GeBE's Windows CE Printer Driver
"GeBE Tiny Driver"
a driver for use on Windows CE 4, WinCE5, WinCE6

User Handbook

GeBE Elektronik und Feinwerktechnik GmbH
Beethovenstr. 15 / D-82110 Germering / Germany / www.gebe.net
Phone:++49 (0) 89/894141-0 / Fax:++49 (0) 89/8402168 / E-Mail: sales.ef@gebe.net

Contents

1. Introduction

CE_TinyDriver is the GeBE printer driver for GeBE's A8 (Atmel) and GeBE's NEC printers in conjunction with Microsoft Windows CE.

1.1. Naming Scheme

Initially built for use on Windows CE 5.0 devices, binaries are also available for use on Windows CE .NET (4.2). The drivers are named using the following convention:

```
Version of WinCE _ Name of the Driver _ Architecture .dll
```

Table 1.1 illustrates some examples of the naming scheme for the driver. Please note, the names of the files can be changed to anything that Windows CE supports. Changing the driver's filename is encouraged.

Table 1.1.: Naming Schemes

filename	type	OS	CPU
CE50_TinyDriver_ArmI.dll	driver	WinCE 5.0	ARM
CE42_TinyDriver_ArmI.dll	driver	WinCE .NET 4.2	ARM
Flash_armI_1.31.cab	installer		ARM

1.2. Required Registry Keys

Most of the registry settings in HKLM\Drivers\BuiltIn\GEP are needed for the driver to function. The registry entry HKLM\Drivers\BuiltIn\GEP\DLL, of type string, with default value "CE5_TinyDriver_ArmI.dll", should be updated to point to the location of the driver should the file be installed somewhere outside of the PATH. Section 3.2 on page 10 discusses the registry keys and values.

1.3. Printer Driver and Port Monitor

GeBE's CE_TinyDriver performs the work of both a port monitor and a printer driver. When the programmer calls StartDoc, the driver could ask the printer for its status. This approach may work and perhaps even work well so long as the status can be quickly obtained from the printer. Although it may work, this approach has its issues. The program and Windows CE wait for the driver before continuing. Meanwhile, the driver is waiting for the printer to respond. Worse than not responding quickly is when the printer does not reply at all (printer has been disconnected from the port or has been turned off). Under this approach, the entire system would seem to hang, waiting indefinitely for an answer that has not come.

The current driver, CE_TinyDriver, does not ask the printer directly for its status upon handling the context commands (StartDoc, StartPage, and so on). Rather, the driver queries the driver's internal port monitor to obtain the last status events from the print. These events are timestamped and provide the data needed by the developer of the driver to know that status of the printer.

The status can be obtained more quickly and a status can be guaranteed, even if the printer has gone off-line.

The current driver needs to be started in Windows CE by the program device.exe as a built-in device. This is necessary for the driver to monitor the port. This allows the driver to continually monitor the port for incoming status events from the printer.

The size of the driver is around 34KB for WinCE 5 and around 56KB for WinCE .NET 4.2. The previous version (without the port monitor-like functionality) was around 17KB for WinCE .NET 4.2.

2. Installation

The driver consists of a file containing the driver itself and a set of registry settings required by the driver. Installation can be performed manually or through an automated installer.

2.1. Manual Installation (.dll)

Manual installation requires that the driver .dll file is copied to a desired location on the Windows CE machine (such as in \Windows) and that the registry settings required by the driver are created and configured (using, for example, the Windows CE Remote Registry Editor program). The registry settings are discussed in Section 3.2.

2.2. Installation through an Automated Installer (.cab)

The driver may be distributed as part of an installer. The installer creates the registry entries as well as copies the driver to the installation location. Also, data is added by the installer so that the driver can be removed through Remove Programs in the Control Panel. For this, the installer needs to be opened from within Windows CE. The file could be copied to the device, usually via ActiveSync.

2.2.1. Application Manager

Other than the DLL file itself, using the installer adds at least two other files to the installation. The names of the files depend on the settings in the installer but usually the Application Manager is instructed by the installer to use file names following “CompanyName AppName” format. (For example, the files GeBE_TinyDriver_1.28.DAT (found in the AppMgr sub directory of the \Windows directory) and GeBE_TinyDriver_1.28.unload (found in the \Windows directory) were created by the Application Manager while using the Installer Wizard during a standard installation.)

