The Relay Cycle Counting

User Reference Manual

This page is left blank intentionally.

DATE	DOCUMENT REVISION	APPLICATION VERSION
11/08/2022	1.0	1.4.0.10
11/08/2022	1.1	1.4.0.20
01/17/2023	1.2	1.4.0.30 - 1.4.0.39
05/25/2023	1.3	1.4.0.40 - 1.4.0.41
08/17/2023	1.4	1.4.0.42
10/16/2023	1.5	1.4.0.48
10/26/2023	1.6	1.4.0.49
11/07/2024	1.7	1.4.0.61

Page Index

1. What is relay cycle counting?

The relay cycle counting monitors physical wearout of the PXI card and its components which are connected inside the PXI or LXI chassis.

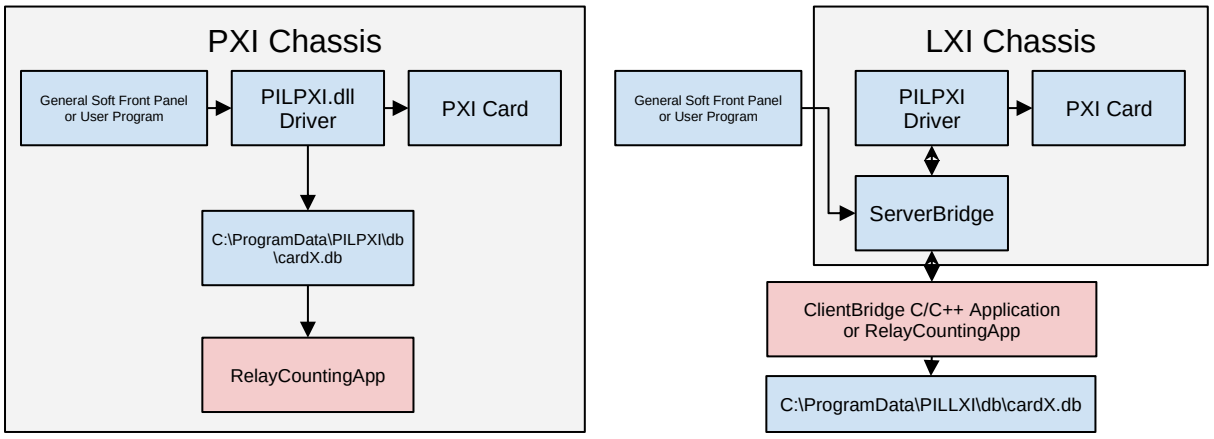
By using relay cycle counting you can quickly determine wearout of the most used electrical components on the PXI card and schedule a service check at your local Pickering sales representative.

The relay cycle counting has only representative meaning and it cannot be used as a warranty exploit.

2. How it works

Relay cycle counting feature is a part of the PXI driver inside your PXI or LXI chassis. The feature monitors the switching activity on the PXI card relays.

The switching data can be viewed by using The RelayCounting Application. Brief description of the relay cycle counting switching flow is represented in the image below. PXI and LXI switching flow slightly differs, but result for obtaining and storing Switching relay's data is same.



Relay cycle counting switching flow.

2.1. End-user Application

- **General Soft Front Panel** - visual application to control relays (without any need for programming).

- **User Program** - any user program that uses Pilpxi.dll, Picmlx.dll or Piplx.dll depending on the application type (PXI or LXI)

2.2. In case of PXI, LXI chassis

PXI chassis: Pilpxi.dll library from PXI Driver provides the following functions:

- Controlling the relays on PCI/PXI Relay Cards (physical hardware)

- Writing relay cycle counting data to “C:\ProgramData**PILPXI**\db\cardX.db”

LXI chassis: Picmlx.dll and Piplx.dll library from C/C++ ClientBridge and .NET driver provides the following functions:

- Creating session to the LXI (Picmlx.dll) and Driving PXI Relay Cards (Piplx.dll)

- Downloads relay cycle counting data via the PXI Relay Cards from the driver.

- Writing relay cycle counting data to “C:\ProgramData**PILLXI**\db\cardX.db”

Relay cycle counting App - GUI application graphically shows relay cycle counting data from database files written by Pilpxi.dll (PXI) or Piplx.dll (LXI).

2.3. Relay cycle counting capable hardware - PXI Cards

Relay cycle counting is currently supported on connected PCI/PXI cards only. PXI cards can be attached to a regular PC using: PCI to PXI Remote Control Kit.

PXI Cards with onboard memory

A PXI card with onboard memory has the ability to store relay cycle counting results since they are first powered on. PXI card stored data are directly obtained from the Pilpxi driver and saved in your local filesystem.

PXI Card without onboard memory

The PXI card without onboard memory can’t store relay cycle counting data on the hardware itself. When you power cycle LXI chassis or you don't have The relay cycle counting feature enabled, data is simply lost.

To prevent data loss as much as possible there is implemented simple file versioning described in section **6.2. File versioning** behavior is the same for PCI/PXI chassis, but data is stored locally on your Host PC.

