objForm.txtScanType.Text = "IATA 2 of 5";
        break;
      case 6:
        objForm.txtScanType.Text = "Interleaved 2 of 5";
        break;
      case 7:
        objForm.txtScanType.Text = "Code 93";
        break;
      case 8:
        objForm.txtScanType.Text = "UPC A";
        break;
      case 9:
        objForm.txtScanType.Text = "UPC E";
        break;
      case 10:
        objForm.txtScanType.Text = "EAN 8";
        break;
      case 11:
        objForm.txtScanType.Text = "EAN 13";
        break;
      case 14:
        objForm.txtScanType.Text = "MSI Plessey";
        break;
```

```
case 15:
  objForm.txtScanType.Text = "EAN 128";
  break;
case 16:
  objForm.txtScanType.Text = "UPC E1";
  break;
case 21:
  objForm.txtScanType.Text = "Trioptic Code 39";
  break;
case 22:
  objForm.txtScanType.Text = "Bookland EAN";
  break;
case 23:
  objForm.txtScanType.Text = "Coupon Code";
  break;
case 48:
  objForm.txtScanType.Text = "RSS-14";
  break;
case 49:
  objForm.txtScanType.Text = "RSS-Limited";
  break;
case 50:
  objForm.txtScanType.Text = "RSS-Expanded";
  break;
case 72:
  objForm.txtScanType.Text = "UPC A + 2";
  break;
case 73:
  objForm.txtScanType.Text = "UPC E + 2";
  break;
case 74:
  objForm.txtScanType.Text = "EAN 8 + 2";
  break;
case 75:
  objForm.txtScanType.Text = "EAN 13 + 2";
  break;
case 80:
  objForm.txtScanType.Text = "UPC E1 + 2";
  break;
case 136:
  objForm.txtScanType.Text = "UPC A + 5";
  break;
```

```
    case 137:
      objForm.txtScanType.Text = "UPC E + 5";
      break;
    case 138:
      objForm.txtScanType.Text = "EAN 8 + 5";
      break;
    case 139:
      objForm.txtScanType.Text = "EAN 13 + 5";
      break;
    case 144:
      objForm.txtScanType.Text = "UPC E1 + 5";
      break;
      } //switch (nBarType)

      objForm.txtScanData.Text = "";

      } // if (msg.WParam.ToInt32() == SM_SCANSTATUS)

      } // if (msg.Msg == unScanMsgID)

// call the base class WndProc for default message handling
        base.WndProc(ref msg);
      }
// protected override void WndProc(ref
// Microsoft.WindowsCE.Forms.Message msg)
//************************************************************
//* PrintLabel
//* Description
//*      Prints Label for some of the Barcodes Scanned
//*      Label Size:  2020 (2.0" x 2.0")
//*      Label Data:
//*         M6039 Platinum
//*         Barcode Type
//*         Barcode + Human Readable
//************************************************************
    public void PrintLabel()
     {
     string strFCN1 = "~201";   // Code 128 Function Code F1
     string strType;            // Barcode Type

         string fmtC39 = "{F,1,A,R,E,200,200,\"C39\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,20,V,23,2,4,12,100,8,C,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";
```

```
     string fmtCodabar = "{F,1,A,R,E,200,200,\"CODABAR\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,18,V,23,100,5,8,100,8,B,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtC128 = "{F,1,A,R,E,200,200,\"C128\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,22,V,23,100,8,8,100,8,B,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtI2OF5 = "{F,1,A,R,E,200,200,\"I2OF5\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,22,V,23,11,3,13,100,8,C,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtC93 = "{F,1,A,R,E,200,200,\"C93\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,18,V,23,100,23,10,100,8,B,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtUPCA = "{F,1,A,R,E,200,200,\"UPCA\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,12,F,25,28,1,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtUPCA2 = "{F,1,A,R,E,200,200,\"UPCA2\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,14,F,25,38,10,2,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtUPCA5 = "{F,1,A,R,E,200,200,\"UPCA5\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,17,F,25,28,11,2,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtUPCE = "{F,1,A,R,E,200,200,\"UPCE\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,7,V,25,56,2,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";
```

