DS715

PRICE-CHECKER

Solid and Seamless scanner built-in
w/Utility tool software

PC-NET

Version: 1.0-44

version	*changes*
0.9	Initial version
0.91	- Added AliveTimeout option (3.2). - removed two known problems that where solved, and added two new ones. - Added chapter 1 Quick Start Guide.
1.0	- Added copyright and warranty notice - a few minor text corrections

Known Issues

The PRICE CHEKER-Control module is not yet implemented; use a telnet-client instead.
The Control interface of the PC_Ethernet_driver is incomplete; several commands will be added in the future.
The broadcast functionality from the control-interface is not yet implemented

Contents

Terms and Definitions

Price checker	Any Our company price-checkers with an ethernet interface; currently price checker.
TCP-port	Number distinguishing different sockets on the same machine.
socket	Abstraction for a network connection.
address	IP-address: number identifying a computer/device on the network.
PC-NET	Project name for the PC-Ethernet-Driver and all related applications.
DB	DataBase
C^{++}	Programming language used for the application.
kdevelop	Integrated Development Environment used for developing the application under Linux.
gcc	Gnu Compiler Collection: Compiler used with kdevelop.
visual studio 6	Microsoft Visual Studio 6: development platform for windows.

1 Quick Start Guide

What you need:

- The PC_Ethernet_Driver for your platform (Windows, Linux, ...).
- Some Database application that can communicate with the Driver (using the protocol described in chapter 3.1). This will supply the information for the price-checkers in response to the barcodes it receives. An example/test application (Price checker-demo-db) is provided (see chapter 4).
- And of course some price checker with ethernet...

Setting up the driver:

- If necessary, in the pcether.ini file, set the TCP-ports that the database and price checker's need to connect to. (see chapter 3.2)
  (If the pcether.ini file doesn't exist, you can create a default one by starting the driver and closing it again using the control-interface.)
- Make sure the price checker's use the correct server-address and -port. (The default price checker-connect port, **9101**, is the same as the default server port of the Price checker).
- Start the driver (**PC_Eth_driver [.exe]**). On Linux/Unix it is automatically started as a daemon; On windows you can set it up as a service.
- Connect a Database-application to the database-socket (default port: **4701**).

[optional] Setting up the Price checker-demo-db:

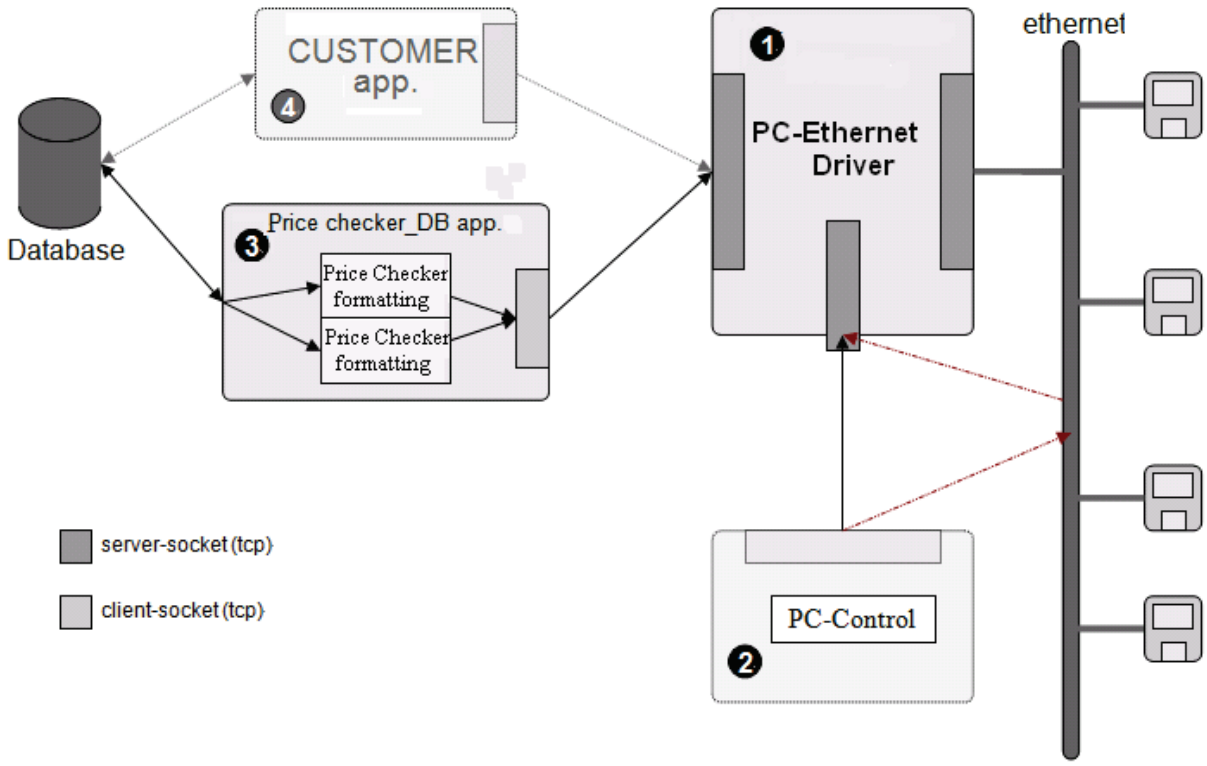
- Create a database-text-file (or edit the supplied example Price cheker_db.txt, which is also loaded as the default). See chapter 4.2 for the format of this file.
- Create (or edit) the Formatting files `"price checkerformat.txt"`. These define the screen-layout for the price checker respectively. See chapter 4.3 for more information.
- Start the demo-application (**PC_demo [.exe]**). If the address and port are different from the defaults you can specify them as parameters on the command line; see chapter 4.4 .

