



Freefloat CELinq

version 2.7

User's Manual



| | |
|---|-----------|
| Contacting Freefloat | 3 |
| Introduction..... | 4 |
| A Typical Use Case | 4 |
| Supported Platforms..... | 5 |
| Getting Started | 6 |
| A Typical Installation | 7 |
| Registering CELinq..... | 9 |
| Configuration..... | 13 |
| CELinq Configuration File (CELinq.txt) | 13 |
| Serial Configuration File (Serial.txt)..... | 14 |
| Scripting in CELinq | 15 |
| Overview | 15 |
| Compile & Runtime Errors | 16 |
| The Lua Language..... | 17 |
| The GS1 Function, ean128(data, strict) | 18 |
| Sample Scripts | 19 |
| Remove text from the start | 19 |
| Remove text from the end..... | 19 |
| Add text at the start..... | 19 |
| Add text at the end | 20 |
| Match certain data..... | 20 |
| Replace a character..... | 20 |
| Appendix A. Lua Copyright..... | 21 |
| Appendix B. Version History..... | 21 |

Copyright © 2005–2009 Freefloat

Contacting Freefloat

Office

Freefloat
P.O. Box 13101
S-402 52 Gothenburg
SWEDEN

Support and Sales Information

| | |
|--------|---|
| Forum | http://forum.freefloat.com/ |
| Web | http://www.freefloat.com/ |
| E-mail | info@freefloat.com |

Introduction

Freefloat CELinq is a wedge software for Windows CE. CELinq captures data from a serial port or built in scanner in some cases, filter and/or modifies the data before sending it to the active application.

The data processing is done using a powerful script language called Lua. CELinq also has extensive support for GS1 barcodes (EAN128).

A Typical Use Case

Project Forklift PC

A company decided to use forklift computers in their warehouse. The computers should be used to wirelessly access their Warehouse Management System via a thin client. During different processes barcode readers identify goods by reading pallet labels.

An analysis showed that pallet labels from different manufacturers did not contain the same information. In some cases information was stored in one barcode, in other cases two or more barcodes was used. In many cases, information needed to be filtered away, some information needed extraction and reordering to fit the input forms of the WMS system.

The first attempt to solve this problem was by configuring the barcode readers. But the scanner was not flexible enough to handle all the variants, and even if it did, the customer did not want to lock themselves in with barcode reader with a particular brand and model.

The manufacturer of the WMS system were asked if it was possible to make adjustments to their system. It was possible, but the costs were high and the manufacturer recommended against it because it would incur costs when upgrading the system and make it difficult to maintain in the future.

The solution to the problem, CELinq, proved to be simple and relatively inexpensive. After a thorough analysis a script was developed to handle the ten different cases of barcode data processing. The barcodes used in this project adhered to the GS1 standard and the built-in support for GS1 in CELinq made it a lot easier to build the script.

Supported Platforms

- ARMV4/ARMV4I processor based devices, tested on Windows CE 4.1 to 5.0 and Pocket PC 2003 to Windows Mobile 5.0
- x86 processor based devices, tested on Windows CE 4.1 and Windows CE 5.0
- Windows 2000/XP/XPe

The generic version uses a serial input module and works most Windows CE devices that has a physical or virtual serial port, for example a serial profile for Bluetooth.

The version for Windows 2000/XP/XPe is primarily used for developing and testing a script/configuration.

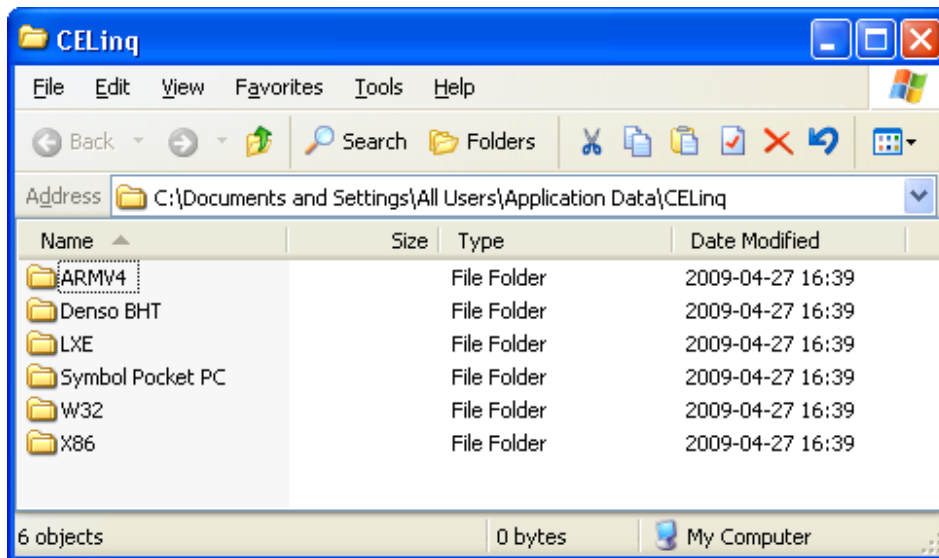
Getting Started

When you install CELinq, shortcuts are created on the Start menu.