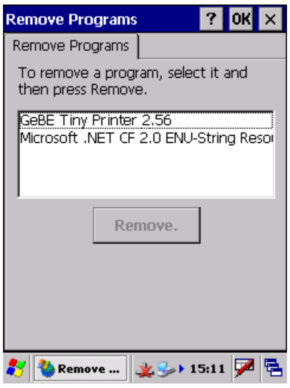
2.2.2. Known Issues

During installation using the installer, the user is asked for the location into which the driver should be installed. The default directory is \Windows. The user, however, may select any other folder on the system. A known issue of the installer is that the corresponding registry entries are not updated to reflect the driver being outside of the path.

In the case that the driver is located outside of the PATH (ie \Windows), the registry keys DLL and Driver need to be edited to point to the driver with full path (ie \FFDISK \GeBE \Drivers \CE5_Tiny_ArmI.dll).

2.3. Upgrading

Upgrading refers to installing a newer version of the driver than the one currently on the system. In both cases (using the .cab installer or copying the .dll manually), it would be best to remove the previous driver. The installer itself is simple and does not handle complex cases well. Removing the previous driver assures that the file is not in use by Windows CE and therefore cannot be removed.



To remove a program, select it and then press Remove.

GeBE Tiny Printer 2.56
Microsoft .NET CF 2.0 ENU-String Reso

Figure 2.1.: the driver's entry in Remove Programs

2.4. Removing

The uninstaller can be used to remove what can be removed of the driver. This typically just removes the entry from the "Remove Programs" control panel as the uninstaller has difficulty removing files in use. One tactic is to rename the driver file then reboot so that it is not loaded upon restart and can be deleted or overwritten. The issue is that the .dll file is being used by WinCE when WinCE starts (if the needed registry keys and the file are where they need to be) even if your program has not been opened nor called the driver.

Uninstalling the driver through the "Remove Programs" control panel may result in an error claiming certain files cannot be removed - because they do not exist or because they are in use. Resetting the machine via a Cold Reset, depending on the Windows CE configuration, may remove the driver from \Windows by restoring the system directory to its default state.

3. The Driver

3.1. Basic Overview

The GeBE TinyDriver, the printer driver for use on Windows CE .NET and Windows CE 5.0 in conjunction with a printer from GeBE's A8 series of printers. Other than the normal commands for interacting with the driver (AbortDC, CreateDC, StartDoc, StartPage, EndPage, EndDoc, and DeleteDC), the driver provides a few other methods: getting the printer's status (for printers which provide this data) and resetting the printer (either using a software command or using hardware via RTS). Also, it is always good to "cleanup" after a DC command failed. For example, if both CreateDC and StartDoc were successful, but StartPage failed, one should call EndDoc and DeleteDC, to clean up the previously successful commands.

1. Get a Device Context
   The device context is obtained through the CreateDC function.

   ```
   printDC.CreateDC( s_szDriverFilename, s_szDriverFilename, s_szPrinterPort, &dm )
   ```

2. Set an AbortProc Callback
   A callback abort function should be set even if the developer does not intend for the user to terminate the print job.

   ```
   SetAbortProc( printDC, AbortProc )
   ```

3. (optional) Display a Cancel Dialog
   The developer may display a dialog allowing the user to cancel the print job. Creating and using the cancel dialog is not covered in this document.

4. Call StartDoc
   The application starts a print job using StartDoc.

   ```
   printDC.StartDoc( &di )
   ```

5. Call StartPage for multiple page documents
   StartPage is called to signal the start of a page.

   ```
   printDC.StartPage()
   ```

6. Perform the Printing
   The application is ready to print and should perform the tasks desired. Now is the time to "paint" on the device context. One could also use the driver to check the printer's status before attempting to print anything. Functions such as DrawText and Rectangle are some of those to be used to "paint" on the context.

7. call EndPage for end of page of multiple page documents
   EndPage should be called to signal the end of a page in multiple page documents.

   ```
   printDC.EndPage()
   ```

During the processing of the print job, the printer could have sent its “out of paper” status code. Upon a call to EndPage, the driver would indicate if an error has occurred (such as running out of paper before the completion of the print job). The function GetLastError should then be called to obtain the error code which would contain the reason why the previous call failed.