2 Introduction

The PC-Ethernet-Driver Package (PC-NET) facilitates the easy connection of multiple price checkers to a database through a single interface. The pc-ethernet-driver acts as a sort of multiplexer; Multiple price checker's can send a request which is then sent to the database. When an answer is received, it is send back to the correct price cheker.

This is achieved by including unique ID and Type information into the message send to the database. By including type-information, which is determined when a price checker connects, the database interface application can format the message appropriately.

The pc-package consists of three components; The PC-Ethernet-Driver forms the hart of the system that all other components connect to. The Database component provides the actual information that is requested by the price checker-devices. Controlling the PC-Ethernet-Driver is done with the (optional) PRICE CHEKER-control module or by using a telnet program.



Description of the components:

1. PC-Ethernet [Our company] : Driver that all price checker devices can connect to. Manages and monitors all connections. All interfaces use TCP-sockets for flexibility and ease of use (see the paragraph about TCP sockets).
2. Price checker-control [Our company]: User Interface for controlling and monitoring PC-Ethernet. Is included with the PC-Ethernet driver.

3. PRICE CHEKER-Demo-DB [Our company]: Application that provides an interface to the database and some basic user-functionality (controlling the display and button-actions).
   Included with the PC-Ethernet Driver (including source code). Meant as a sample application and/or a framework for developing a more extensive back-office application, it can also be used as a very simple price-checking database. It uses a plain-text file as “database”, so it will not perform well for large amounts of items! See chapter 4 .
4. Customer application : Alternatively to the PRICE CHEKER-DB, the customer can use his own application; to interface with the PC-Ethernet Driver, all that needs to be done is to establish a TCP connection with it. The format of the data that is send over this connection will be well documented and provided by Our company.

```
note that, though the price checker can connect through the PC-Ethernet driver, it does
need different handling by the DB-application!
```