Here you can view this manual, or browse the folder containing CELinq program files and configuration.

Select **Browse Application Folder** on the **Start** menu to open a **Windows Explorer** window displaying the program folders:



| Folder | Contents |
|------------------|--|
| ARMV4 | Generic serial version for Windows CE devices using a ARM family processor |
| X86 | Generic serial version for Windows CE devices using a x86 family processor |
| Denso BHT | Device specific version with scanner module, tested on BHT-200, BHT-400, and BHT-700 |
| LXE | Device specific version with scanner module, tested on MX8 |
| Symbol Pocket PC | Device specific version with scanner module, tested on MC50 and MC70 |
| Datalogic Falcon | Device specific version with scanner module, tested on Falcon X3 |
| W32 | Generic serial version for standard Windows |

A configuration and installation of CELinq on a Windows CE device consists of the following steps:

- Browse to the folder containing the files corresponding to your device platform
- Configure settings for the serial port or the settings for the built-in scanner, whichever applies to you device
- Optionally configure CELinq main settings
- Write a script if needed, the default script passes barcode data through unmodified
- Copy the files to the device, using ActiveSync or a USB memory stick
- Start CELinq and test the installation

A Typical Installation

Let's say you have Intel x86 Windows CE device and a serial barcode reader connected to COM1 of the device. The barcode reader is configured to send data in 19200 bps.

The variant of CELinq for the x86 platform has two configuration files, CELinq.txt and Serial.txt.

CELinq.txt contains very few settings that you normally don't need to modify. Serial.txt contains settings for input (serial port settings).

The default speed setting in Serial.txt is 9600, in this case you need to change the speed to 19200. Go to the folder for the x86 platform by using **Browse Application Folder** on the start menu. Then navigate into the X86 folder. and open the file Serial.txt in your favorite text editor (or Notepad). The file should look like this:

```
// Serial port. COM1, COM2, etc.
//
// For Windows CE the port name must include the colon, i.e. "COM1:"
Port=COM1:

// Communication speed. 110, 300, 600, 1200, 2400, 4800,
// 9600, 19200, 57600, and 115200.
Speed=9600

// Data bits. 7 or 8
Data=8

// Stop bits. 1 or 2
Stop=1

// Parity. None, Odd, Even, Mark, Space
Parity=None

// Dtr. Disable, Enable, Handshake
Dtr=Handshake

// Rts. Disable, Enable, Handshake
Rts=Handshake

// XonXoff. Indicates whether XON/XOFF flow control is used during data
reception. Disable, Enable
XonXoff=Disable

// The character that the input module uses to determine the end of a string
InputTerminator=<cr>

// Timeout is an alternative way of determining the end of a string.
// A sensible value is 100. The value's unit is milliseconds.
// If a timeout is specified, InputTerminator is ignored.
Timeout=0
```

Change the value for the Speed setting to

```
Speed=19200
```

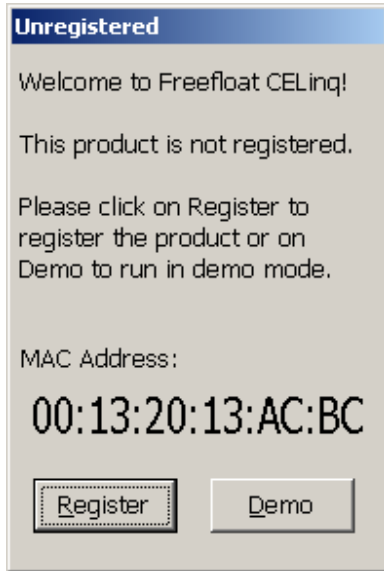
and save the file.

Now transfer all files in the X86 folder to the Windows CE device. Place them in a folder on a flash disk so they won't disappear if the Windows CE device is rebooted. Please see the documentation for the Windows CE device for instructions on how to transfer files to it and what folders that are persistent.

