



# C# Software Development Kit

How to Use StarIO for [Portable Printers](#) in C#

Thermal Line Mode Printing for Star Portable Printers

This SDK contains a C# Visual Studio 2005 project for use on Microsoft Windows XP, Vista, & 7

<b>Compatible Star Printer Models:</b> <ul style="list-style-type: none"><li>• SM-S200</li><li>• SM-S300</li><li>• SM-T300</li><li>• SM-S400</li></ul> <b>Supported Interfaces:</b> <ul style="list-style-type: none"><li>• Serial</li><li>• Wireless LAN</li><li>• Bluetooth</li></ul>	<b>Functions Include:</b> <ul style="list-style-type: none"><li>• Sample Receipt (2, 3, &amp; 4 inch)</li><li>• Read Printer Status</li><li>• Direct Raster Printing</li><li>• 1D Barcodes</li><li>• 2D Barcodes</li><li>• Page Mode</li><li>• Code Pages</li><li>• Character Sets</li><li>• Set Printer Font</li><li>• Line Feed</li><li>• Text Formatting</li><li>• Mag Stripe Reading</li></ul>
--	--

Requirements: Visual Studio 2005 or later and .NET Framework 2.0 or later.

NOTE:

- This sample program provides source code and dependencies for 32-bit & 64-bit.
- Executable files for 64-bit cannot be executed in a 32-bit environment but 32-bit programs can run on 64-bit operating systems. This is set as 32-bit by default.
- When you open this project in Windows Vista or 7, execute Visual Studio as an administrator before opening the sln file. You can achieve this by right-clicking on the Visual Studio 2005 icon and clicking "Run as administrator" in the menu displayed.




## Table of Contents

- ❖ [About this Manual](#)
- ❖ [How to compile and run the C# SDK](#)
- ❖ [Using the SDK with Portable Star Micronics Printer](#)
  - [Port Name and Interface Relation](#)
  - [Overview of how this C# SDK is designed](#)
- ❖ [StarIO - \(StarIOPort.DLL\)](#)
  - [How to include StarIO into your project](#)
  - [Configuring your project for 32- or 64-bit](#)
  - [StarIO Methods Quick Overview](#)
    - [Class Variables](#)
      - [PortName](#)
      - [PortSettings](#)
      - [Timeout](#)
    - [GetPort](#) - Opening the port to the printer
    - [WritePort](#) - Writing data (print job) to the printer
    - [ReadPort](#)
    - [ReleasePort](#) - Closing the port to the printer
    - [ResetDevice](#)
    - [GetParsedStatus](#)
- ❖ [Tips for software application development when using StarIO](#)
  - [Classes](#)
  - [Key Terms](#)
  - [Hexadecimal Dumping Mode](#)
  - [The StarIO Convenience](#)
  - [Additional Features](#)
  - [Communication Options](#)
- ❖ [Additional Resources](#)
  - Star Micronics Developers Network
    - Updated versions of this manual and source code
    - Star Micronics Printer Drivers
    - Technical Questions/Support
  - [ASCII Table](#)

## About this Manual

This manual is designed to help you understand StarIO and how to build a C# application to interact with Star Micronics Portable Thermal Line Mode Printers. It is important to understand the basics of the C# language and the .NET framework. Although this SDK is for the programming language C#, there are other SDKs available at our website in the Developers section. Check the Developers section of our site for the newest SDKs, technical documentation, FAQs, and much more additional resources.

### Key Legend:

<i>Warning</i>		Explains potential issues
<i>Avoid Doing This</i>		Explains things not to do
<i>Note</i>		Provides important information and tips

### CAUTION:

- The information in this manual is subject to change without notice.
- STAR MICRONICS CO., LTD. has taken every measure to provide accurate information, but assumes no liability for errors or omissions.
- STAR MICRONICS CO., LTD. is not liable for any damages resulting from the use of information contained in this manual.
- Reproduction in whole or in part is prohibited.

