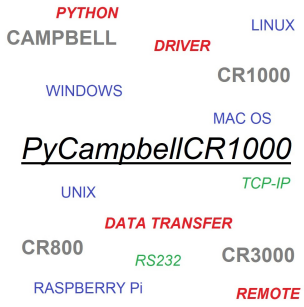

PyCampbellCR1000

Release 0.4dev

Salem Harrache & Lionel Darras

Jan 08, 2018

1 About

This project was conducted by the laboratory ISTerre Grenoble (FR) as part of the development of an observation network OMIV (Observatoire Multi-disciplinaires des Instabilités de Versants)

2 Description

PyCampbellCR1000 is a python project which aims to allow the communication with Campbell CR1000 Type Datalogger

The main features include automatic collecting of data and settings (read only) as a list of dictionnaries.

The tool can be used in your python scripts for data post-processing, or in command line mode to collect data as CSV.

We don't update anything from PyCampbellCR1000 besides time, because we are assuming that the dataloggers are already configured.

Note: PyCampbellCR1000 uses the PyLink lib, offers a universal communication interface with File-Like API.

2.1 Examples

We init communication by giving the datalogger URL.

```
>>> from pycampbellcr1000 import CR1000
>>> device = CR1000.from_url('tcp:host-ip:port')
>>> # or with Serial connection
>>> device = CR1000.from_url('serial:/dev/ttyUSB0:38400')
```

To get time, use:

```
>>> device.gettime()
datetime.datetime(2012, 7, 16, 12, 27, 55)
```

To get data, you have to enter the table name where it is stored. If you don't know the table name, you cannot collect the list of available tables in the datalogger.

```
>>> device.list_tables()
['Status', 'Table1', 'Public']
```

Choose the time period to get your data from *start date* to *stop date*.

```
>>> import datetime
>>> start = datetime.datetime(2012, 7, 16, 11, 0, 0)
>>> stop = datetime.datetime(2012, 7, 16, 12, 0, 0)
>>> data = device.get_data('Table1', start, stop)
>>> data[0]["Datetime"]
datetime.datetime(2012, 7, 16, 11, 0)
```

```
>>> data[0]["CurSensor1_mVolt_Avg"]
2508.0
```

```
>>> print(data.filter(('Datetime', 'CurSensor3_mAmp_Avg')).to_csv())
Datetime,CurSensor3_mAmp_Avg
2012-07-16 11:00:00,18.7
2012-07-16 11:01:00,18.48
...
2012-07-16 11:59:00,17.25
```

2.2 Features

- Collecting data as a list of dictionaries
- Collecting data in a CSV file
- Reading and adjusting the data logger's internal clock
- Retrieving table definitions
- Listing table names
- Reading settings
- Collect file list and download file content
- Tested with CR1000 and CR800 dataloggers (should work with CR3000 datalogger)
- Various types of connections are supported (TCP, UDP, Serial, GSM)
- Comes with a command-line script
- Compatible with Python 2.6+ and 3.x

2.3 Installation

You can install, upgrade, uninstall PyCampbellCR1000 with these commands

```
$ pip install pycampbellcr1000
$ pip install --upgrade pycampbellcr1000
$ pip uninstall pycampbellcr1000
```

Or if you don't have pip

```
$ easy_install pycampbellcr1000
```

Or you can get the source code from github.

```
$ git clone https://github.com/LionelDarras/PyCampbellCR1000.git
$ cd PyCampbellCR1000
$ python setup.py install
```

2.4 About CR1000 Type Datalogger

This can be read in the BMP5 Transparent Commands Manual Rev 9/08 :

Note: The CR1000 type datalogger (CR1000, CR3000, and CR800) is a rugged and versatile measurement device. This datalogger contains a CPU and both digital and analog inputs and outputs. The CR1000 type datalogger uses a PakBus operating system and communicates with applications via the BMP5 message protocol. Programs sent to the datalogger are written in a BASIC-like language that includes data processing and analysis routines. These programs run on a precise execution interval and will store measurements and data in tables.

PyCampbellCR1000 implement part of Pakbus protocol and can thus communicate with this type of datalogger.