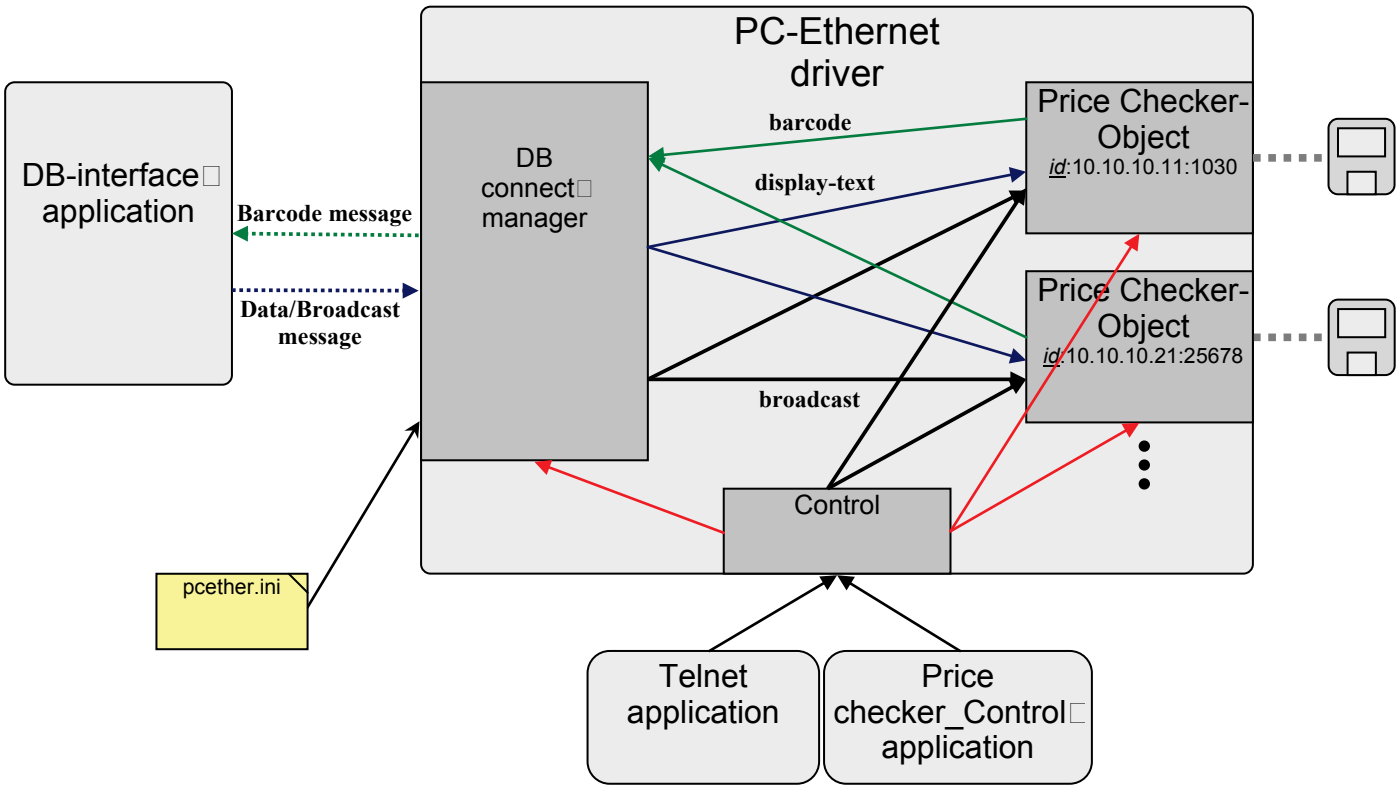
TCP sockets

The different modules communicate through (network-) TCP-sockets. This approach has several advantages:

- **standardization** – All a customer application needs to do to interface with the driver, is connect through a standard TCP connection.
- **platform independence** – TCP connections are the same on any platform.
- **location independence** – TCP connections can be made across a network, so the PRICE CHEKER-control module could be run on a workstation instead of the server (see⋯⋯▸in diagram). For large organizations, even PC-Ethernet and the Database could be run on different machines.
- **security** – Access to the different modules can be protected by a standard firewall.

3 PC-Ethernet-Driver

The PC-Ethernet-Driver contains three interfaces, which are implemented as tcp/ip-server-sockets. This means that the price checker's must be in client mode and that the DB-interface-application must connect to a running PC-Ethernet-driver. Each price checker connects to the same server-socket which creates a separate connection for each price checker. The Database and Control sockets each allow only one connection at a time.



The sending of a barcode and the receiving of the answer are asynchronous; This means that multiple barcodes can be sent to the DB-app before the first answer is received.

3.1 DB-message Format

The communication with the Database-interface application (DB-app) is done with ascii texts over a tcp/ip connection. Default port is **4701**.
These messages are sent using a html-like tag system. The tags allow the ID and Type of a price checker to be included in the message.

An ID is created as a combination of the ip-address and the port-number of the sender. The inclusion of the port-number makes it possible to establish multiple connections from the same ip-address.

A barcode received from the price checker is packaged into a message as follows,
using the "***request***" tag:

```
<request id=10.10.10.11:1030 type=price checker>barcode</request>
```

The DB-app will send the answer back using the same ID, so that the driver will know to which price checker to send it. The Type can be used by the DB-app to format the messages differently for the different price checker types.

An answer from the DB-app looks like this:

```
<data id=10.10.10.11:1030>data_for_barcode</data>
```

The "***data***" tag can be used to send some data to any price checker device using its ID. When the ID option is omitted the message is send to the last price checker that sent a request.

A message can be sent to all connected devices at once by using the "***broadcast***" (or "***bc***") tag:

```
<broadcast>message to all</broadcast>
```

3.2 Configuration File

The configuration file for the PC-ethernet-driver is called “pcether.ini”. The driver looks for it in the directory in which it was started; when it isn't found the hardcoded defaults are used. These will be written to a new pcether.ini file that is created when the driver is closed normally.

[To create a default pcether.ini, start the driver and use the quit command on the control-interface.]