## How to compile and run the C# SDK

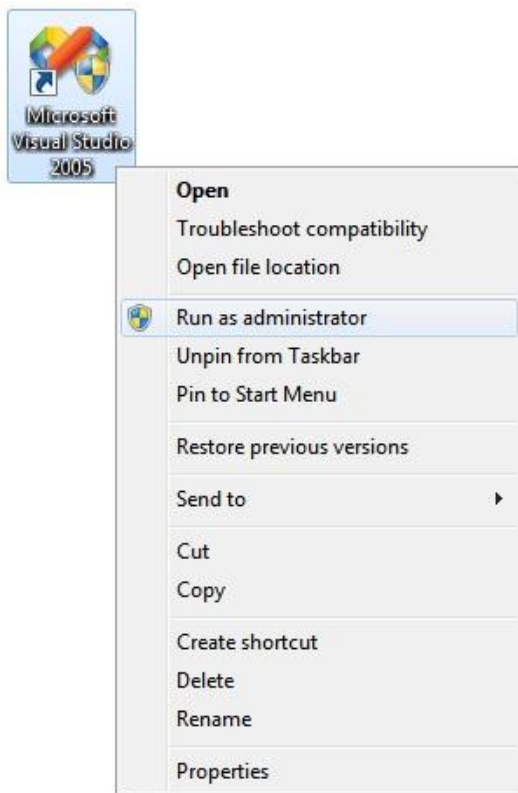
This section will explain:

1. How to open the Visual Studio 2005 C# SDK project.
2. Compiling the project.
3. Running the project.

How to open the Visual Studio 2005 C# SDK project:

In Windows XP, open Visual Studio 2005.

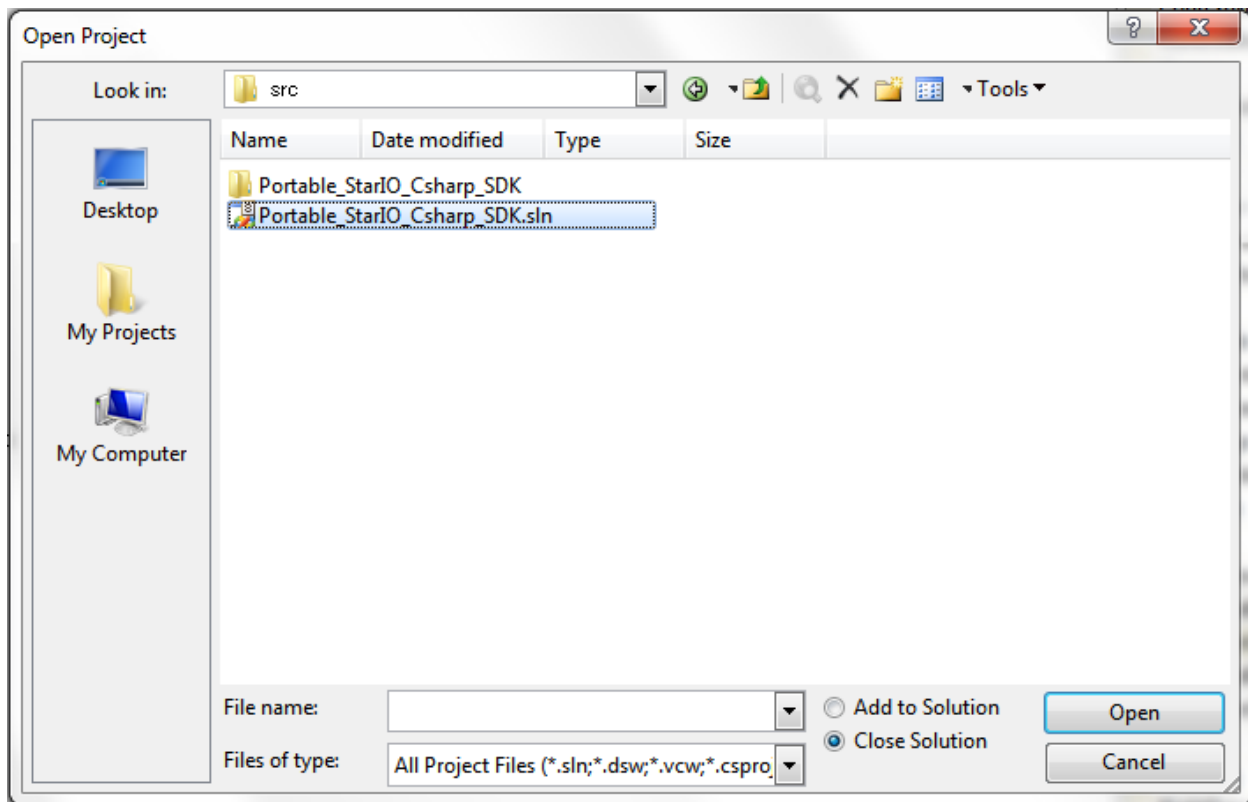
In Vista or 7, right click on Visual Studio 2005 icon and click “Run as administrator”.