We only tested it with CR1000 and CR800 dataloggers. If you have a CR3000 datalogger, feel free to test the tool and inform us about the compatibility with your machine.

3 Command-line usage

PyCampbellCR1000 has a command-line script that interacts with the datalogger.

```
$ pycr1000 -h
usage: pycr1000 [-h] [--version] {gettime,settime,getprogstat,getsettings,
                                  listfiles, getfile,listtables,getdata,
                                  update} ...

Communication tools for Campbell CR1000-type Datalogger

optional arguments:
  -h, --help            Show this help message and exit
  --version             Print PyCR1000's version number and exit.

The PyCR1000 commands:
    gettime             Print the current datetime of the datalogger.
    settime             Set the given datetime argument on the datalogger.
    getprogstat         Retrieve available programming statistics information
                        from the datalogger.
    getsettings         Retrieve the datalogger settings.
    listfiles           List all files stored in the datalogger.
    getfile             Get the file content from the datalogger.
```

```
    listtables          List all tables stored in the datalogger.
    getdata             Extract data from the datalogger between start
                        datetime and stop datetime.By default the entire
                        contents of the data will be downloaded.
    update              Update CSV database records with getting automatically
                        new records.
```

3.1 Gettime

The *gettime* command gives, as its name suggests, the current datetime of the datalogger.

```
$ pycr1000 gettime -h
usage: pycr1000 gettime [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                        [--code CODE] [--debug]
                        url

Print the current datetime of the datalogger.

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1 or serial:/COM1:19200:8N1

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
```

Note

address ID is the pakbus address of the first datalogger connected to the communication port. node ID is the pakbus address of the client datalogger in a network of several dataloggers;

Example

```
$ pycr1000 gettime serial:COM1:38400
2012-07-16 21:57:30
```

3.2 Settime

Allows us to update the datalogger datetime and returns the new value.

```
$ pycr1000 settime -h
usage: pycr1000 settime [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                        [--code CODE] [--debug]
                        url datetime

positional arguments:
url                 Specify URL for connection link. E.g. tcp:iphost:port or
                    serial:/dev/ttyUSB0:115200:8N1 or serial:/COM1:115200:8N1
datetime            The chosen datetime value. (like : "2012-07-16 21:58:23" or "2012-
↪07-16 21:58")
```

```
optional arguments:
-h, --help          Show this help message and exit
--timeout TIMEOUT   Connection link timeout (default: 10.0)
--src_addr SRC_ADDR       Source address ID (default: None)
--src SRC_NODE            Source node ID (default: 2050)
--dest_addr DEST_ADDR     Destination address ID (default: None)
--dest DEST_NODE          Destination node ID (default: 1)
--code CODE         Datalogger security code (default: 0)
--debug             Display log (default: False)
```

Example

```
$ pycr1000 settime serial:/dev/ttyUSB0:19200:8N1 "2012-07-16 23:00:00"
Old Time : 2012-07-16 22:00:12
Current Time : 2012-07-16 23:00:00
```

3.3 Getprogstat

Retrieve available programming statistics from the datalogger.

```
$ pycr1000 getprogstat -h
usage: pycr1000 getprogstat [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                            [--code CODE] [--debug]
                            url

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
```

Example

```
$ pycr1000 getprogstat tcp:localhost:1112
CompResult : CPU:CR1000_LABO.CR1 -- Compiled in PipelineMode.
PowUpProg : CPU:CR1000_LABO.CR1
OSSig : 12288
ProgName : CPU:CR1000_LABO.CR1
CompState : 1
ProgSig : 2993
OSVer : CR1000.Std.24
CompTime : 2012-07-13 11:49:02
SerialNbr : E4668
```

3.4 Getsettings

Retrieve the datalogger settings as CSV.

```
$ pycr1000 getsettings -h
usage: pycr1000 getsettings [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                            [--code CODE] [--debug] [--output OUTPUT]
                            [--delim DELIM]
                            url

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
  --output OUTPUT     Filename where output is written (default: <stdout>)
  --delim DELIM       CSV char delimiter (default: ',')
```