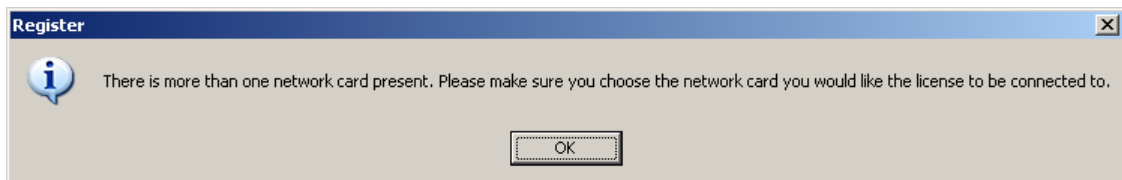
When the files have been installed, start *CEInq.exe*.

Registering CELinq

When CELinq is started on the CE device for the first time it is unlicensed. If you have only one network card in the CE device the following dialog box is displayed:



If you have more than one network card in the CE device the following message and dialog box are displayed:

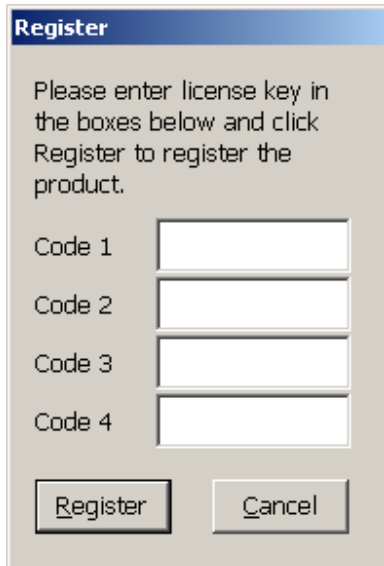




To license CELinq, write down the MAC address. If you have more than one network card, select one of them, preferably the built-in network card. Then contact your reseller or Freefloat to buy a license.

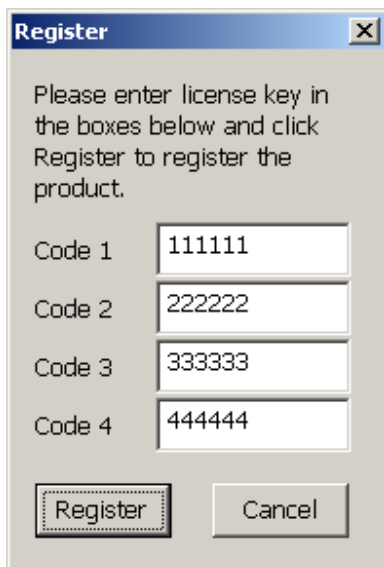
Until you have received the license key, you can run CELinq in demonstration mode. The demonstration mode has no limitations except that it will only process 20 barcodes. After 20 barcodes you can restart CELinq to get 20 more.

When you have received the license key, select the network card in the list (if you have more than one network card) and click on the Register button in the dialog box above. The Register dialog box will be displayed:



The image shows a dialog box titled "Register". It contains the following text: "Please enter license key in the boxes below and click Register to register the product." Below this text are four input fields labeled "Code 1", "Code 2", "Code 3", and "Code 4". At the bottom of the dialog box are two buttons: "Register" and "Cancel".

Enter the digits from the license key into the boxes. Leave out the hyphens (-). If the license key you received was 111111-222222-333333-444444 your dialog should look like this:



The image shows the same "Register" dialog box as above, but now the input fields contain the digits from the license key: "Code 1" contains "111111", "Code 2" contains "222222", "Code 3" contains "333333", and "Code 4" contains "444444". The "Register" button is highlighted with a dashed border, indicating it is the active element.

Click on the **Register** button to complete the registration process.

If you made a mistake when entering the license key, you will get the following message:



Please re-type the license key correctly.

When you enter the correct license key the following message is displayed:



Configuration

This topic explains the settings in the configuration files *CELinq.txt* and *Serial.txt*.

CELinq Configuration File (CELinq.txt)

InterKeyDelay

Specifies the delay between each simulation of a key. Given in milliseconds.

InputModule

Specifies the module responsible for talking to the barcode device.

KeyboardLayout

Can be set to Swedish, Belgian or US. This setting needs to reflect the keyboard setting in the Windows CE device.

AllowMultipleInstances

If set to FALSE (default) only one instance of CELinq is allowed.

StartupDelay

If autostarted, sometimes you need to delay until the OS/system is completely initiated.
The delay is in mS

RDPMODE

Changes the method used to send characters to the active application.

WIN CE Remote Desktop Client needs this option enabled in order to see the keystrokes.