Once Visual Studio is running, click on File->Open->Project/Solution...

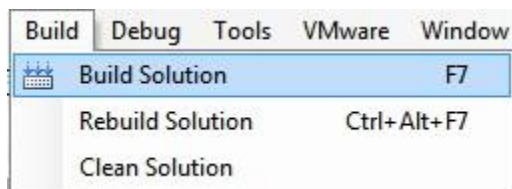


Navigate to the C# SDK folder titled “Portable\_StarIO\_Csharp\_SDK” and click on the “sln” file titled “Portable\_StarIO\_Csharp\_SDK.sln” to open the SDK project.



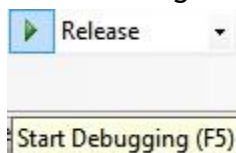
### Compiling the project:

Click on the menu item “Build” and then click “Build Solution” or hit F7



### Running the project:

Click on the green arrow to “Start Debugging” or hit F5



## Using the SDK with Portable Star Micronics Printers

Please make sure you have a compatible Star Micronics Portable Line Mode Printer Model.

### Port Name and Interface Relation:

StarIO uses specific port names to identify what port will be used. These are very important to understand because not following the naming convention correctly will fail to communicate with the printer.

Interface	Port Name	Port Settings
Serial	COMn	MINI;57600,n,8,1,h
Bluetooth	COMn	MINI;57600,n,8,1,h
Wireless LAN (TCP/IP)	tcp:"IP Address"	MINI;

## Overview of how this C# SDK is designed

This overview will touch briefly on key components of the SDK and how to find them.

Focus on the file “SMForm.cs” which contains all the business logic and StarIO commands.

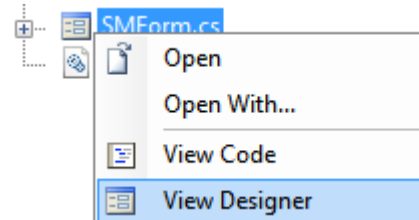
To navigate to a specific command is easy; simply right click on the file called “SMForm.cs” in the solution explorer. Then click on the option “View Designer” which will open the designer.

So let us say for example you wanted to find out what code block is executed for the “Print Sample Receipt” button.

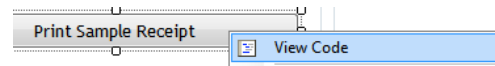
All you need to do is double click on the button in the designer view which will switch the view from designer to code view and take you to the code block handling the mouse click event for the “Print Sample Receipt” code.

With this tip known, you can now fully explore all of the advanced StarIO commands with ease.

Please note that there is a stopping point in the code where everything below the line “CODE BELOW THIS POINT IS NOT USEFUL AS SDK EXAMPLES” is not really useful as code samples because they are related specifically to this SDK program. This code is mostly for the UI so please do not get confused into including this code into your business application.



Click “View Designer”



Right Click Button, then click “View Code”

```
//-----
//  Sample Receipt
//-----
/// <summary>
/// This function will print a sample receipt to your Star
/// For command codes and examples on how to use spe
/// look at the sample receipt string and reference the
/// You can find these command codes and more in the
/// To access this documentation, open the SDK Readme
/// </summary>
private void cmdPrint_Click(object sender, EventArgs e)
{
    string receipt = "\x1b\x1d\x61\x1" +
        "\x5B" + "If loaded.. Stored Logo 1 goes here" + "\x5D\r" +
        "\x1B\x1C\x70\x1\x0" + //Store
        "Star Micronics Receipt" +
}
```

Visual Studio takes you to the event code block that holds the StarIO commands.



This SDK contains a folder named “Documentation” and the folder contains helper RTF documents for the program to load into the right help window. If these files / folder are moved and/or renamed from this folder, the RTF files will not get loaded correctly and the SDK will not display this helpful info.

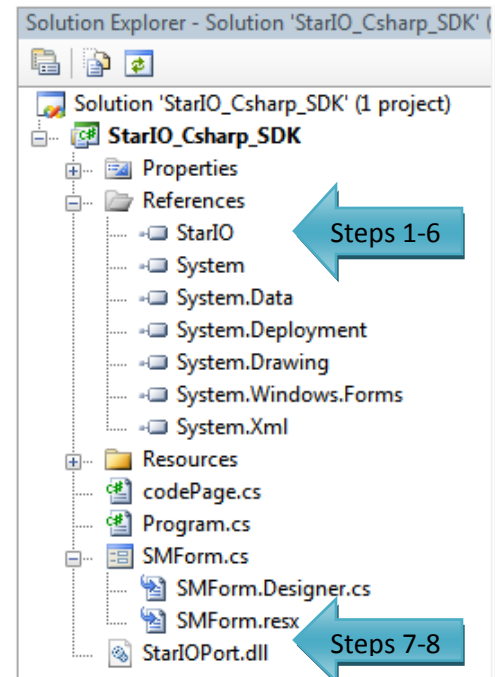
## StarIO - (StarIOPort.dll & StarIO.dll)