A default pcether.ini is shown in the frame on the right.

```
logpath=
[ControlServer]
port=4700
address=0.0.0.0
reuseport=1
Prompt=Price cheker-telnet>
SocketOpenText=Socket: Control_Server is connected\r\n
[DBServer]
port=4701
address=0.0.0.0
reuseport=1
DbReplyTime=1
DbReceiveSize=1000
TagTimeout=5000
[TagDefinitions]
TagStart=<
TagClose=>
ClosingTagSep=/
OptionSep==
[ConnectManager]
port=9101
address=0.0.0.0
reuseport=1
[Socket]
AliveTimeout=30
DBOfflineText=<ESC>%<ESC>'00Database Offline...
<ESC>'01\nNo price information<ESC>'02\navailable.
```

The items

The configuration options are divided into groups representing different parts of the software; each socket, for instance, has its own group that contains settings for its TCP/IP-port and -address. All groups, except the main-group, are preceded with the name of the group between '[' and ']'.

item	*group*	*description*
address	ControlServer, DBServer, ConnectManager	Sets the IP-address of the network-interface on which to listen. The default of 0.0.0.0 means that it will listen on all network-interfaces.
ClosingTagSep	TagDefinitions	Character that defines a closing-tag. Must be the first character after the tag-start.
DBOfflineText	Socket	When the database is not operational, this text will be send to a device that requests information from it. Any '<ESC>' is expanded to the price checker escape character.
AliveTimeout	Socket	Time in seconds after which the connection to an inactive device is checked; If the device doesn't respond, the connection is closed.
DbReceiveSize	DBServer	Size (in bytes) of the receive buffer.
DbReplyTime	DBServer	Sets the interval-time (in milliseconds) of the receive-loop; check every xx milliseconds for available input.
logpath	-	Sets the file to which status messages are sent; when empty (the default), stdout is used (which is usually the console).
OptionSep	TagDefinitions	Character used to separate the name and value of a tag-option.
port	ControlServer, DBServer, ConnectManager	Defines the TCP-port on which the server listens for connections.

item	*group*	*description*
Prompt	ControlServer	Defines the text used as prompt for the control-socket.
reuseport	ControlServer, DBServer, ConnectManager	Can be 1 or 0; 1 means the port is immediately released when closed, this is the default.
SocketOpenText	ControlServer	Text presented to a user logging in to the control-socket.
TagClose	TagDefinitions	Character that defines the end of a tag.
TagStart	TagDefinitions	Character that defines the start of a tag.
TagTimeout	DBServer	Time (in milliseconds) to wait for a (corresponding) closing tag after receiving a start-tag.

3.3 Control Interface

The control interface is a TCP socket with a simple telnet-server.
The default port for this interface is **4700**.

Only one connection at a time is possible.

When connecting the user is greeted with the text defined by the `SocketOpenText` config-option (see 3.2 -Configuration File). The user is then presented with a (configurable) prompt.

The following commands can be used:

command	*parameters*	*description*
version	-	Version information.
help ?	-	Display a help text with a short description of all commands.
close	-	Ends the connection to the control-interface.
quit	-	Closes the PC-ethernet-driver application.

4 PRICE CHEKER-Demo-DB

The PRICE CHEKER-Demo-DB application serves two purposes: Firstly to be an example and a basic framework for building a database-interface-application for use with the PC-Ethernet-Driver. Secondly as a simple test-application for testing the PC-ethernet-driver.

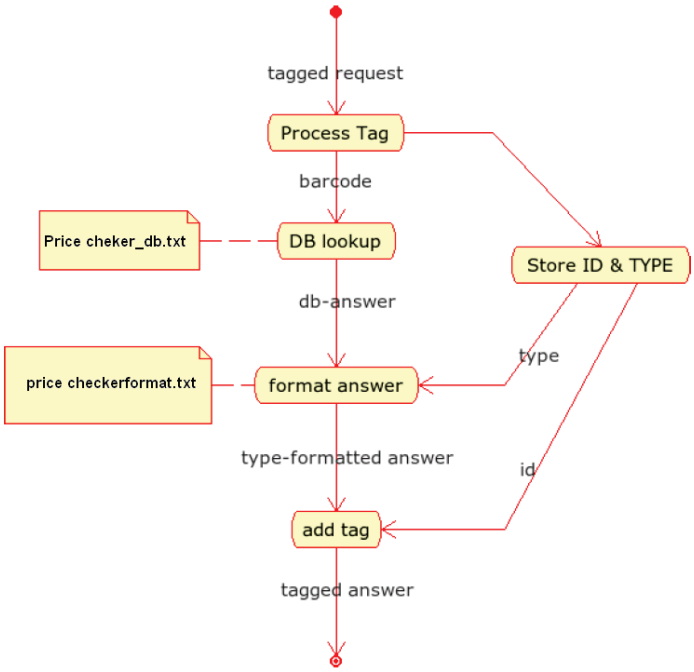
The PRICE CHEKER-Demo-DB was specifically designed for the above mentioned purposes; it is not meant to be used in a “real” environment. Although this would not be impossible for small database sizes, you do so at your own risk!