Example

```
$ pycr1000 getsettings tcp:127.0.0.1:1112
SettingId,SettingValue,ReadOnly,LargeValue
0,'CR1000.Std.24\x00',1,0
1,'E\x00\x12<',1,0
...
```

3.5 Listfiles

Lists all files stored in the datalogger.

```
$ pycr1000 listfiles -h
usage: pycr1000 listfiles [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                          [--code CODE] [--debug]
                          url

Lists all files stored in the datalogger.

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
```

```
  --code CODE           Datalogger security code (default: 0)
  --debug               Display log (default: False)
```

Example

```
$ pycr1000 listfiles tcp:localhost:1112
CPU:
CPU:templateexample.cr1
CPU:CR1000_LABO.CR1
```

3.6 Getfile

Get the file content from the datalogger.

```
$ pycr1000 getfile -h
usage: pycr1000 getfile [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                        [--code CODE] [--debug]
                        url filename output

positional arguments:
  url                   Specify URL for connection link. E.g. tcp:iphost:port or
                        serial:/dev/ttyUSB0:19200:8N1
  filename              Filename to be downloaded.
  output                Filename where output is written

optional arguments:
  -h, --help            Show this help message and exit
  --timeout TIMEOUT     Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE           Datalogger security code (default: 0)
  --debug               Display log (default: False)
```

Example

```
$ pycr1000 getfile tcp:localhost:1112 "CPU:templateexample.cr1" ./templateexample.cr1
$ head templateexample.cr1
'CR1000/800/850 Series Datalogger
'This Program is the Same as the default template in the
'CRBasic Editor.

'Declare Public Variables
'Example:
Public PTemp, batt_volt

'Declare Other Variables
'Example:
```

3.7 Listtables

Lists all table names stored in the datalogger.

```
$ pycr1000 listtables -h
usage: pycr1000 listtables [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                           [--code CODE] [--debug]
                           url


positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
```

Example

```
$ pycr1000 listtables tcp:localhost:1112
Status
Table1
Public
```

3.8 Getdata

Extract data from the datalogger between start datetime and stop datetime. By default the entire contents will be downloaded.

```
$ pycr1000 getdata -h
usage: pycr1000 getdata [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                        [--code CODE] [--debug] [--start START] [--stop STOP]
                        [--delim DELIM]
                        url table output

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1
  table               The table name used for data collection
  output              Filename where output is written

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
  --start START       The beginning datetime record (like : "2012-07-17 11:25")
                      (default: None)
  --stop STOP         The stopping datetime record (like : "2012-07-17 11:25")
```

```
                      (default: None)
  --delim DELIM       CSV char delimiter (default: ,)
```

Example

```
$ pycr1000 getdata tcp:localhost:1112 'Table1' \
--start "2012-07-17 10:25" --stop "2012-07-17 11:25" ./my_data.csv
Your download is starting.
Packet 0 with 47 records
Packet 1 with 13 records
--------------------------
60 new records were found
```

```
$ head my_data.csv
Datetime,RecNbr,CurSensor1_mVolt_Avg,CurSensor3_mVolt_Avg, CurSensor3_mAmp_Avg
2012-07-17 10:25:00,75717,2509.0,2508.0,15.97
2012-07-17 10:26:00,75718,2509.0,2508.0,15.69
2012-07-17 10:27:00,75719,2509.0,2508.0,17.76
2012-07-17 10:28:00,75720,2509.0,2508.0,16.75
2012-07-17 10:29:00,75721,2509.0,2508.0,16.58
2012-07-17 10:30:00,75722,2509.0,2508.0,16.64
```