### How to include StarIO into your project:

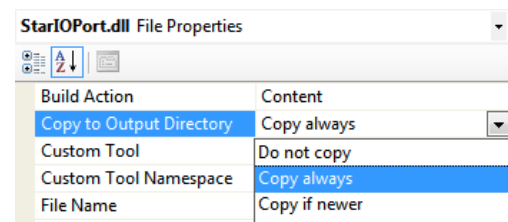
The file StarIOPort.dll is a dynamically linking library that you can include into your C# projects to expose StarIO methods. The file StarIO.dll is a .NET wrapper for StarIOPort.dll.

To include this DLL into your project:

1. Right-Click on the Project in Solution Explorer
2. Choose Add Reference
3. Click the "Browse" tab
4. Click the "Dependencies" folder attached with this SDK and select the folder "x86" or "x64" based on your target platform.
5. Click add existing item, then "StarIO.dll"
6. In the Solution Explorer, select StarIO under references. Set the Copy to Output - Property of StarIO.dll to Copy always.
7. StarIOPort.dll must reside in the same folder as StarIO.dll to work. To make sure your project copies the DLLs to your output debug folder, right click on the project name again and select "Add" -> "Existing Item..." then select show all files and select StarIOPort.dll
8. To ensure output debug directory is coping the DLL when you build, click on the item for StarIOPort.dll and under properties, select "Copy to Output Directory" and set to copy always.
9. To expose StarIO, add  
`"using StarMicronics.StarIO;"`  
 at the top of your main code.
10. Now you can access all of StarIO's methods!



*StarIO being used as a Reference*



*Set StarIOPort.dll to always copy*



**WARNING:** Make sure StarIO.dll and StarIOPort.dll are in the same directory as each other. StarIO.dll links to StarIOPort.dll by looking in the same folder that the StarIO.dll is located in.

### Configuring your project to x64 or x86 with StarIO:

EXE	Runs on x86?	Runs on x64?	Can use x86 C# dll?	Can use x64 C# dll?	Can use “Any CPU” C# dll?	Can use x86 C++ native dll?	Can use x64 C++ native dll?
x86 C# exe	Yes, in CLR	Yes, in WOW64 +CLR	Yes	No	Yes	Yes	No
x64 C# exe	No	Yes, in CLR	No	Yes	Yes	No	Yes
“Any CPU” C# exe	Yes, in CLR	Yes, in CLR	Only on x86	Only on x64	Yes	Only on x86	Only on x64

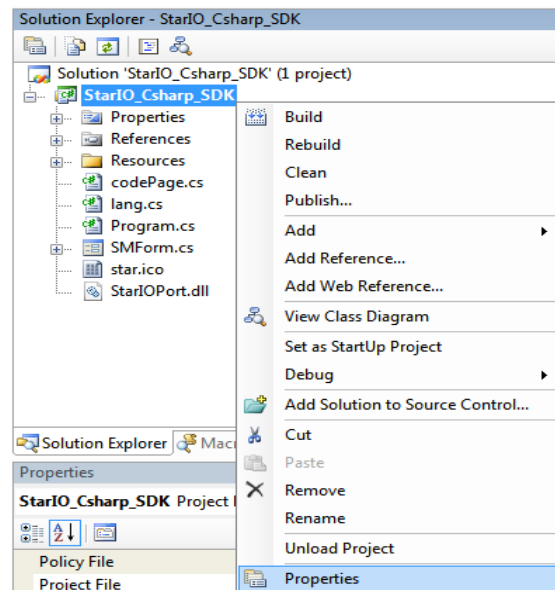
Compiling your project with the correct StarIOPort.DLL is very important to get the maximum speed from your CPU. Your main two choices are 32-bit (x86) and 64-bit (x64) operating systems. WoW64, which means **Windows 32-bit On Windows 64-bit**, allows you to make an x86 application with 32-bit binaries to run on either 32-bit or 64-bit. ***We recommend setting your project to x86 to allow it to run on both 32 and 64 OS.*** If you wish to make a 100% 64-bit project then use StarIO dlls under the folder “Dependencies/x64” for 64-bit libraries of StarIO.