```
     string fmtEAN8 = "{F,1,A,R,E,200,200,\"EAN8\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,8,V,25,42,6,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN82 = "{F,1,A,R,E,200,200,\"EAN82\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,10,V,25,25,14,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN85 = "{F,1,A,R,E,200,200,\"EAN85\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,13,V,25,40,15,2,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN13 = "{F,1,A,R,E,200,200,\"EAN13\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,13,V,25,24,7,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN132 = "{F,1,A,R,E,200,200,\"EAN132\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,15,V,25,40,16,2,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN135 = "{F,1,A,R,E,200,200,\"EAN135\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,18,V,25,25,17,2,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtMSI = "{F,1,A,R,E,200,200,\"MSI\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,11,V,25,100,9,7,100,8,B,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtEAN128 = "{F,1,A,R,E,200,200,\"EAN128\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"T,1,22,V,6,1,0,1,1,1,O,C,0,0|" +
"B,2,22,V,23,100,8,8,100,8,B,0|" +
"T,3,22,V,133,1,0,1,1,1,O,C,0,0|}";
```

```
     string fmtUPCE2 = "{F,1,A,R,E,200,200,\"UPCE2\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,9,V,25,35,12,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

     string fmtUPCE5 = "{F,1,A,R,E,200,200,\"UPCE5\"|"+
"C,150,49,0,50,8,8,A,L,0,0,\"M6039 Platinum\",1|" +
"B,1,12,V,25,20,13,4,100,7,L,0|" +
"T,2,22,V,133,1,0,1,1,1,O,C,0,0|}";

    Ultra.Print rPrint = new Print();
// Instantiate Print Class

// --Get the Barcode Type
     strType = objForm.txtScanType.Text;

// --Clear any errors
     rPrint.ClearError();

// --Print Format & Batch
     switch (strType)
     {
     case "Code 39":
       rPrint.Text = fmtC39;
       rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
       rPrint.Text = "1,\"" + strScanData + "\"|2,\"" +
strScanData + "\"|3,\"" + strType + "\"|}";
       objForm.lblStatus.Text = "";
       break;

    case "Codabar":
       rPrint.Text = fmtCodabar;
       rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
       rPrint.Text = "1,\"" + strScanData + "\"|2,\"" +
strScanData + "\"|3,\"" + strType + "\"|}";
       objForm.lblStatus.Text = "";
       break;

     case "Code 128":
       rPrint.Text = fmtC128;
       rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
       rPrint.Text = "1,\"" + strScanData + "\"|2,\"" +
strScanData + "\"|3,\"" + strType + "\"|}";
       objForm.lblStatus.Text = "";
       break;
```

```
 case "Discrete 2 of 5":
  objForm.lblStatus.Text = "D2of5 not available to print";
  break;

 case "IATA 2 of 5":
  objForm.lblStatus.Text = "IATA 2of5 not available to print";
  break;

 case "Interleaved 2 of 5":
  rPrint.Text = fmtI2OF5;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
  rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|3,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "Code 93":
  rPrint.Text = fmtC93;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
  rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|3,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC A":
  rPrint.Text = fmtUPCA;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC E":
  rPrint.Text = fmtUPCE;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 8":
  rPrint.Text = fmtEAN8;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;
```

```
 case "EAN 13":
  rPrint.Text = fmtEAN13;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "MSI Plessey":
  rPrint.Text = fmtMSI;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
  rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strScanData
+ "\"|3,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 128":
  rPrint.Text = fmtEAN128;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|";
  rPrint.Text = "1,\"" + strScanData + "\"|2,\"" + strFCN1 +
strScanData + "\"|3,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC E1":
  objForm.lblStatus.Text = "UPC E1 not available to print";
  break;

 case "Trioptic Code 39":
  objForm.lblStatus.Text = "Trioptic Code 39 not available to
print";
  break;

 case "Bookland EAN":
  objForm.lblStatus.Text = "Bookland EAN not available to
print";
  break;
```