3.9 Update

Update CSV database records by getting automatically new records.

```
$ pycr1000 update -h
usage: pycr1000 update [-h] [--timeout TIMEOUT] [--src SRC] [--dest DEST]
                       [--code CODE] [--debug] [--delim DELIM]
                       url table db

positional arguments:
  url                 Specify URL for connection link. E.g. tcp:iphost:port or
                      serial:/dev/ttyUSB0:19200:8N1
  table               The table name used for data collection
  db                  The CSV file database

optional arguments:
  -h, --help          Show this help message and exit
  --timeout TIMEOUT   Connection link timeout (default: 10.0)
  --src_addr SRC_ADDR       Source address ID (default: None)
  --src SRC_NODE            Source node ID (default: 2050)
  --dest_addr DEST_ADDR     Destination address ID (default: None)
  --dest DEST_NODE          Destination node ID (default: 1)
  --code CODE         Datalogger security code (default: 0)
  --debug             Display log (default: False)
  --delim DELIM       CSV char delimiter (default: ,)
```

Examples

If the file does not exist, it will be created automatically.

```
$ pycr1000 update tcp:localhost:1112 'Table1' ./db.csv
Your download is starting.
Packet 0 with 48 records
Packet 1 with 47 records
```

```
Packet 2 with 47 records
Packet 3 with 47 records
Packet 4 with 47 records
Packet 5 with 47 records
...
Packet 1574 with 47 records
Packet 1575 with 47 records
Packet 1576 with 47 records
Packet 1577 with 47 records
Packet 1578 with 47 records
Packet 1579 with 7 records
---------------------------
74261 new records were found
```

1 minute later...

```
$ pycr1000 update tcp:localhost:1112 'Table1' ./db.csv
Your download is starting.
Packet 0 with 1 records
---------------------------
1 new record was found
```

5 seconds later...

```
$ pycr1000 update tcp:localhost:1112 'Table1' ./db.csv
Your download is starting.
Packet 0 with 0 records
---------------------------
No new records were found
```

3.10 Debug mode

You can use debug option if you want to print log and see the flowing data.

```
$ pycr1000 gettime tcp:localhost:1112 --debug
2012-07-17 11:58:11,866 INFO: init client
2012-07-17 11:58:11,866 INFO: Get the node attention
2012-07-17 11:58:11,866 INFO: Create header
2012-07-17 11:58:11,866 INFO: Packet data: 90 01 58 02 00 01 08 02 09 01 00 02 07 08
2012-07-17 11:58:11,866 INFO: Calculate signature for packet
2012-07-17 11:58:11,866 INFO: Calculate signature nullifier to create packet
2012-07-17 11:58:11,866 INFO: Quote packet
2012-07-17 11:58:11,866 INFO: Write: BD 90 01 58 02 00 01 08 02 09 01 00 02 07 08 F6␣
↪86 BD
2012-07-17 11:58:11,866 INFO: Wait packet with transaction 1
2012-07-17 11:58:11,931 INFO: Read packet: A8 02 10 01 08 02 00 01 89 01 00 01 FF FF␣
↪6C 75
2012-07-17 11:58:11,931 INFO: Unquote packet
2012-07-17 11:58:11,931 INFO: Check signature : OK
2012-07-17 11:58:11,931 INFO: Decode packet
2012-07-17 11:58:11,931 INFO: HiProtoCode, MsgType = <0, 89>
```

3.11 API reference

4 High level API

class pycampbellcr1000.device.**CR1000**(*link*, *dest_addr=None*, *dest=1*, *src_addr=None*, *src=2050*, *security_code=0*)

Communicates with the datalogger by sending commands, reads the binary data and parses it into usable scalar values.

Parameters

- **link** – A *PyLink* connection.
- **dest_addr** – Destination physical address (12-bit int) (default dest)
- **dest** – Destination node ID (12-bit int) (default 0x001)
- **src_addr** – Source physical address (12-bit int) (default src)
- **src** – Source node ID (12-bit int) (default 0x802)
- **security_code** – 16-bit security code (default 0x0000)

bye()

Send a bye command.

classmethod from_url(*url*, *timeout=10*, *dest_addr=None*, *dest=1*, *src_addr=None*, *src=2050*, *security_code=0*)

Get device from url.