To change it to x86, x64, or Any CPU:

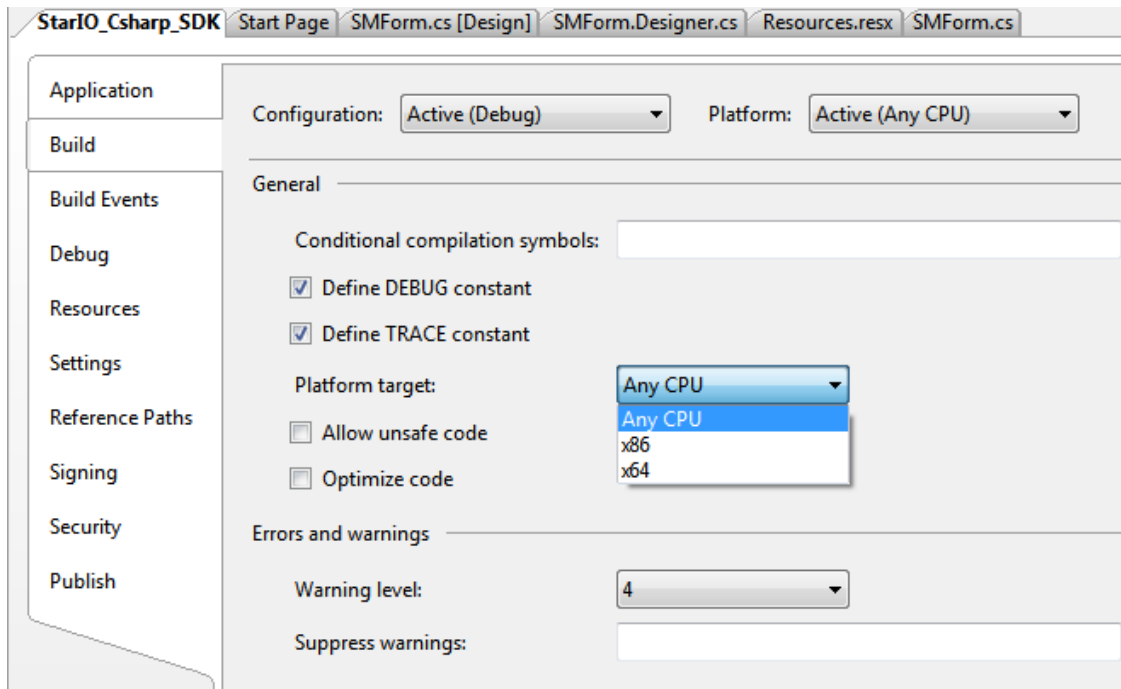
Right Click on your C# project name and click Properties.

OR

Click on the project name then use the keyboard shortcut Alt+F7



1. Click on the “Build” tab on the left and find “Target Platform”.



2. Select the platform you wish to compile for (Any CPU, **x86(recommended)**, x64).
3. Now that you have selected your platform, you must add 32-bit or 64-bit StarIO DLL.



**WARNING:** If you set your project to “Any Cpu” and use 32-bit StarIO libraries, you will find that Windows x64 will not be able to run it. To fix this, setup your project as x86 and use only x86 StarIO which will run on both systems. If you are developing purely for the x64 environment, then set to x64 with x64 StarIO.

4. Once you have selected your target platform and saved your changes, follow the directions in [“How to include StarIO to your project”](#) on how to add a reference to StarIO 32-bit or 64-bit DLL binaries into your project with the corresponding library.

### StarIO Methods Overview:

**Class Variables** include portName (string), portSettings (string), and Timeout (int).



These 3 variables will be “read only” if accessed directly. To assign them use [GetPort\(portName,portSettings,Timeout\);](#) which will allow you to pass in variables to this methods which then assigns the 3 class variables with values.

**portName** is what you will be using to specify the port of communication to the printer.

*Ex. “tcp:192.168.1.2” “COM4”*

**portSettings** is what you will use for configuring Serial connections correctly.

*Ex. “MINI;57600,n,8,1,h”*

The following are the acceptable inputs from left to right:

baud: 38400, 19200, 9600, 4800, 2400

parity: n, e, o

data-bits: 8, 7

stop-bits: 1

flow-ctrl: n, h

**Timeout** is a millisecond timeout controlled internally and is used for communication in the APIs (this parameter guarantees that all of the below APIs will complete in a bounded amount of time, but does NOT guarantee the exact timeout length)

### GetPort

```
public static StarIOPort GetPort(String portName, String portSettings, Int TimeoutMillis)
    throws StarIOPortException
```

GetPort is what you will be using to “open” the port to the printer. Using one of the valid inputs for portName and portSettings as mentioned previously before this, you can pass your connection string into the StarIO class so that it will correctly set its private variables.