8. end the print job with EndDoc
   EndDoc signifies the end of the print job. As with the call to EndPage, in the event that the function returned a value indicated that there was an error, GetLastError could be used to obtain that error.
9. (optional) remove the cancel dialog
   If a cancel dialog was presented now would be the time to remove it. Considering the printing has completed (successfully or unsuccessfully), there is no longer anything to cancel.
10. clean up by removing the device context

3.2. Registry Entries

3.2.1. Required Registry Settings

The printer driver, CE_TinyDriver.dll, has been implemented as a stream interface driver and is meant to be opened by the Device Manager (device.exe). As such, a few registry settings are required in order for the driver to function.

Table 3.1.: Required Settings in HKLM\Drivers\BuiltIn\GEP

Setting	Type	Value
DLL	REG_SZ	“\FFSDISK\GeBE\CE5_TinyDriver.dll”
Prefix	REG_SZ	“GEP”
FriendlyName	REG_SZ	“GEP”
Flags	REG_DWORD	0
Index	REG_DWORD	1
Order	REG_DWORD	8

The setting “DLL” points to the DLL file containing the printer driver. The full path is used when the file is located outside of the PATH (ie outside of \Windows). The rest of the settings correspond to using stream interface drivers with the Device Manager. Other than “DLL” containing the full path to the file containing the library, these few settings should not be changed.

3.2.2. Settings used to configure the Driver

Parity, StopBits, and XonXoff The settings “Parity”, “StopBits”, and “XonXoff” modify the connection settings. The possible “Parity” values and their meanings are as follows: No Parity (0), Odd Parity (1), Even Parity (2), Mark Parity (3), and Space Parity (4). The default is no parity (0).

The possible “StopBits” values and their meanings are as follows: 1 stop bit (0), 1.5 stop bits (1), and 2 stop bits (2). The default is two stop bits (2). Note that Windows CE does not support a setting of zero stop bits.

There are two possible values for the “XonXoff” setting. A value of 1 means that the feature is enabled (ie that XON/XOFF should be used) while a value of 0 means that XON/OFF should not be used. The default is to use XON/XOFF software handshaking (1).

Table 3.2.: Configurable Settings in HKLM\Drivers\BuiltIn\GEP

Setting	Type	Value	Possible Values
Parity	REG_DWORD	0	0, 1, 2, 3, 4
StopBits	REG_DWORD	2	0, 1, 2
XonXoff	REG_DWORD	1	0, 1
PrintTimeout	REG_DWORD	800	
ResetTime	REG_DWORD	1100	
ResetWaitTime	REG_DWORD	200	
NumWakeUpBytes	REG_DWORD	0	
NumFlushRetries	REG_DWORD	11	
FlushRetryWait	REG_DWORD	1	
PortBufferSize	REG_DWORD	7	1-2048

PrintTimeout The setting "PrintTimeout" defaults at 800 ms. Any reasonable value could be assigned to this setting. The setting is in terms of milliseconds. A larger number, for example 1600 ms (1.6 seconds), corresponds to a longer wait (in this case twice as long as the default) before timing out while waiting for a print job to complete. A smaller number, for example 400 ms (4/10 of 1 second), corresponds to a shorter wait (in this case half as long as the default). Although 800 ms should be fine in most circumstances, integrators may need to tweak the setting to their specific application, requirements, and conditions.

ResetTime The setting "ResetTime" is the time in ms used to reset the printer using the RTS line. The default value of 1100 refers to putting the RTS to low for 1100 ms, to induce a reset, then high again.

ResetWaitTime The setting "ResetWaitTime" is the time in ms to wait after the reset before moving to the next task. This value should be large enough to allow time for the printer to initialize itself. The default of 200 means that the driver will wait 200 ms after a reset before continuing the driver's normal operations.

NumWakeUpBytes The "wakeup" feature can be configured through the "NumWakeUpBytes" setting. The feature can be explicitly disabled by setting the value to 0. If the entry is missing, the driver defaults to disabling the "wakeup" feature. Specifying any possible value both enables the "wakeup" feature and defines the number of "wakeup" bytes to send to the printer. The "wakeup" bytes are sent one at a time, one after another, to the printer for each and every creation of the device context.

The number of "wakeup" bytes required will vary from configuration to configuration and depend on things such as the baudrate in use. A setting of 100 "NumWakeUpBytes" may be a good starting point for testing the number required for a particular setup.