Parameters

- **url** – A *PyLink* connection URL.
- **timeout** – Set a read timeout value.
- **dest_addr** – Destination physical address (12-bit int) (default dest)
- **dest** – Destination node ID (12-bit int) (default 0x001)
- **src_addr** – Source physical address (12-bit int) (default src)
- **src** – Source node ID (12-bit int) (default 0x802)
- **security_code** – 16-bit security code (default 0x0000)

get_data(*tablename*, *start_date=None*, *stop_date=None*)

Get all data from *tablename* from *start_date* to *stop_date* as ListDict. By default the entire contents of the data will be downloaded.

Parameters

- **tablename** – Table name that contains the data.
- **start_date** – The beginning datetime record.
- **stop_date** – The stopping datetime record.

get_data_generator(*tablename*, *start_date=None*, *stop_date=None*)

Get all data from *tablename* from *start_date* to *stop_date* as generator. The data can be fragmented into multiple packets, this generator can return parsed data from each packet before receiving the next one.

Parameters

- **tablename** – Table name that contains the data.

- **start_date** – The beginning datetime record.
- **stop_date** – The stopping datetime record.

getfile(*filename*)
Get the file content from the datalogger.

getprogstat()
Get programming statistics as dict.

gettime()
Return the current datetime.

list_tables()
List the tables available in the datalogger.

ping_node()
Check if remote host is available.

send_wait(*cmd*)
Send command and wait for response packet.

settime(*dtime*)
Sets the given *dtime* and returns the new current datetime

settings
Get device settings as ListDict

table_def
Return table definition.

class pycampbellcr1000.utils.**Dict**(**args*, ***kwds*)
A dict with somes additional methods.

filter(*keys*)
Create a dict with only the following *keys*.

```
>>> mydict = Dict({"name":"foo", "firstname":"bar", "age":1})
>>> mydict.filter(['age', 'name'])
{'age': 1, 'name': 'foo'}
```

to_csv(*delimiter=u', '*, *header=True*)
Serialize list of dictionaries to csv.

class pycampbellcr1000.utils.**ListDict**
List of dicts with somes additional methods.

filter(*keys*)
Create a list of dictionaries with only the following *keys*.

```
>>> mylist = ListDict([{"name":"foo", "age":31},
...                    {"name":"bar", "age":24}])
>>> mylist.filter(['name'])
[{'name': 'foo'}, {'name': 'bar'}]
```

sorted_by(*keyword*, *reverse=False*)
Returns list sorted by *keyword*.

to_csv(*delimiter=u', '*, *header=True*)
Serialize list of dictionaries to csv.

exception pycampbellcr1000.exceptions.**NoDeviceException**
Can not access to device.

exception pycampbellcr1000.exceptions.**BadSignatureException**
No valid signature.

exception pycampbellcr1000.exceptions.**BadDataException**
No valid data packet.

exception pycampbellcr1000.exceptions.**DeliveryFailureException**
Delivery failure.

5 Low level API

class pycampbellcr1000.pakbus.**PakBus**(*link*, *dest_addr=None*, *dest=1*, *src_addr=None*, *src=2050*, *security_code=0*)
Interface for a pakbus client.

Parameters

- **link** – A *PyLink* connection.
- **dest_addr** – Destination physical address (12-bit int) (default dest)
- **dest** – Destination node ID (12-bit int) (default 0x001)
- **src_addr** – Source physical address (12-bit int) (default src)
- **src** – Source node ID (12-bit int) (default 0x802)
- **security_code** – 16-bit security code (default 0x0000)

compute_signature(*buff*, *seed=43690*)
Compute signature for PakBus packets.

compute_signature_nullifier(*sig*)
Compute signature nullifier needed to create valid PakBus packets.

decode_bin(*types*, *buff*, *length=1*)
Decode binary data according to data type.

decode_packet(*data*)
Decode packet from raw data.

encode_bin(*types*, *values*)
Encode binary data according to data type.

get_bye_cmd()
Create Bye Command packet.

get_clock_cmd(*adjustment=(0, 0)*)
Create Clock Command packet.