2.4. Relay cycle counting data setup

To make “Relay cycle counting App” useful you need **relay cycle counting database files (*.db)**. Following requirements have to be met to successfully generate relay cycle counting data database files.

2.4.1. PXI chassis requirements

- Having installed PXI card in PXI chassis.
- Having installed a recent version of Pilpxi.dll (PXI Driver) library.
- Relay cycle counting must be enabled in the Registry (or using the counting settings tab of the Relay cycle counting application).
- Application must use **Pilpxi.dll.**

2.4.2. LXI chassis requirements

- LXI firmware 5.15.0 or higher.If it is not configured to do relay cycle counting , then it can be activated as mentioned in section **4.2. Enabling relay cycle counting in LXI chassis.**
- Having installed a recent version of Picmlx.dll and Piplx.dll (ClientBridge C/C++ driver) library.
- Application must use Picmlx.dll and Piplx.dll (ClientBridge C/C++ driver) library and connection to the LXI chassis must be made in order to work.

3. PXI Relay cycle counting setup

3.1. Installation requirements

PXI relay cycle counting is supported by the PXI Driver v4.33 (PILPXI) library, released on May 23 2018 and higher.

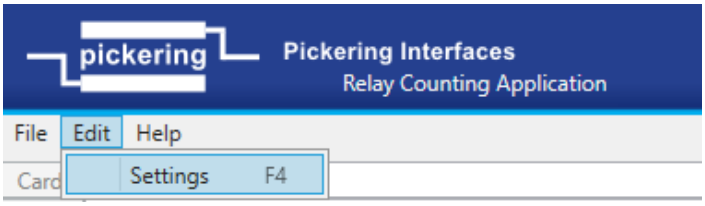
You can download the most recent PXI Driver library from the PXI Software page. Please click on **Direct IO and VISA (32 & 64-Bit)** to Download the most recent PILPXI Library and install onto your system.

3.2. Enabling relay cycle counting

Do the following to enable relay cycle counting in PILPXI driver library:

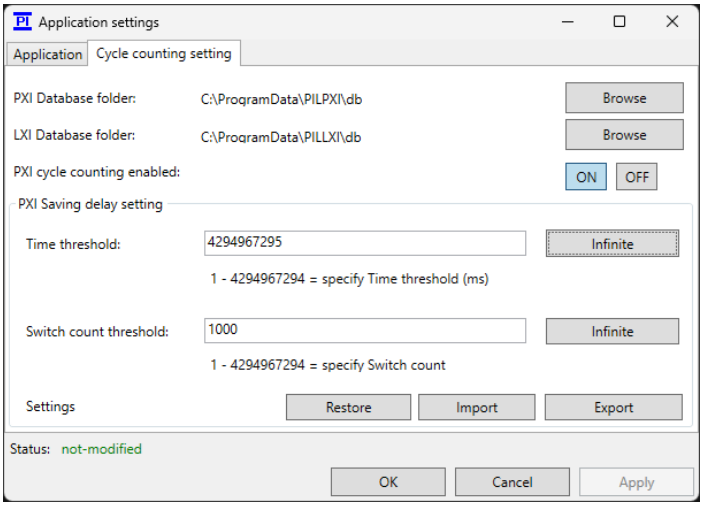
1. Run Relay cycle counting Application.

2. Under Edit -> Settings (F4), open the Application Settings dialog.



The Relay cycle counting Application System Menu.

3. Under Application settings change Tab from Application to **Counting settings**. To change your PXI saving delay setting.



The Relay cycle counting Application Settings dialog.

4. Fit a small non-zero value to Switch count threshold, for example “5” instead of default value “1000” to set Switch count data flush to DB file on your PXI.

5. Click on **Apply** to override default settings and restart your PC.

3.3. Application using relay cycle counting

You can now run any application that is capable of using **Pilpxi.dll** from PXI Driver in **Pickering Direct I/O Driver or IVI VISA** mode.

If you are using **General Soft Front Panel** application and have NI-VISA drivers installed then you should ensure that this application uses really **Pilpxi.dll**

To verify that the General Soft Front Panel uses **Direct I/O mode** do the following:

1. Run General Soft Front Panel

2. Open **Help -> About**

3. Verify that PCI/PXI Connected by: contains the word **Direct I/O or VISA** as in the picture below.

The General Soft Front Panel - About section.

3.4. Preparatory steps

1. Please ensure that you followed the previous chapter Relay cycle counting data setup.

2. You need to have installed the **“Relay cycle counting Application”**

3. Run any application that uses the PXI Driver library and is able to switch relays (to generate relay cycle counting data). For Example:
   - **General Soft Front Panel** (installed with PILPXI Driver library)

- **User program**

4. Generate sample relay cycle counting - for example switching Relay on and off in the General Soft Front Panel several times and for several seconds. Or you can run the User program for some time.