NumFlushRetries The setting "NumFlushRetries", with a default value of 11, is the number of times the driver should attempt to write its internal buffer to the physical port, in cases where the printer did not return and error message and neither did WinCE.

FlushRetryWait The setting "FlushRetryWait", with a default value of 1, corresponds to the number of milliseconds to wait before continuing with the next retry. With a higher positive value, the WinCE system is able to work on other tasks before returning to the *Flush Retry Loop*.

PortBufferSize The setting "PortBufferSize", with a default value of 7, sets the number of bytes the driver should send to the port. This setting has an effect on print speed because it affects how fast the printer receives data. The default of 7 is purposely lower than should be needed. A more reasonable value is something closer to 64.

3.2.3. Other, Related Keys

For developers to use the driver, the keys mentioned in section 3.2.1 were required while the ones mentioned in section 3.2.2 allowed modifying the driver's settings. The keys in HKLM\Printers and HKLM\Ports are not needed for a developer to use the CE_TinyDriver printer driver. Instead, the keys in HKLM\Printers and HKLM\Ports define a standard printer and port, which programs like WordPad use to list available printers and ports.

HKLM\Printers

Table 3.3.: Related Registry Settings in HKLM\Printers\GeBE PrinterName

Setting	Type	Value
Driver	REG_SZ	"\FFSDISK\GeBE_DriverFiles\CE5_TinyDriver.dll"
High Quality	REG_SZ	"200"
Color	REG_SZ	"Monochome"
Draft Quality	REG_SZ	"50"

The SubKey "GeBE PrinterName" (under HKLM\Printers) refers to the name given to the printer. Typically GeBE's product names, for example "GeBE Flash", are used but customers could customize the SubKey to a name of their choosing. The "Driver" setting contains a string that points to the location of the DLL file containing the printer driver. The full path should be used when driver sits outside of the PATH (ie somewhere other than in \Windows). The default value set by the installer is of just the file name without any path data. See section 2.2.2 regarding this issue.

The setting "High Quality" is required and denotes the resolution in high-quality mode. The value of 200 is used as the default value.

The setting "Color" indicates whether the device can print in color or not. The setting needs to be set to "Color" for color to be enabled in the Print DialogBox. The default value is "Monochrome".

The setting "Draft Quality" need not be present but when present it indicates the resolution in draft-quality mode. Not all printers support a draft-quality mode. The default is a value of 50.

HKLM\Ports

Table 3.4.: Global Printer Registry Settings in HKLM\Ports

Setting	type	value
Port5	REG_SZ	"COM1: 115200"
Port6	REG_SZ	"COM1: 9600"
Port7	REG_SZ	"COM3: 38400"
Port8	REG_SZ	"COM3: 9600"
Port9	REG_SZ	"LPT1"

The "Ports" SubKey contains a list of settings and their values where the values indicate the ports. The names of the settings are the word "Port" followed by a number. It is important that the needed port is in the list. In table 3.4, the value "COM1: 115200" is assigned to setting "Port5". The value could have just as easily been assigned to a setting "Port9" or any other similarly named value (such as "Port45"). The value "COM1: 115200" refers to a serial connection on COM1 at 115200 baud.

3.3. The Driver's Application Programming Interface (API)

Using the driver's application programming interface, the printer's status can be obtained and the printer can be reset. These two requests can be made either using the RTS line[1] or by sending normal commands to the printer.

---

[1] Please see the A8 Printer manual. For the printer to accept status requests and reset requests over the RTS line, the option needs to be enabled on the printer.

3.3.1. Request Printer Reset

As mentioned, the printer can be reset using the API, using either RTS or not. Fulfilling reset and status requests via the RTS line needs to be activated in the printer’s firmware before it can be used. Also, in the case when the printer’s input buffer is full, sending another command, that of requesting a printer reset, will go unfulfilled since the input buffer is already full. Requesting a reset over the RTS line provides a means to reset the printer even when the printer’s buffer is full. [2]

Table 3.5.: TinyDriver API: ResetPrinter

BOOL ResetPrinter(PWSTR sPort, BOOL bRTSReset, int iTimeout)	
	Request a reset of the printer
PWSTR sPort	Port Name and Baudrate (i.e. “COM3: 115200”)
BOOL bRTSReset	TRUE to use RTS; FALSE to use a command
int iTimeout	Milliseconds to wait before a timeout
Return values:	
TRUE	completed successfully
FALSE	terminated with an error

---

[2] Resetting the printer not using an RTS reset should work correctly for both GeBE’s NEC and GeBE’s Atmel based printers. Using an RTS reset is known to work on GeBE’s Atmel based printers and may also be supported with future versions of the NEC based printers and future releases of the printer driver.