```
 case "Coupon Code":
  objForm.lblStatus.Text = "Coupon Code not available to
print";
  break;

 case "RSS-14":
  objForm.lblStatus.Text = "RSS-14 not available to print";
  break;

 case "RSS-Limited":
  objForm.lblStatus.Text = "RSS-Limited not available to
print";
  break;

 case "RSS-Expanded":
  objForm.lblStatus.Text = "RSS-Expanded not available to
print";
  break;

 case "UPC A + 2":
  rPrint.Text = fmtUPCA2;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC E + 2":
  rPrint.Text = fmtUPCE2;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 8 + 2":
  rPrint.Text = fmtEAN82;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 13 + 2":
  rPrint.Text = fmtEAN132;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;
```

```
 case "UPC E1 + 2":
  objForm.lblStatus.Text = "UPC E1 + 2 not available to
print";
  break;

 case "UPC A + 5":
  rPrint.Text = fmtUPCA5;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC E + 5":
  rPrint.Text = fmtUPCE5;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 8 + 5":
  rPrint.Text = fmtEAN85;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "EAN 13 + 5":
  rPrint.Text = fmtEAN135;
  rPrint.Text = "{B,1,N,1|E,0,0,1,1,0,1|1,\"" + strScanData +
"\"|2,\"" + strType + "\"|}";
  objForm.lblStatus.Text = "";
  break;

 case "UPC E1 + 5":
  objForm.lblStatus.Text = "UPC E1 + 5 not available to
print";
  break;
  } //switch (strType)

  } // public void PrintLabel()