5. Verify that there are generated some relay cycle counting data (*.db). Please check the following directories.

Operating system	**Relay cycle counting data path**
Windows XP SP3	C:\Document and Settings\All Users\Application Data\PILPXI\db\
Windows 7 or later	C:\ProgramData\PILPXI\db\

Above directories are set System and Hidden by default. Therefore they can be invisible to File Explorer and other regular Windows applications. However you can always access them by specifying their path manually according to the above table.

If relay cycle counting seems to not work, please check the registry path in your system using The Registry Editor - **"regedit.exe"** and look for the following values.

Registry path for 32-bit Application		
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432NODE\Pickering Interfaces Ltd\Pilpxi		
Binary Value	TimeTimeout	range: 0 .. 4294967294
Binary Value	CountTimeout	range: 0 .. 4294967294
Binary Value	CountingEnabled	range: 0 .. 1

Registry path for 64-bit Application		
HKEY_LOCAL_MACHINE\SOFTWARE\Pickering Interfaces Ltd\Pilpxi		
Binary Value	TimeTimeout	range: 0 .. 4294967294
Binary Value	CountTimeout	range: 0 .. 4294967294

Binary Value	CountingEnabled	range: 0 .. 1

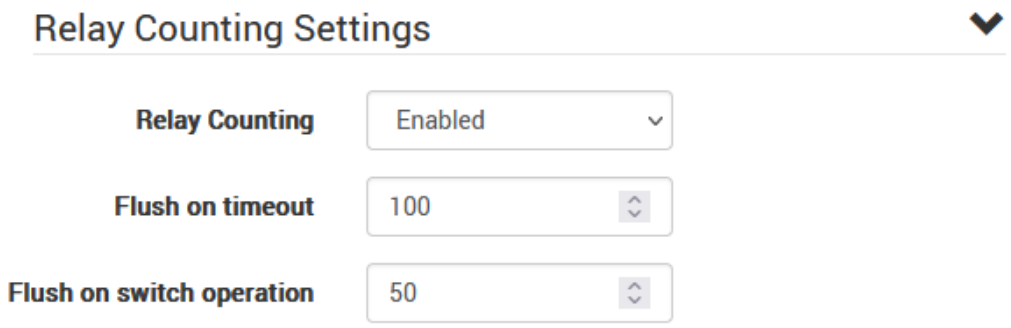
4. LXI relay cycle counting setup

4.1. Installation requirements

LXI relay cycle counting is supported by the C/C++ ClientBridge Driver and .NET Driver, released on April 5, 2022 and newer.

4.2. Enabling relay cycle counting in LXI chassis

Currently, relay cycle counting on LXI is supported by LXI firmwares (from version 5.15.0) as a manual toggle switch under LXI Web management under Diagnostics. The chassis must be rebooted for changes to take effect.



Relay cycle counting toggle from LXI web management - Diagnostics

4.3. Preparatory steps

1. Please ensure that you followed the previous chapter **Relay cycle counting data setup**.

2. Having **LXI firmware** that supports Relay cycle counting from **version 5.15.0** in your LXI chassis, for more information please contact *support@pickeringtest.com*.

3. Having to manually toggle the relay cycle counting switch on LXI chassis Web management pages. Navigate under **Diagnostics -> Toggle Enabled/Disabled Combo box** in section **" Relay cycle counting Settings"**.

4.4. Application using relay cycle counting

You can run any application that is capable of using **Piplx.dll** from ClientBridge C/C++. Relay cycle counting data are obtained on the C function call **PIPLX_CountFreeCards();** If inner behavior of the function proceeds correctly, data from LXI chassis will be saved at **"C:\ProgramData\PILLXI\db\cardX.db"**

1. Run a user program which uses Piplx.dll and performs any switching operation with physical cards.

2. User program must call C function: **PIPLX_CountFreeCards()**;

3. Verify that there are generated some relay cycle counting data (*.db). Please check the following directories.

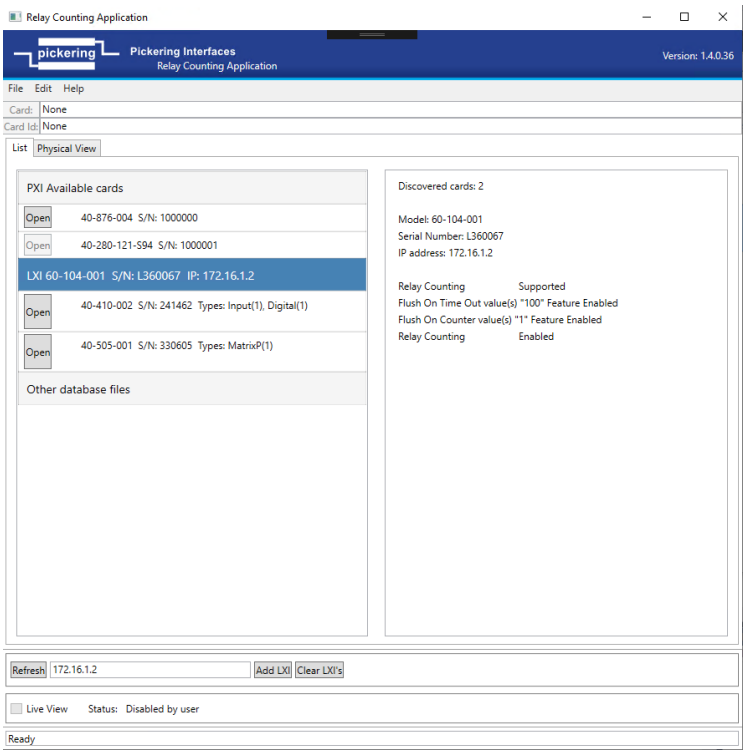
Operating system	Relay cycle counting data path
Windows XP SP3	C:\Document and Settings\All Users\Application Data\PILLXI\db\
Windows 7 or later	C:\ProgramData\PILLXI\db\