3.3.2. Request Printer Status

The printer's status can be obtained from the printer through the driver either by using the RTS line or by sending a command to the printer. Again, when the printer's input buffer is full, incoming commands are lost. [3]

Table 3.6.: TinyDriver API: UpdatePrinterStatus

BOOL UpdatePrinterStatus(PWSTR sPort, BOOL bWithRTS, int iTimeout)	
	Request the printer to send its current status
PWSTR sPort	Port Name and Baudrate (i.e. "COM3: 115200")
BOOL bWithRTS	TRUE to use RTS; FALSE to use a command
int iTimeout	Milliseconds to wait before a timeout
Return values:	
TRUE	completed successfully
FALSE	terminated with an error

---

[3] This API function should work correctly for both GeBE's NEC and GeBE's Atmel based printers.

3.3.3. Obtain Latest Status Messages

The last few status messages from the printer can be obtained through the driver's internal buffer. Once the driver's internal status message buffer is full, older status messages are lost as newer messages arrive. Also, the messages are cleared from the driver's cache upon being obtained through the function GetPrinterStatus.

The function can be called in such a way as to fill the buffer with status bytes but it can also be called in such a way as to provide the amount of space needed for the available data. The size of the available data is returned by the function GetPrinterStatus when the buffer is NULL. In this case, no bytes should be requested (the third parameter should be 0). In this particular case the function should just return the size of the data available.

Table 3.7.: TinyDriver API: GetPrinterStatus

int GetPrinterStatus(PWSTR sPort, sPrinterStatus* pBuffer, int iLength)	
	Obtains the latest status messages from the driver's internal buffer
PWSTR sPort	Port Name and Baudrate (i.e. “COM3: 115200”)
BYTE* pBuffer	Pointer to a buffer, into which the status bytes will be written
int iLength	Maximum length of the buffer, in bytes
Return values:	
-1	terminated with an error
not -1	number of bytes written

Table 3.8.: TinyDriver API: sPrinterStatus struct

```
struct sPrinterStatus
{
Byte bySec
Byte byMin
Byte byHour
Byte byDay
Byte byMonth
Word wYear
Byte byStatus
}
```

3.3.4. Stopping the Driver from monitoring the Port

Starting with Version 2 of the driver, the driver can free the port. In order to receive incoming data, the driver would attach itself to the port then record the data as it arrived. Sometimes though, it is desirable to interface with the port directly after having loading the port. Version 2 of the driver can be forced to leave the port with the following API function but it does so on its own after a timeout period.

Table 3.9.: TinyDriver API: StopPortMonitor

BOOL StopPortMonitor(PWSTR sPort)	
	Request a reset of the printer
PWSTR sPort	Port Name and Baudrate (i.e. “COM3: 115200”)
Return values:	
TRUE FALSE	completed successfully terminated with an error

3.4. External Syncronization

Upon a call to EndDoc, the driver searches for the ETX in its internal buffer. If not found, it looks again 100 ms later. It will do this until either the ETX (3) has been found or the ETX has not been found within the timeout "PrintTimeout" time (defaults to 800ms). Although there are two documents (files) to print, the test program apparently splits the text into several “documents” (as noted by StartDoc and EndDoc).

As is, the price to pay to know that an item sent to the printer printed is the current waiting for the 3 time come back from the printer. In our tests here, with the printer at 4.6V, the 3 would be received around 250 ms after being sent to the printer. One way to better use the driver as is, would be to split the document into StartPage and EndPage calls. This should mean just one StartDoc call and one EndDoc call for the entire document, but has the disadvantage that StartPage/EndPage do not provide a means to the STX and ETX. Another suggestion would be to split the item to be printed into bigger chunks, calling StartDoc and EndDoc less often.

3.5. Waking up Printers in Sleep Mode

The driver can be used to “wake up” printers from low power, sleep mode. The feature is configurable through the registry entry NumWakeUpBytes, as explained in the table 3.2.2.