Parameters **adjustment** – Clock adjustment (seconds, nanoseconds).

get_collectdata_cmd(*tablenbr*, *tabledefsig*, *mode=4*, *p1=0*, *p2=0*)
Create Collect Data Command packet

Parameters

- **tablenbr** – Table number that contain data.
- **tabledefsig** – Table definition signature.
- **mode** – Collection mode code (p1 and p2 will be used depending on value).
- **p1** – 1st parameter used to specify what to collect (optional)

- **p2** – 2nd parameter used to specify what to collect (optional)

get_fileupload_cmd(*filename*, *offset=0*, *swath=512*, *closeflag=1*, *transac_id=None*)
Create Fileupload Command packet.

Parameters

- **filename** – File name as string
- **offset** – Byte offset into the file or fragment
- **swath** – Number of bytes to read
- **closeflag** – Flag if file should be closed after this transaction
- **transac_id** – Transaction number for continuing partial reads (required by OS>=17!)

get_getprogstat_cmd()
Create Get Programming Statistics Transaction packet.

get_getsettings_cmd()
Create Getsettings Command packet.

get_hello_cmd()
Create Hello Command packet.

get_hello_response(*transac_id*)
Create Hello Response packet.

pack_header(*hi_proto*, *exp_more=2*, *link_state=None*, *hops=0*)
Generate PakBus header.

Parameters

- **hi_proto** – Higher level protocol code (4 bits). 0x0: PakCtrl, 0x1: BMP5
- **exp_more** – Whether client should expect another packet (2 bits)
- **link_state** – Link state (4 bits)
- **hops** – Number of hops to destination (4 bits)

parse_collectdata(*raw*, *tabledef*, *fieldnbr=[]*)
Parse data returned by Collectdata Response.

parse_filedir(*data*)
Parse file directory format.

parse_tabledef(*raw*)
Parse table definition.

quote(*packet*)
Quote the PakBus packet.

read()
Receive packet over PakBus.

unpack_clock_response(*msg*)
Unpack Clock Response packet.

unpack_collectdata_response(*msg*)
Unpack Collect Data Response body.

unpack_failure_response(*msg*)
Unpack Failure Response packet.

unpack_fileupload_response(*msg*)
Unpack Fileupload Response packet.

unpack_getprogstat_response(*msg*)
Unpack Get Programming Statistics Response packet.

unpack_getsettings_response(*msg*)
Unpack Getsettings Response packet.

unpack_hello_response(*msg*)
Create Hello Response packet.

unpack_pleasewait_response(*msg*)
Unpack PeaseWait Response packet.

unquote(*packet*)
Unquote the PakBus packet.

wait_packet(*transac_id=None*)
Wait for an incoming packet.

Parameters **transac_id** – Expected transaction number.

write(*packet*)
Send packet over PakBus.

6 Licence

PyCampbellCR1000 has been developped on licence GNU GPL v3.

https://www.gnu.org/licenses/gpl-3.0.en.html

7 Feedback & Contribute

Your feedback is more than welcome. Write email to the Author.

There are several ways to contribute to the project:

1. Post bugs and feature requests on github.
2. Fork the repository on Github to start making your changes.
3. Test with new CR1000 Type dataloggers.
4. Write a test which shows that the bug was fixed or that the feature works as expected.
5. Send a pull request and bug the maintainer until it gets merged and published. :) Make sure to add yourself to AUTHORS.

To cite this software : "Harrache, S., Darras, L. (2012). PyCampbellCR1000. Communication tools for Campbell CR1000-type Dataloggers (version x.y) [software]. Available at https://pypi.python.org/pypi/PyCampbellCR1000."

8 Changelog

8.1 Version 0.4

Released on 2017-11-28

- Distinction between pakbus address and node in the pakbus protocol.
- Minor bug correction to can visualize help on python 3.x.
- Allow seconds in settime.

8.2 Version 0.3

Released on 2014-03-26.

- Bug correction about communication with NL115, NL120 and NL200 modules using “CSI/O” serial port.
- Support python 3.3 and 3.4

8.3 Version 0.2

Released on 2012-07-26.

- Fixed a bug related to update command (on Windows)
- Used UTC datetime
- Decoded pakbus packet properly
- Other minor bugs fixed

8.4 Version 0.1

Released on 2012-07-18.

First properly tagged release.

Index

U

W