Above directories are set **System and Hidden** by default. Therefore they can be invisible to File Explorer and other regular Windows applications. However you can always access them by specifying their path manually according to the above table.

There should exist at least one *.db file (for example 40-523-022,32004,1.00.db) For more information on how to read this file, please follow section - **6.2. File Format specification**.

5. Relay cycle counting Application - User manual

Relay cycle counting application supports opening relay cycle counting database files (*.db), obtaining data from PXI and LXI chassis.



The Relay cycle counting Application - initial startup view.

5.1. User interface description

5.1.1. Application Menu Bar

- File
  - Open (Ctrl+O)
  - Exit (Alt+X)
- Edit
  - Settings (F4)

- Help
  - Contents (F1)
  - About (Ctrl+Alt+F1)

5.1.2. List

The device list allows the user to see available PCI/PXI cards in case of PXI chassis or add a remote device like LXI via typing IP address into Edit Line field and pressing **ADD LXI** button.

In the right side of the List view panel you can see basic information about PXI Cards and relaycount status. By pressing the Open button at the specific PXI card you are able to view Relay cycle counting Data more precisely in Physical View form.

5.1.3. Physical View

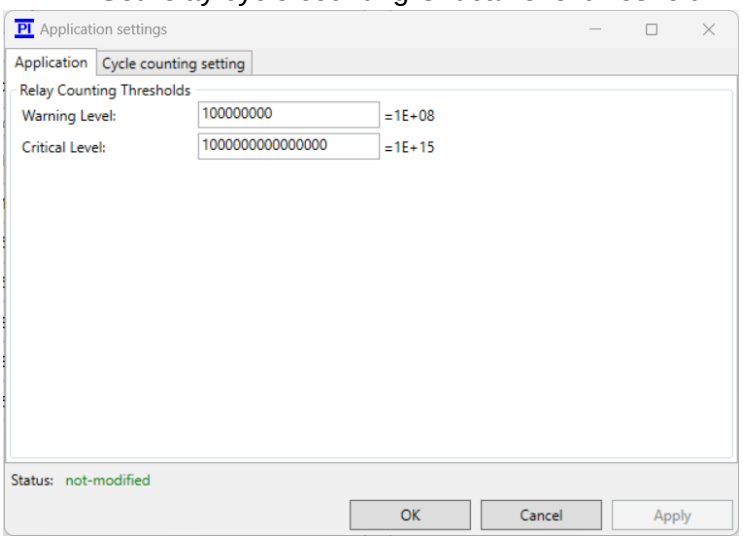
The Physical View allows the user to see a list or graphical representation and switch count of relay cycle counting data for each switching relay on the PXI card.

5.2. Application settings

5.2.1. Application Tab

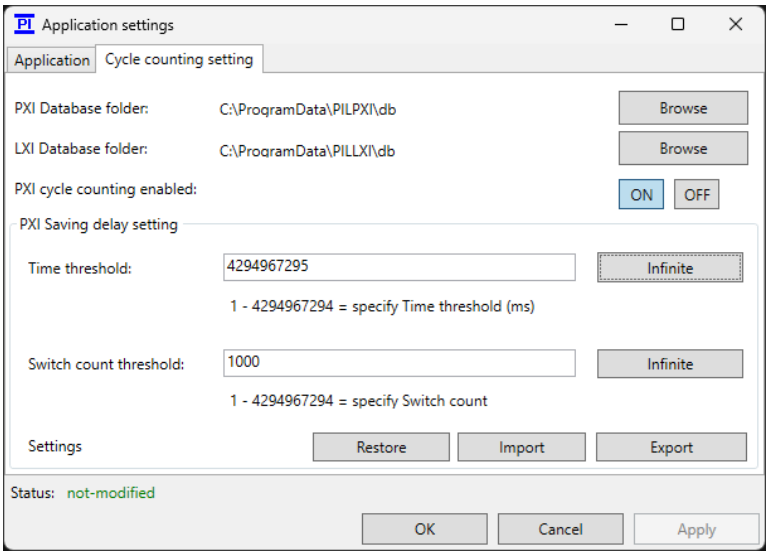
This Application settings enables user to do a specific actions like:

- Set relay cycle counting Warning level threshold
- Set relay cycle counting Critical level threshold



Application Settings - Counting settings TAB

5.2.2. Cycle counting settings Tab



Application Settings - Cycle counting settings TAB

The counting settings enables user to do a specific actions like:

- Browsing PXI and LXI default location of the database folders.
- Enable/Disable Relay cycle counting
- Set PXI Saving delay setting
    - **Time threshold** - sets time between each write to the database (in milliseconds)
    - **Switch count threshold** - sets number of operations after which the count is written into the database.
- Import, Export or Restore PXI relay cycle counting settings.