The following would be an actual usage of GetPort in C#:

```
IPort port = null;

Try
{
    port = StarMicronics.StarIO.Factory.I.GetPort(portName, portSettings, 10000);
}
Catch(PortException)
{
    //There was an error opening the port
}
```

**IPort** is a part of StarIO and this will allow you to create a “port” handle. The above example shows the port being created and set to null then being assigned the actual port hook on the following line that contains GetPort.



Always use a **Try**, **Catch** when using **GetPort**. If the port cannot be opened because of connection problems, your program will crash unless you use a **Try**, **Catch** like the above example.

## ReadPort

```
public int ReadPort(Byte[] readBuffer, Int offset, Int size)
```

throws [StarIOPortException](#)

This method reads data from the device. Only use this if you really need to read raw bytes from the printer.



**Do not use this method to read raw status.**

Use [GetOnlineStatus](#) or [GetParsedStatus](#) for getting status.

### Parameters:

`readbuffer` – A Byte Array buffer into which data is read.

`offset` - specifies where to begin writing data into the `readBuffer[]`

`size` – Total number of bytes to read.

### Returns:

The number of bytes that were actually read. Under some interface types, this function will succeed even when no data was read in. Your application should call this function a limited number of times until the expected data has been read in or until an application determined retry threshold has been reached.

### Throws:

[StarIOPortException](#) - when a communication failure occurs

## ReleasePort

```
public static void ReleasePort(StarIOPort port)
```

This function closes a connection to the port specified.

### Parameters:

`port` - [StarIOPort](#) type representing a previously initialized port.



Always release (close) ports that you get (open).

Leaving a port open will cause future calls to open the port to fail.

## WritePort

```
public int WritePort(Byte[] writeBuffer, Int offset, Int size)
```

throws [StarIOPortException](#)

This method writes data to the device. Use this to print to the printer, send commands, etc. The following is an example of how to use this method:

Please keep in mind this is the simplest way to send data to the printer.

The C# SDK has code in `printToPrinter` that is more complex than this but that code block will show you how to verify data transmission to the printer whereas this code is just dumping it:

```
//Set a byte array to send to the printer
//command = { A, B, C, D, Feed 3mm, Full Cut}
Byte[] command = {0x41, 0x42, 0x43, 0x44, 0x1B, 0x7A, 0x00, 0x1B, 0x64, 0x02};

UInt bytesWritten = 0;

Try
{
    While(command.Length > bytesWritten)
    {
        bytesWritten += port.WritePort(command, bytesWritten, (unit)command.Length - bytesWritten);
    }
}
Catch(PortException)
{
    //There was an error writing to the port
}
```

Remember to use a [Try](#), [Catch](#) for safe programming practices.

### Parameters:

`writeBuffer` - Contains the output data in a byte array.

`offset` - Specifies where to begin pulling data from `writeBuffer` .

`size` - Number of bytes to write.

### Returns:

The number of bytes that were actually written. Under some interface types, this function will succeed even when no data was written out. Your application should call this function a limited number of times until all the data has been written out or until an application determined retry threshold has been reached.

**Throws:**

`StarIOException` - when a communication failure occurs

**ResetDevice**

```
public void ResetDevice()
```

`throws StarIOException`

This method resets the device at the hardware level.

**Throws:**

`StarIOException` - when a communication failure occurs

**GetParsedStatus**

```
public StarPrinterStatus GetParsedStatus ()
```

`throws StarIOException`

This method retrieves detailed status form the printer with StarIO.

**Returns:**

`StarPrinterStatus` structure giving the current device status

**Throws:**

`StarIOException` - when a communication failure occurs

This method uses a class structure that is included with StarIO called `StarPrinterStatus`

This structure gives the printer's status in both boolean and binary form.

Create the `StarPrinterStatus` object in your project by doing the following:

```
StarPrinterStatus printerStatus = port.GetParsedStatus();

If(printerStatus.Offline == false)
{
    If(printerStatus.BlackMarkError == true){
        //There was a black mark error
    }
}
```

```
    If(printerStatus.CompulsionSwitch == true){  
        //Cash drawer is open  
    }  
    Else{  
        //Cash drawer is closed  
    }  
}  
Else{  
    //If True, then the printer is offline.  
}
```

There are different statuses that are pulled when you initialize **StarPrinterStatus**.