3.6. GeBE Driver Demo Program

A program has been created by GeBE to demonstrate some fundamentals with using printer drivers. The driver demo program may be extended but is meant more just as reference of how to interact and use the driver.

3.6.1. Source Code

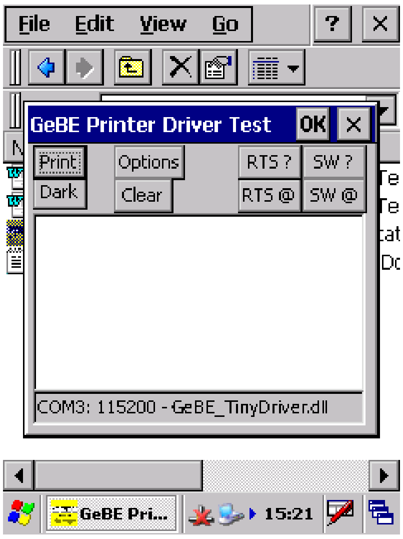
The project was built with Microsoft Embedded Visual C++ 4 SP4. One should also be able to import the source files, if not the project itself, into MS Visual Studio 2005.

3.6.2. Interface

The driver demo program has a rather simplistic interface. Nonetheless, the program allows one to verify the connection settings and that the driver has been installed properly. The interface is broken into two parts, a main window and the options window for setting driver and/or connection settings.

Main Window

Across the top of the main window are several buttons. In the middle is a ListBox which gets filled with status events both from the printer and as a consequence of interacting with the program (clicking one of the buttons, for example).



COM3: 115200 - GeBE_TinyDriver.dll

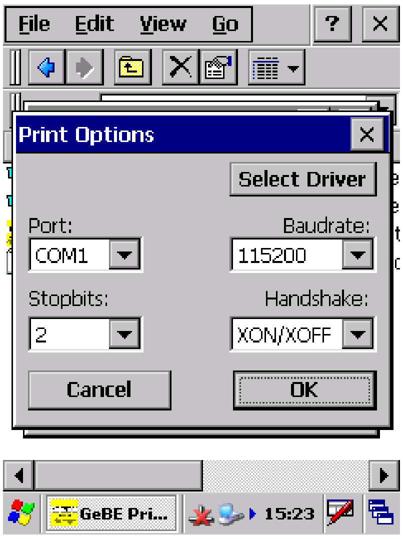
Figure 3.1.: GeBE's Driver Demo Program: Main Window

As mentioned in table 3.10, clicking the "Clear" button has the effect of replacing the contents of the ListBox with the word "Clear". Clicking the "Clear" button when only the word "Clear" appears in the ListBox will empty the ListBox of its content. Double-clicking an item in the ListBox requests that the contents of the ListBox (all of the status messages) be saved to a file stored at the root of the Windows CE device (ie to the directory \status_debug_list.txt).

Print Options Window

Table 3.10.: Buttons of the Main Window

Title	Description
Print	sends a small test print to the printer
Dark	sends a rectangles which have been filled in (in order to overheat the printer)
Options	brings up the options window where one can set the driver and connection settings
Clear	clears the status ListBox (replacing it with the word ”Clear”
RTS ?	using the hardware RTS line, asks the printer for its current status
RTS @	using the hardware RTS line, instructs the printer to restart
SW ?	using a software command, asks the printer for its status
SW @	using a software command, instructs the printer to restart



Port: COM1
Baudrate: 115200
Stopbits: 2
Handshake: XON/XOFF

Figure 3.2.: GeBE’s Driver Demo Program: Options Window

Table 3.11.: Interface of the Options Window

Title	Description
Select Driver	sends a small test print to the printer
Port	sends a rectangles which have been filled in (in order to overheat the printer)
Stopbits	brings up the options window where one can set the driver and connection settings
Baudrate	selects the baudrate
Handshake	selects the type of handshake
Cancel	closes the Options window without saving changes
OK	closes the window while saving the settings

4. Miscellaneous

4.1. Frequently Asked Questions, with Answers

4.1.1. What is the window that is shown with the first print using the driver?

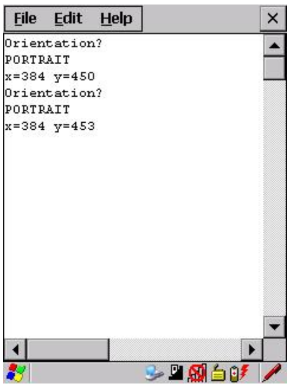
A small console window (image 4.1 on page 21) was used by earlier versions of the TinyDriver. Closing the window would harm neither the performance nor the functionality of the driver. Versions of the driver released after the beginning of March 2008 no longer opened the console window. Utilizing a more recent version of the driver would rid one of the previous driver's console window.