5.3. Loading relay cycle counting data files (*.db)

Application supports loading and viewing relay cycle counting database files (*.db) which are created and transferred from different PCs such as Windows/Linux Workstations.

This can be simply made by clicking on the Application Menu Bar, choosing **File -> Open** Action. After popping up the File Explorer Dialog, navigate to your relay cycle counting database file backups to view them.

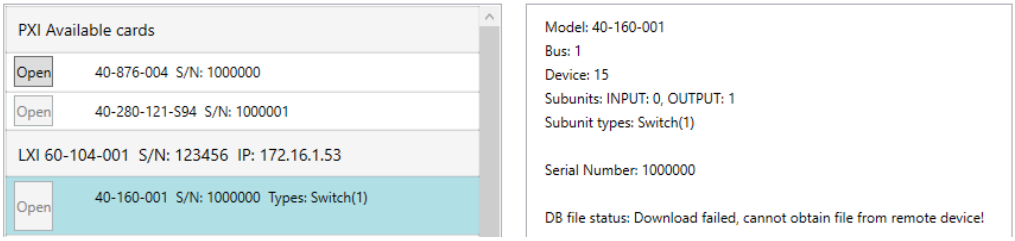
Any user opened relay cycle counting data file will be present under **“Other database files”** - section in device list.

5.4. Obtaining card relay cycle counting data from PXI

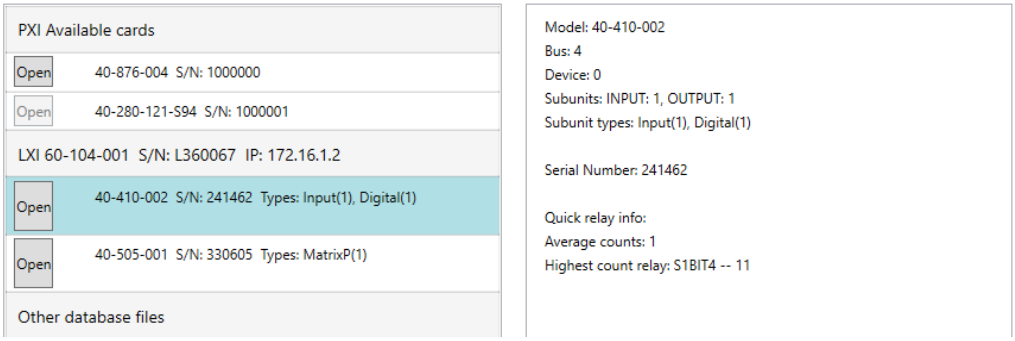
Relay cycle counting data are obtained from PCI/PXI cards locally via PILPXI driver. Make sure your PCI/PXI Card is properly connected to the Host PC or PXI chassis with fast link connection (Thunderbolt or PCIe to PXIe Remote control kit, including module and cable). After any switching operation on the PXI card, relay cycle counting data should be accessible from the Application, under the **OPEN** button.

5.5. Obtaining card relay cycle counting data from LXI

Relay cycle counting data are obtained from the LXI chassis only when **LXI is added to the device list**. When relay cycle counting data are obtained from LXI successfully there will be an active **OPEN** button for each supported card, if not the button will be grayed out.



The Card does not have relay cycle counting data available.



The Card does have relay cycle counting data available.

The **REFRESH** button refreshes and provides a new set of relaycounting data for each LXI added in the list.

Refresh operation will reconnect each LXI chassis in the device list and download new data from them. This might take some time, depending on your LAN network speed.

Network connection is not permanent - “**Live view”** is not available for the LXI.

5.6. Relay cycle counting Data Storage

5.6.1. PXI chassis (PCI/PXI Cards)

PXI card relay cycle counting data are stored on your Host PC directly.

The default storage location for relay cycle counting data files under Microsoft Windows is **“C:\ProgramData\PILPXI\db\”.**

5.6.2. LXI chassis

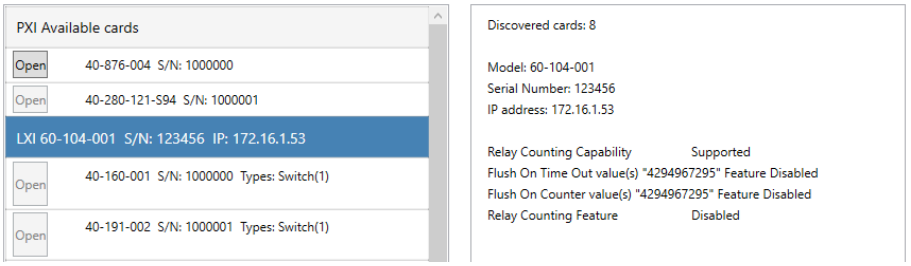
PXI card relay cycle counting data are pulled from LXI via a network connection. For proper function, LXI must be added to the device list via IP address. After adding the LXI chassis to the device list, a one-time session to the LXI is made, and data are pulled directly from the LXI to the PC host machine.