**This is a list of statuses that are in the class structure **StarPrinterStatus**:**

CoverOpen returns a **Boolean**.

Offline returns a **Boolean**.

CompulsionSwitch returns a **Boolean**.

OverTemp returns a **Boolean**.

UnrecoverableError returns a **Boolean**.

CutterError returns a **Boolean**.

MechanicalError returns a **Boolean**.

HeadThermistorError returns a **Boolean**.

ReceiveBufferOverflow returns a **Boolean**.

PageModeCommadError returns a **Boolean**.

HeadUpError returns a **Boolean**.

VoltageError returns a **Boolean**.

ReceiptPaperEmpty returns a **Boolean**.

ReceiptPaperNearEmptyInner returns a **Boolean**.

ReceiptPaperNearEmptyOuter returns a **Boolean**.

RawStatus returns a **Byte[]** array.

## Tips for App Development when using StarIO

Star Micronics prides itself as the industry leader in great POS products and with great power comes great responsibility. Below is a tips section just to help you get on the fast track to software development with StarIO.

**TIP #1:** If you are going to be coding a large project, create a class to abstract all the printing methods into class(s) instead of having the code reside in the main code block. This will help with code reusability and will also save you time in the long run from having to find one line of code in the main code. By having StarIO only reside in the class(s), you will be fully taking advantage of object oriented programming.

**TIP #2:** Know what the differences and definitions of (ASCII & Unicode), (Hex & Decimal), and (Byte & Char) are. A byte is normally 8-bits long which would be 8 digits of binary (1s and 0s). These bytes are just 8 bits of binary data but bytes can also be int or char. The three different variable types basically hold the data in the same way but there are slight differences. Try to code with Bytes instead of Chars, ints, or strings when choosing a variable to contain your print job data. ASCII to Unicode and vice versa conversions are sometimes unsecure so make sure you know what and how the encoding class works with these. Big mistakes made in Unicode are culture-sensitive search and casing, surrogate pairs, combining characters, and normalization which are answered [here](#).

**TIP #3:** HEX DUMP MODE! If you are debugging and your application seems to have a bug in it use hex dump mode on the printer. This is the best way to verify what is being sent out of the computer is being received correctly. To put the printer in hex dump mode, turn the printer off, open the cover to the paper, hold the feed button down, turn the printer back on, close the cover, let go of the feed button. Hex dump mode is a sure fire way to verify hex data is sent correctly. When in hex dump mode, printer functions will not work.

**TIP #4:** Do not waste time trying to reverse engineer StarIO command codes. All the available StarIO commands are available in the Thermal Line Mode Spec Manual and that is the best resource to use when researching a specific StarIO command. This SDK & Manual was built to help you (The Developer) have a very easy job ahead of you to program for Star Printers.

**TIP #5:** If there is a command that is not covered in this SDK but you wish to see a code snippet of that command in use then visit our Developers' section for a possible code block that matches your needs.

**TIP #6:** StarIO, ESC/POS, UPOS: JavaPOS, POS for .NET, & OPOS are all different ways to communicate with the printer. Visit our Developers' section for more info on these. This SDK covers StarIO only.

## Additional Resources

This section will share resources that will help you develop good software with StarIO.

Please get the programmers manual for Star Portable Printers from the link below.

### [Star Micronics Developers Network](#)

Browse Star Micronics' FAQs, ask a question, look up information, etc.

The Developers Network gets you access to:

- Updated Versions of this Manual and Source Code
- Star Micronics Printer Drivers
- Technical Questions/Support

### [Character Encoding in the .NET Framework](#)

If you don't know what ASCII and Unicode is, this is a good place to start.

### [Microsoft .NET Internationalization](#)

Good resource for more detail on internationalization.

### [Visual C# Developer Center](#)

Great place to learn more about the C# language.

### [Unicode.org](#)

The Unicode Consortium - Good place to learn more about Unicode.

### [1D Barcodes](#)

Barcode Island is a great resource for specs on 1D barcodes.

### [2D Barcodes](#)

Great place for information on 2D Barcodes, [QR Codes](#), and [PDF417](#)

### [Code Pages](#)

Learn about Code Pages here.