The database it uses is read from a simple text-file and completely stored in memory. It uses a simple text search, and is therefore easy to build and understand, but not very efficient.

It is written, as much as possible, in standard C++. It was developed and runs on Windows and Linux, but it should not be difficult to port to other platforms. The windows version was compiled and tested with visual studio 6 on windows2000. The Linux version was developed using kdevelop 3.2 and compiled with gcc 3.3 . Both use the same source code; platform specific code is selected with the “WIN32” define-option. Project files for kdevelop and visual studio are included with the source code.

4.1 Overview

The main task of the PRICE CHEKER-Demo-DB is to answer requests from the PC-Ethernet-Driver. The picture to the right shows a simplified overview of how this is done.



After a complete request-tag has been received, the tag is processed; the different parts of the tag are extracted: the ID and TYPE are stored for later use and the barcode is passed on to the database. (see 3.1 -DB-message Format for the layout of the tags)

The barcode is searched for in the list previously loaded from a text file (default `Price cheker_db.txt`). For each barcode multiple strings can be defined, each with a unique name that can be used as an alias in the formatting files.

The answer is then formatted according to the previously stored type-information. The file `price checkerformat.txt` specifies the format for their respective types. For the type “NONE” no formatting is done at all.

This formatted answer is then packaged in a tag containing the ID from the request and send back to the PC-Ethernet-Driver.

4.2 Database File

A database file contains two types of entries: Comments and Items.

Comments are single lines starting with '@#'; these are ignored.

Items consist of multiple lines, the first of which must start with '@>', directly followed by the key for this item (usually a barcode). The following lines each contain one “alias-name” with its string, separated by a ':'. When the alias name is encountered in a format-file, its value is substituted in its place. An item ends when a '@' character is found at the start of the next line.

```
@# TEST DATABASE FILE for the Demo_db app

@>F4011462280035
name:Blue marker pen
price:$0.50

@>F4011462280011
name:Yellow marker pen
price:$0.50

@>unknown
name:Unknown article
```

When a key is not found in the database, the special item “unknown” is returned instead.

4.3 Format Files

The format files (price checkerformat.txt) defines the layout of the screens of their respective devices.

```
<ESC>B0<ESC>$
<ESC>.0{name}<ETX>
{info}
<ESC>B1<ESC>.8<#0x80> {price}<EXT><ESC>B0
```

```
<ESC>%<ESC>'01{name}
<ESC>'03{info}
<ESC>'06        <ESC>=2000<#128> {price}<ESC>=1000
```

When the files are loaded, any occurrence of '<ESC>' is replaced with the price checker escape-character ('\x1b'). This is necessary because the escape character itself, which equals the ascii-escape character, is not readable and can therefore not be used in a normal text file.

When applying the formatting, any text between '{' and '}' is considered the name of an alias, which is then replaced by the corresponding string from the database.

4.4 Startup Parameters

Run the Demo application with the -h option (`Price cheker_demo -h`) to display the following help screen with all the possible parameters:

Note that when the -h or -v options are used, the application will return immediately after displaying the information.

```
Usage: Price cheker_demo [-fhipv]

  -f=FILENAME   : set the filename of the database-text file to use.
                  default: Price cheker_db.txt
  -h            : this helptext.
  -i=IPADDRESS  : the IP-address of the PC_ethernet_driver to connect to.
                  In dotted-decimal notation (e.g. 222.222.222.222)
                  default: 127.0.0.1 (localhost)
  -p=PORTNUMBER : the number of the port of the PC_ethernet_driver.
                  default: 4701
  -v            : display version information.
```

The default address of 127.0.0.1 means that the application will try to connect to a PC-ethernet-driver running on the same machine.

4.5 Commands

When the demo application is started normally, the user is presented with a prompt (`demo>`). Use the help command (“help” or “?”) to get information about the available commands:

```
Command help:
  - ?, help      : This helptext.
  - quit         : Quit the Our company PriceChecker_Database demo.
  - send <file>   : Send the contents of <file> to the PC_Ethernet_Driver.
                   NO formatting is done; the contents of the file is sent
                   as-is, including special characters like LF or CR.
  - loadformat    : (re)Loads the Price checker format files
  - loaddb <name> : Load the database with filename <name>.
                   If no name specified reload the last database.
```

Index