The default storage location for relay cycle counting data files under Microsoft Windows is **“C:\ProgramData\PILLXI\db\”.**

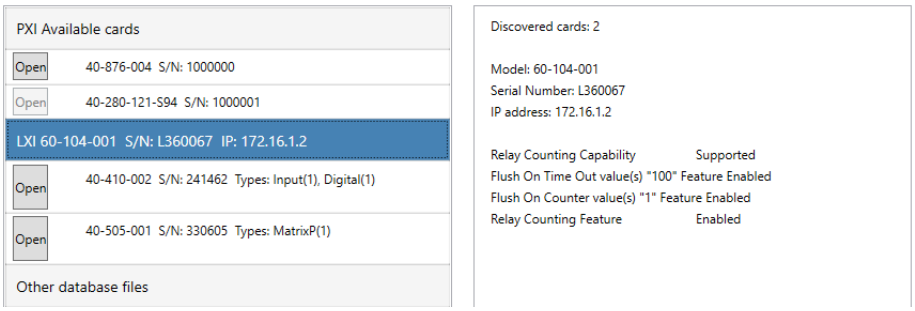
5.7. Troubleshooting

5.7.1. Checking relay cycle counting support on the LXI

When PILPXI driver inside LXI does not have relay cycle counting enabled, the application will notify User after selecting the LXI device from the device list as shown in the image below.



The LXI without relay cycle counting support.



The LXI with relay cycle counting support.

6. Relay cycle counting Database File Reference

6.1. Overview

Relay cycle counting database files are written by PXI driver. The shortest interval is defined by the FlushTimeout registry entry (in seconds). You can set this value comfortably with the counting settings tab of the relay cycle counting application.

6.2. File versioning

The relay cycle counting database file has implemented a simple version checking mechanism. The database file is compared and updated between LXI and Host PC vice versa, following the rule:

- When the file version inside LXI is greater than Host PC, the *.db file will be downloaded from LXI chassis to Host PC.

- When the file version inside LXI is lower than Host PC, a *.db file will be uploaded to the LXI from Host PC.

- When files are the same, the LXI db file always has higher priority than the Host PC and will be kept.

This behavior is also implemented inside the ClientBridge C/C++ Library.

6.3. File format specification

Fileformat holds ASCII text-based data with the following structure.

Delimiter tokens

; - is reserved for next characters
\n - new line
, - single value string delimiter

File header - must be present with correct structure.

Example:
PILPXIDB004;60-891-006,410155,1.0

PILPXIDB004 - Magic header

60-891-006,410155,1.0 - Card info (single value string)

60-891-006 - Card Name
410155 - Card serial number
1.0 - Card firmware version

First Token character

G - File generation
A - Firmware defined architecture
L - Logical, number of sub-units
H - Record type header, comment
S - Subunit
R - Relay
E - End of file

End of file

- **E;EOF**

Rules

- Logical and Physical component information must remain the same.

Decoding line examples

A;3;Loops;1;32

A - Architecture
3 - number of Loops (or other type)
Loops - (Internal description)
1;32 - means there are total of 32 *(1 * 32)* components present..
1 - number of component along rows (X)
32 - number of component along columns (Y)

L;1 - 1 logical layer

S;0;1;32;32;1;SWITCH(32)

S - subunit
0 - first layer
1 - number of components along X
32 - number of components along Y
32 - CRC for components X*Y
1 - subunit type identifier (see enums for PIRC_SubInfo())
SWITCH(32) - string type identifier

R;L;S1BIT9;1

R - relay
L - logical layer
S1BIT9 - 1st sub-unit, 9th bit
1 - number of switching operations

R;P;L1BIT9;1

R - relay
P - physical layer
L1BIT9 - 1st loop, 9th bit (physical component)
1 - number of switching operations

6.4. Backup and Restore

Relay cycle counting database files are the only persistent data containing information about relay's usage.

If these files are lost (for example because operating system reinstall or hard disk replacement) then counters will start again from zero - such data will be skewed without providing reliable information about relay usage.

Do the following to backup relay cycle counting database data:

1. Locate directory with database files. You have few options:

    a. Copy and Paste **Database Folder**: value from Counting settings tab of the relay cycle counting Application.

    b. Use table below:

In case of PXI chassis	
Operating system	**Relay cycle counting data path**
Windows XP SP3	C:\Document and Settings\All Users\Application Data\PILPXI\db\
Windows 7 or later	C:\ProgramData\PILPXI\db\

In case of LXI chassis	
Operating system	**Relay cycle counting data path**
Windows XP SP3	C:\Document and Settings\All Users\Application Data\PILLXI\db\
Windows 7 or later	C:\ProgramData\PILLXI\db\

Above directories are set System and Hidden by default. Therefore they can be invisible to File Explorer and other regular Windows applications. However you can always access them by specifying their path manually according to the above table.

    c. If you are a programmer you can use this snippet of code to find the parent database directory.