4.1.2. Sometimes my printouts begin with characters not in the document to be printed. How does one get rid of this *garbage*?

Image 4.2 (on page 22) illustrates an example of this issue. In this case, the text 't5' appears before the desired output. Effects such as this most often occur when the data was sent to the printer while the printer was still in *sleep mode*. The beginning few pieces of data to the printer are lost as the printer switches back out of *sleep mode*.

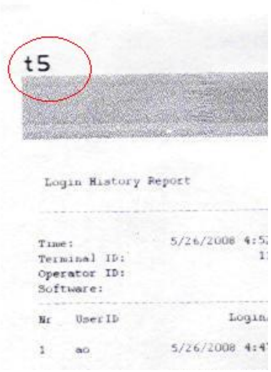
Versions of the driver built after the beginning of April 2008 contain a feature to send data to a printer in *sleep mode* to "wake" it before sending the data to be printed. Initially 0x00 was sent 137 times to bring the printer out of *sleep mode*. Builds of the TinyDriver after May 2008 allow users to configure the "wakeup" feature and default to disabling the feature. Via a setting in the registry, the "wakeup" bytes can be explicitly disabled or explicitly enabled and the quality of bytes to be sent specified. Refer to table 3.2 on page 11 for more information regarding the "wakeup" feature and registry setting.

Rather than using a more recent version of the driver, a workaround is to implement the sending of bytes used for "waking" the printer in the program itself. One need only send a small document of nothing more than a few bytes of zeros to the printer (directly or through the driver) before sending the normal job to be printed.



```
Orientation?
PORTRAIT
x=384 y=450
Orientation?
PORTRAIT
x=384 y=453
```

Figure 4.1.: Driver's Console Window



```
t5

Login History Report
-------------------------------------------
Time:                     5/26/2008 4:5
Terminal ID:                          1
Operator ID:
Software:
-------------------------------------------
Nr    UserID                       Login

1     ao                  5/26/2008 4:4
```

Figure 4.2.: *Garbage* before the Printout

4.1.3. How can a document be printed that is wider than it is long, without the printout being rotated into landscape mode?

The solution is simple: use a more recent version of the driver. The effect of the driver ignoring the program's orientation (landscape/portrait) settings, always rotating the document automatically when the width is larger than the height is a known issue with the older drivers and one that has been eliminated with newer versions.

List of Figures

List of Tables

A. Code - GetPrinterStatus

```
/* request the size of available status data */
int ret = pGetPrinterStatus((PWSTR)s_szPrinterPort, NULL, 0);

/* add data to list */
tmpCString.Format(L"pGetPrinterStatus NULL 0 returned: %X",ret);
this->AddToList(tmpCString);

if (ret == -1)
{
 /* an error with the request has occurred */
}
else if(ret >= 0)
{ /* request for the size of available data was successful */
 if(ret > 0)
 { /* more than 0 pieces of data are available */

 /* store the number of status units available in the variable count */
 int count = ret/sizeof(sPrinterStatus);
 /* add data to list */
 tmpCString.Format(L"# SPrinterStatus: %X", count);
 this->AddToList(tmpCString);

 /* create a variable pStatus large enough to hold all of the available data */
 sPrinterStatus* pStatus = new sPrinterStatus[ret];
 memset(pStatus, 0, ret);

 /* request the status data, to fill the pStatus buffer with no more than "ret" amount of data */
 ret = pGetPrinterStatus((PWSTR)s_szPrinterPort, pStatus, ret);

 /* add data to list */
 tmpCString.Format(L"pGetPrinterStatus pStatus, ret: %X", ret);
 this->AddToList(tmpCString);

 for(int i=0;i<count;i++)
 {
 tmpCString.Format(L"%02d.%02d.%04d %02d:%02d:%02d
 this->AddToList(tmpCString);
 }
 delete pStatus;
}
}
 FreeLibrary(hDll);
```

Figure A.1.: Code Snippet, Using GetPrinterStatus