## ASCII Table Resource

ASCII Hex Symbol	ASCII Hex Symbol	ASCII Hex Symbol	ASCII Hex Symbol
0 0 NUL	16 10 DLE	32 20 (space)	48 30 0
1 1 SOH	17 11 DC1	33 21 !	49 31 1
2 2 STX	18 12 DC2	34 22 "	50 32 2
3 3 ETX	19 13 DC3	35 23 #	51 33 3
4 4 EOT	20 14 DC4	36 24 \$	52 34 4
5 5 ENQ	21 15 NAK	37 25 %	53 35 5
6 6 ACK	22 16 SYN	38 26 &	54 36 6
7 7 BEL	23 17 ETB	39 27 '	55 37 7
8 8 BS	24 18 CAN	40 28 (	56 38 8
9 9 TAB	25 19 EM	41 29 )	57 39 9
10 A LF	26 1A SUB	42 2A *	58 3A :
11 B VT	27 1B ESC	43 2B +	59 3B ;
12 C FF	28 1C FS	44 2C ,	60 3C <
13 D CR	29 1D GS	45 2D -	61 3D =
14 E SO	30 1E RS	46 2E .	62 3E >
15 F SI	31 1F US	47 2F /	63 3F ?
ASCII Hex Symbol	ASCII Hex Symbol	ASCII Hex Symbol	ASCII Hex Symbol
64 40 @	80 50 P	96 60 `	112 70 p
65 41 A	81 51 Q	97 61 a	113 71 q
66 42 B	82 52 R	98 62 b	114 72 r
67 43 C	83 53 S	99 63 c	115 73 s
68 44 D	84 54 T	100 64 d	116 74 t
69 45 E	85 55 U	101 65 e	117 75 u
70 46 F	86 56 V	102 66 f	118 76 v
71 47 G	87 57 W	103 67 g	119 77 w
72 48 H	88 58 X	104 68 h	120 78 x
73 49 I	89 59 Y	105 69 i	121 79 y
74 4A J	90 5A Z	106 6A j	122 7A z
75 4B K	91 5B [	107 6B k	123 7B {
76 4C L	92 5C \	108 6C l	124 7C
77 4D M	93 5D ]	109 6D m	125 7D }
78 4E N	94 5E ^	110 6E n	126 7E ~
79 4F O	95 5F _	111 6F o	127 7F □

Use this to compare hex values to symbol (ASCII) values.



Star Micronics is a global leader in the manufacturing of small printers. We apply over 50 years of knowhow and innovation to provide elite printing solutions that are rich in stellar reliability and industry-respected features. Offering a diverse line of Thermal, Hybrid, Mobile, Kiosk and Impact Dot Matrix printers, we are obsessed with exceeding the demands of our valued customers every day.

We have a long history of implementations into Retail, Point of Sale, Hospitality, Restaurants and Kitchens, Kiosks and Digital Signage, Gaming and Lottery, ATMs, Ticketing, Labeling, Salons and Spas, Banking and Credit Unions, Medical, Law Enforcement, Payment Processing, and more!

High Quality POS Receipts, Interactive Coupons with Triggers, Logo Printing for Branding, Advanced Drivers for Windows, Mac and Linux, Complete SDK Packages, Android, iOS, Blackberry Printing Support, OPOS, JavaPOS, POS for .NET, Eco-Friendly Paper and Power Savings with Reporting Utility, ENERGY STAR, MSR Reading, *future*PRNT, StarPRNT... How can Star help you fulfill the needs of your application?

Don't just settle on hardware that won't work as hard as you do. Demand everything from your printer. Demand a Star!

Manual Version	Release Date
1.0	Aug 2011

### Star Micronics Worldwide

Star Micronics Co., Ltd.  
536 Nanatsushinya  
Shimizu-ku, Shizuoka 424-0066  
Japan  
+81-54-347-2163  
<http://www.star-m.jp/eng/index.htm>

Star Micronics America, Inc.  
1150 King Georges Post Road  
Edison, NJ 08837  
USA  
1-800-782-7636  
+1-732-623-5500  
<http://www.starmicronics.com>

Star Micronics EMEA  
Star House  
Peregrine Business Park, Gomm Road  
High Wycombe, Buckinghamshire HP13 7DL  
UK  
+44-(0)-1494-471111  
<http://www.star-emea.com>

Star Micronics Southeast Asia Co., Ltd.  
Room 2902C. 29th Fl. United Center Bldg.  
323 Silom Road, Silom Bangrak, Bangkok 10500  
Thailand  
+66-(0)-2-631-1161 x 2  
<http://www.starmicronics.co.th/>