Serial Configuration File (Serial.txt)

Port

Specifies the serial port to use. Valid values are *COM1*, *COM2*, and so on.

Speed

Specifies the serial communication speed of the serial port. Valid values are 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 57600, and 115200.

Data

Specifies the number of data bits used. Valid values are 7 and 8.

Stop

Specifies the number of stop bits used. Valid values are 1 and 2.

Parity

Specifies the parity used. Valid values are *None*, *Odd*, *Even*, *Mark*, and *Space*.

Rts

Specifies the RTS line behavior. Valid values are *Disable*, *Enable* and *Handshake*.

Dtr

Specifies the DTR line behavior. Valid values are *Disable*, *Enable* and *Handshake*.

Disable: Disables (lowers) the line when the device is opened and leaves it off.

Enable: Enables (raises) the line when the device is opened and leaves it on.

Handshake: Raises the line when the input buffer is less than one-half full and lowers the line when the buffer is more than three-quarters full.

XonXoff

Indicates whether XON/XOFF flow control is used during data reception.

Valid values are *Disable* and *Enable*.

InputTerminator

The character that the input module uses to determine the end of a string. Can be normal text and/or control character mnemonics.

Timeout

Timeout is an alternative way used by the input module for determining the end of a string. The value's unit is milliseconds. A sensible value is 100. If a timeout is specified, the setting

InputTerminator is ignored.

Scripting in CELinq

Overview

CELinq has an embedded script language called Lua.

When CELinq receives data from a device the method called process in the script is called. The code in the script determines what the output will be.

The name of the script file is script.lua. When you are testing a script, remember to restart CELinq between changes to recompile the script.

Let's say you want to add the prefix ABC to the output when a barcode is read. The script method to do this is:

```
function process( data )
  return "ABC" .. data
end
```

The return value of the method process is what CELinq sends to the active application.

The default script included is:

```
function process( data )
  str = trimControlChars( data )

  return translateKeys( str .. "{Enter}" )
end
```

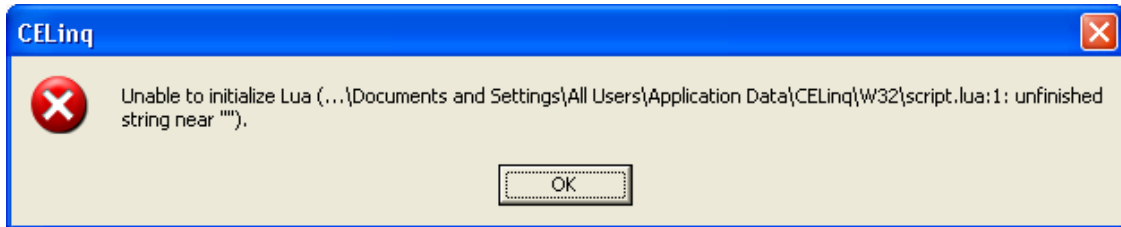
It uses two methods **trimControlChars** and **translateKeys**. The code for these are included in the default script.

trimControlChars removes any control characters sent by the barcode scanner. The reason for trimming away control characters is that these might cause problems when writing more sophisticated scripts unless they are explicitly handled. Please note that **trimControlChars** should not be used when you are using the **ean128** function to parse GS1 data.

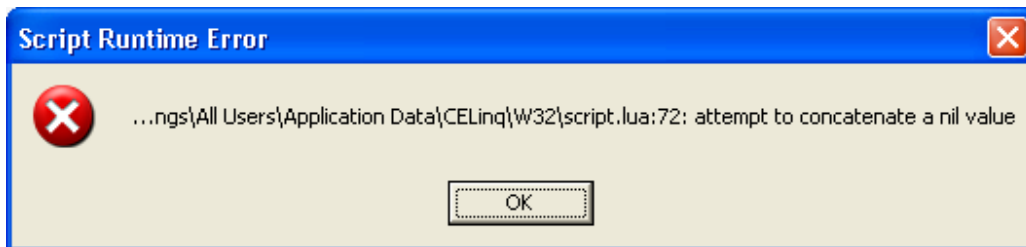
translateKeys translates key names to character codes. This method makes it easier to specify keys like Enter and Home by using a key name in the format "{Enter}" instead of having to look up a character code. All keys supported is defined in the table **keyTable** in the default script.

Compile & Runtime Errors

If you make a mistake, for example create a syntax error, an error message is displayed when the script is compiled:



Also, some errors can appear when the script is running, so called run-time errors. Here is an example:



The error messages display the path to the script, the row number where the error was detected, and an error description.