C++ Language

```
#include <windows.h>
#include <ShlObj.h>

// …

CHAR szPath[MAX_PATH];
char buf[MAX_PATH];

// …

// error handling omitted for brevity
SHGetFolderPath(NULL, CSIDL_COMMON_APPDATA, NULL, 0,
szPath);

// append PILPXI\db to get full path
sprintf_s(buf, MAX_PATH, “%s\\PILPXI\\db”, szPath);

// or

sprintf_s(buf, MAX_PATH, “%s\\PILLXI\\db”, szPath);
```

2. Now backup all *.db files to your backup device using common file copy commands (including Windows COPY or XCOPY commands)

To restore Backed up *.db files do the following:

Batch Script

```
:: create directory for *.db files - please check above table for correct
path

md ‘c:\ProgramData\PILPXI\db\’
:: or
md ‘c:\ProgramData\PILLXI\db\’

xcopy \my_backup_path\*.db ‘c:\ProgramData\PILPXI\db\’
:: or
xcopy \my_backup_path\*.db ‘c:\ProgramData\PILLXI\db\’
```

7. Relay cycle counting C API Library

7.1. Overview

RelayCounting C API library gives a programmer access to reading stored database files on your local machine via simplified API interface. API is distributed inside a installer of RelayCounting Front Panel Application.

7.2. Public API Enumeration Constants

7.2.1. Enum definition for function return error codes

RC_NO_ERR - No error

RC_ER_NO_CARD - No card present with specified number
RC_ER_NO_INFO - Card information unobtainable - (file problem)
RC_ER_BAD_SUB - Card has no sub-unit with specified number
RC_ER_BAD_BIT - Sub-unit has no bit with specified number
RC_ER_INVALID_ARG_POINTER - Invalid user-argument function pointer
RC_ER_INVALID_POINTER - Internal error pointer inside library
RC_ER_FILESYSTEM - Filesystem error
RC_ER_INTERNAL_PTR_FREE -Internal Free pointer error
RC_ER_UNKNOWN_LIMIT_TYPE - Unknown limit type
RC_ER_NUM - Number of error codes

7.2.2. Enum definition for Sub-unit type specifier codes

returned by PIRC_SubInfo() function

TYPE_PHYSICAL - Physical card structure
TYPE_SW - Uncommitted switches
TYPE_MUX - Relay multiplexer (single-channel only)
TYPE_MUXM - Relay multiplexer (multi-channel capable)
TYPE_MAT - Standard matrix
TYPE_MATR - RF matrix
TYPE_DIG - Digital outputs
TYPE_RES - Programmable Resistor
TYPE_ATTEN - Programmable Attenuator
TYPE_PSUDC - Power supply - DC
TYPE_BATT - Battery simulator
TYPE_VSOURCE - Programmable voltage source
TYPE_MATP - Matrix with restricted operating modes
TYPE_MUXMS - Relay multiplexer (MUXM hardware emulated as MUX)

TYPE_FI - Sub-unit with restricted operation based on other Sub-unit operation
TYPE_DM - Displacement Module simulator
TYPE_PSOURCE - Power Source module
TYPE_DIO - DIO Subunit
TYPE_MATS - Matrix with self aligning capability
TYPE_DAC - DAC Subunit
TYPE_COMP - Comparator / Event Detector

7.2.3. Enum definition for relay count counting limits as LimitType

for PIRC_SetSwitchLimit(), PIRC_GetSwitchLimit() functions

COUNT_WARNING - Warning limit type
COUNT_CRITICAL - Critical limit type
COUNT_LIMIT_NUM - Number of limit types

7.3. Public API Functions

7.3.1. PIRC_Version

Description:

Get PIRC driver version.

Returns:

Variable Function Type DWORD

Version number of PIRC software driver, for example return value = 100 indicates version 1.00

7.3.2. PIRC_CardDataAvailable

Description:

Obtain a list of database files of Pickering cards from the local system.

Parameters:

CardListFnStr - pointer to 2D CHAR array which will allocate Card List string

StrLen - returns length of MAX_BUF=255 string inside the Card List string

ListLen - returns number of filenames inside the Card List string

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.3. PIRC_CardDataFree

Description:

Clear all data allocated by 'PIRC_CardDataAvailable' function

Parameters:

CardListFnStr - pointer to Card String List
ListLen - Length of the Card String List

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.4. PIRC_OpenSpecifiedCardData

Description:

Open card data via card filename string.