  } // public class MyMessageWindow :
Microsoft.WindowsCE.Forms.MessageWindow

} // namespace ScanNPrintCS
```

INDEX

A

AimDuration function 4-2
applications
  building 2-4
  writing 2-4

B

bar codes
  AimDuration 4-2
  BdirRedundancy 4-3
  Codabar 4-10
  Code128 4-12
  Code39 4-14
  Code93 4-17
  D2of5 4-19
  GoodScanWav 4-4
  I2of5 4-21
  LinearSecurity 4-5
  MSI 4-24
  NoReadWav 4-5
  Postamble 4-8
  Preamble 4-7
  RSS 4-26
  Timeout 4-9
  UPCEAN 4-28
Battery functions 3-4
battery level
  checking if okay for printing 3-4
  retrieving 3-5
BdirRedundancy function ... 4-3
black mark sensor,
  retrieving state of 3-19
building
  applications 2-4
Byte function 3-16

C

Calibrate function 3-2
calibrating supplies 3-2
checking
  if battery level is okay for printing 3-4
ClearError function 3-23
clearing
  motion control errors 3-23
Codabar
  scanning 4-10
Codabar bar codes
  retrieving configuration values ... 4-10
Codabar function 4-10
Code128
  scanning 4-12
Code128 bar codes
  retrieving configuration values ... 4-12
Code128 function 4-12
Code39
  scanning 4-14
Code39 bar codes
  retrieving configuration values ... 4-14
Code39 function 4-14
Code93
  scanning 4-17
Code93 bar codes
  retrieving configuration values ... 4-17
Code93 function 4-17
CommitChanges function .. 4-34
contents, of SDK 1-3
copying, application to printer 2-6
copying, data to printer 2-6
creating
  MPCLII packets 2-1
current supply type, setting 3-3, 3-27

D

D2of5
scanning 4-19
D2of5 bar codes
retrieving configuration values... 4-19
D2of5 function 4-19
DataMode function 4-35
DisableAllCodes function . 4-38
DisableScanning function . 4-39
disabling scans 4-39
display 1-4
documentation, related 1-3

E

EnableScanning function .. 4-37
enabling scans 4-37
errors
clearing for motion control 3-23

F

fBlackMark function 3-19
features, of printer 1-4
Feed function 3-7
feeding labels 3-7
File function 3-8
FileParse function 3-18
fOnDemand function 3-21
fonts
descriptions 1-6
functions
AimDuration 4-2
Battery 3-4
BdirRedundancy 4-3
Byte 3-16
Calibrate 3-2
ClearError 3-23
Codabar 4-10
Code128 4-12
Code39 4-14
Code93 4-17
CommitChanges 4-34
D2of5 4-19
DataMode 4-35
DisableAllCodes 4-38
DisableScanning 4-39
EnableScanning 4-37
fBlackMark 3-19
Feed 3-7
File 3-8
FileParse 3-18
fOnDemand 3-21
GoodScanWav 4-4
I2of5 4-21
ISBatteryOkToPrint 3-4
LastPrintStatus 3-12
LinearSecurity 4-5
LockCfgMenu 3-24
Misc 3-23
MSI 4-24
nBatteryLevel 3-5
NoReadWav 4-5
nStatus 3-25
nStockType 3-3, 3-27
Postamble 4-8
Preamble 4-7
printing 3-1
Printing 3-7
RSS 4-26
ScannerMode 4-40
scanning 4-1
SendScanStatus 4-42
Sensor 3-19
Stock 3-2
Text 3-10
TextDoubleByte 3-14
Timeout 4-9
Trigger 4-43
TriggerMode 4-44
UPCEAN 4-28
funtions
scanning 4-40

G

General
scanning 4-2
General bar codes
retrieving configuration values 4-2
GoodScanWav function 4-4

H

hardware requirements 1-2

I

I2of5
  scanning 4-21
I2of5 bar codes
  retrieving configuration values ... 4-21
I2of5 function 4-21
import files 2-4
information about supplies,
  specifying 3-2
initiating trigger 4-43
initiating triggers 4-44
IsBatteryOkToPrint function 3-4

K

keyboard 1-5

L

labels
  feeding 3-7
LastPrintStatus function ... 3-12
LinearSecurity function 4-5
LockCfgMenu function 3-24

M

memory
  description 1-4
Misc functions 3-23
motion control errors, clearing 3-23
MPCLII packets
  creating 2-1
  loading batch 3-18
  loading individually 3-8, 3-10, 3-14, 3-16
MSI
  scanning 4-24
MSI bar codes
  retrieving configuration values ... 4-24
MSI function 4-24

N

nBatteryLevel function 3-5
network notes 2-7
NoReadWav function 4-5
nStatus function 3-25
nStockType function .. 3-3, 3-27

O

on-demand sensor,
  retrieving state of 3-21

P

packets, MPCLII
  creating 2-1
  loading batch 3-18
  loading individually 3-8, 3-10, 3-14, 3-16
Postamble function 4-8
Preamble function 4-7
Print subsystem
  retrieving status of 3-12, 3-25
  writing MPCLII packets to 3-8, 3-10, 3-14, 3-16, 3-18
printer
  features 1-4
printing
  functions 3-1
Printing functions 3-7

R

requirements, system 1-2
retrieving
  battery level 3-5
  black mark sensor state. 3-19
  on-demand sensor state 3-21
  Print subsystem status 3-12, 3-25
RSS
  scanning 4-26
RSS bar codes
  retrieving configuration values ... 4-26
RSS function 4-26

S

scan, disabling 4-39
scan, enabling 4-37
scan, functions 4-40
ScannerMode function 4-40
scanners 1-4
scanning
  Codabar bar code configuration . 4-10
  Code128 bar code configuration 4-12
  Code39 bar code configuration 4-14
  Code93 bar code configuration 4-17
  D2of5 bar code configuration4-19
  functions 4-1
  General bar code configuration4-2
  I2of5 bar code configuration4-21
  MSI bar code configuration4-24
  RSS bar code configuration4-26
  UPCEAN bar code configuration 4-28
SDK
  contents 1-3
SendScanStatus 4-42
SendScanStatus Codes 4-46
Sensor functions 3-19
setting
  current supply type .3-3, 3-27
software requirements 1-2
speaker 1-4
specifying supplies information3-2
status of Print subsystem,
  retrieving 3-12, 3-25
Stock functions 3-2
supplies
  calibrating 3-2
  specifying information about3-2
system requirements 1-2

T

Text function 3-10
TextDoubleByte function ... 3-14
Timeout function 4-9
Trigger function 4-43
trigger, initiating 4-43, 4-44
TriggerMode function 4-44

U

UPCEAN
  scanning 4-28
UPCEAN bar codes
  retrieving configuration values .. 4-28
UPCEAN function 4-28

W

Windows notes 2-7
writing
  applications 2-4
  MPCLII packets to Print subsystem3-8, 3-10, 3-14, 3-16, 3-18

Visit **www.paxar.com** for sales, service, supplies, information, and telephone numbers for our International locations.

TOLL FREE:
1-800-543-6650 (In the U.S.A.)
1-800-363-7525 (In Canada)