The Lua Language

From <http://www.lua.org/about.html>:

Lua is a powerful, fast, light-weight, embeddable scripting language.

Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics.

Put simply, Lua is what makes data processing in CELinq very flexible and powerful.

The reference manual for Lua can be found at the Lua site:

<http://www.lua.org/>

There is also a printed book on the Lua language, called **Programming in Lua**, which is more accessible than the reference manual.

Apart from Lua and its built-in language, CELinq expose a function for parsing GS1 (EAN128) codes.

Lua is a generic script language and has methods to manipulate string, tables, files, and so on.

The GS1 Function, ean128(data, strict)

Description

Parses the contents of a GS1-128 code (earlier called UCC-128 or EAN-128). For variable length fields followed by another field, the data must be delimited by a Group Separator (GS, ASCII 29, hex 1D).

Please refer to <http://www.gs1.org> for information about GS1 Application Identifiers.

Arguments

| Argument | Type | Description |
|---------------|---------|--|
| <i>data</i> | String | The GS1-128 data to be parsed and split into separate fields. |
| <i>strict</i> | Boolean | In strict mode, spaces are not allowed in alphanumeric fields. |

Returns

A table where the keys are the Application Identifiers (AIs) and the values are the contents of the fields. If the parsing fails, a nil value is returned. The parsing can fail if the code is not a GS1 code or if the code doesn't follow the standard.

Example

```
-- Sample code that dumps the contents of an EAN128 (GS1) barcode
function process( data )
  output = ""

  -- Parse the GS1 code
  fields = app.ean128( data, true )

  if fields then
    -- Output application identifiers and values
    for k,v in pairs(fields) do
      output = output .. "AI: " .. k .. " Value: " .. v .. string.char(13)
    end
  else
    output = "GS1 parsing failed." .. string.char(13)
  end

  return output
end
```

Sample Scripts

Below are simple examples of common tasks.

Remove text from the start

```
--Remove text from the start
function process( data )
  str = trimControlChars( data )

  -- Remove the first 3 characters from the string
  str = string.sub( str, 4 )

  return translateKeys( str .. "{Enter}" )
end
```

Remove text from the end

```
--Remove text from the end
function process( data )
  str = trimControlChars( data )

  -- Remove the last 2 characters from the string
  str = string.sub( str, 1, -3 )

  return translateKeys( str .. "{Enter}" )
end
```

Add text at the start

```
-- Add text at the start
function process( data )
  str = trimControlChars( data )

  -- Add 123 at the start of the string
  str = "123" .. str

  return translateKeys( str .. "{Enter}" )
end
```

Add text at the end

```
-- Add text at the end
function process( data )
  str = trimControlChars( data )

  -- Add XYZ at the end of the string
  str = str .. "XYZ"

  return translateKeys( str .. "{Enter}" )
end
```

Match certain data

```
-- Match certain data
function process( data )
  str = trimControlChars( data )

  -- If the string starts with 00, add a text at the end
  if string.find( str, "^00" ) then
    str = str .. "<-- MATCHED"
  end

  return translateKeys( str .. "{Enter}" )
end
```

Replace a character

```
-- Replace a character
function process( data )
  str = trimControlChars( data )

  -- Replace "0" with " Zero "
  str = string.gsub( str, "0", " Zero " )

  return translateKeys( str .. "{Enter}" )
end
```

Appendix A. Lua Copyright

Copyright © 1994-2008 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Appendix B. Version History

| Version | Changes | Date |
|---------|--|------------|
| 2.0 | Major upgrade. Added support for LXE devices. Replaced static settings (for example Prefix/Suffix) with the Lua script language to make CELinq more flexible. | 2009-05-06 |
| 2.1 | Fix for using com port larger than COM9 in WIN32 | 2009-10-15 |
| 2.2 | Removed checking of return value from SetupComm in Serial module since it always return FALSE in CE 6.0. Changed default install folder to "<System Disk>\Freefloat\Freefloat CELinq <version>" to avoid problems with Vista/Win7 UAC limitations. Version number added to install folder to enable parallel installations of different CELinq versions. | 2010-10-22 |
| 2.3 | Added support for Datalogic Falcon | 2011-02-08 |
| 2.4 | Changed the serial module to enable arbitrary portnames | 2011-03-15 |
| 2.5 | Quickfix for a customer project: {Delay} = keycode 0x0a = 100 mS sleep | 2011-04-19 |
| 2.6 | Added option "StartupDelay" | 2011-11-23 |
| 2.7 | Added option "RDPMode" | 2011-12-06 |