Parameters:

CardListFnStr - card filename string identifier of card to open

ListLen - pointer to variable to receive the Pickering card number of all available card units

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

Example:

CardFnStr = "40-136-011,272181,1.00.db"... points to On Windows "C:\ProgramData\PILPXI\db\40-136-011,272181,1.00.db"

7.3.5. PIRC_CloseSpecifiedCardData

Description:

Close card data

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.6. PIRC_EnumerateSub

Description:

Get Pickering card sub-unit counts.

Parameters:

CardNum - Pickering card reference of target

SubNum - pointer to variable to receive subunit count

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.7. PIRC_CardDataId

Description:

Get Pickering card data identification string

Parameters:

CardNum - Pickering card reference of target

Str - pointer to character string to receive result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

*Parameter Variable CHAR *Str*

The result contains “<type code>,<serial number>,<revision number>”

7.3.8. PIRC_CardDataRevision

Description:

Obtain card count file iteration.

Parameters:

CardNum - Pickering card reference of target

Count - pointer to variable to accept result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.9. PIRC_SubInfo

Description:

Get Pickering card data sub-unit information (numeric format).

Parameters:

CardNum - sub-unit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

TypeNum - pointer to variable to receive type code result. Using subunit type codes specifier.

Rows - pointer to variable to receive row dimension result

Cols - pointer to variable to receive column dimension result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.10. PIRC_SubType

Description:

Get Pickering card data sub-unit information (in string format).

Parameters:

CardNum - Pickering card reference of target

SubNum - subunit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

Str - pointer to character string to receive result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.11. PIRC_GetCrosspointOps

Description:

Get the number of a matrix crosspoint, count operation.

Parameters:

CardNum - Pickering card reference of target

SubNum - subunit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

Row - crosspoint row (Y) location

Column - crosspoint column (X) location

Count - pointer to variable to accept result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.12. PIRC_GetSwitchOps

Description:

Get the number of a switch, count operation.

Parameters:

CardNum - Pickering card reference of target

SubNum - subunit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

BitNum - output bit number

Count - pointer to variable to accept result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.13. PIRC_GetMaxOps

Description:

Get the number of maximum count operations for subunit.

Parameters:

CardNum - Pickering card reference of target

SubNum - subunit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

Count - pointer to variable to accept result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.14. PIRC_GetMeanOps

Description:

Get the mean of a sample count operation for a subunit.

Parameters:

CardNum - Pickering card reference of target

SubNum - subunit of target to access. Zero value will return physical unified subunit view, any value higher than zero will return a logical view of the subunit.

Count - pointer to variable to accept result

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.15. PIRC_SetSwitchLimit

Description:

Set relay cycle counting switch limit.

Parameters:

LimitType - set limit type

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.16. PIRC_GetSwitchLimit

Description:

Get relay cycle counting switch limit.

Parameters:

LimitType - set limit type

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.3.17. PIRC_GetErrorMessage

Description:

Get a string description of an error code

Parameters:

ErrorCode - the error code to be described

Returns:

Variable Function Type DWORD

Zero - for success
Non-zero - for error code

7.4. Example Programs

7.5. Foundation

C Language

```
#include <stdio.h>   // Standard Input/Output C Library functions
#include <pirc.h>    // RelayCounting C Api Header File

// Main C entry function
int main(int argc, char **argv) {

   PCHAR* CardList;   // Pointer to card string list
   DWORD ListLen = 0; // Length of card list
   DWORD StrLen = 0;  // Max length of card string inside list
   DWORD result = 0;  // Function result return value


   ///////////////////////
   // Initialization sequence
   ///////////////////////

   // Obtain available card names from system path directory
   if(!(result=PIRC_CardDataAvailable(&CardList,&StrLen,&ListLen)))
   {
      // Go through ListLen and print content of the CardList using
      // CardIdStr as increment!
      for (int CardIdStr=0; CardIdStr < ListLen; CardIdStr++)
      {
```

```
         printf("Card id: %d\tfilename: '%s'\n", CardIdStr,
               CardList[CardIdStr]);
      }

      // <INSERT OTHER SAMPLE CODE SNIPPETS HERE>
      // . . .
      // <INSERT OTHER SAMPLE CODE SNIPPETS HERE>
   }

   ////////////////////
   // Shutdown sequence
   ////////////////////

   // Release all allocated data from PIRC_CardDataAvailable func
   if((result=PIRC_CardDataFree(&CardList,ListLen)))
   {
      // if PIRC_CardDataFree has failed, print error code and
      // message.
      printf("Error result=%d, msg=%s\n", result,
              PIRC_GetErrorMessage(result));
   }

   return 0;
}
```

Other examples with code can be found at **(RelayCycleCountApp Folder)\\PircSDK\\Examples**.

Examples	
DatabaseList	Example shows how to list database files present in the system through PiRC SDK - C API.
DBFileLoader	Example shows direct print of the database file structure through C API.
GetCrossPointOps	Example shows how to obtain cross point operations from PiRC SDK - C API.
GetSwitchOps	Example shows how to obtain switch operations fromPiRC SDK - C API.
SetGetLimits	Example shows how to set/get count limits from PiRC SDK - C API.