Piko tool – v1.4.1

Get help:

```
Usage: Piko.py --host IP [options]

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -v, --verbose         Verbose mode, print headers
  -q, --quiet           Quiet mode, print only values
  --timestamp           Output timestamp
  --db=DB NAME          database name

  SQL connection management:
    Management functions to connect to an SQL Server

    --my                Connect to a MySQL server
    --my-srv=SERVER     Mysql server IP or hostname
    --my-usr=USERNAME   Mysql username
    --my-pwd=PASSWORD   Mysql password

  Inverter online communication options:
    Define the inverter IP (or DNS name) and the port to connect to

    --host=HOSTNAME     IP address or DNS name
    --port=81           TCP online port
    --id=255            RS485 bus address
    --tref=c800         Temp reference
    --debug             Show data frames

  Inverter history communication options:
    Define the inverter http credential

    --user=USERNAME     http username
    --password=PASSWORD
                        http password

  Inverter data options:
    Select the data to be fetched from inverter

    -s, --status        Get inverter status
    -i, --index         Get inverter total index (Wh)
    -d, --daily         Get inverter daily index (Wh)
    -p, --power         Get inverter current power (W)
    -n, --name          Get inverter name
    -r, --serial        Get inverter serial number
    -m, --model         Get inverter model
    -e, --timers        Get inverter timers
    -y, --history       Get history
    -t, --tech          Get technical data
    -a, --all           Get all realtime data and print as text
    -c, --csv           Get all realtime data and export as csv

  EmonCMS api options:
    Define the emon http credential

    -u EMONURL, --emonurl=EMONURL
                        URL of emoncms to export data
    -k EMONKEY, --emonkey=EMONKEY
                        API key of emoncms to export data
    --emonid=EMONID     Node ID of emoncms to export data
```

Mandatory Parameters are --host to provide inverter IP address or DNS name and, if you changed the RS485 bus address (e.g. if you have more than one inverter connected) the --id to provide the RS485 address (255 is assumed by default).

Get version:

```
# ./Piko.py --version
Piko.py v1.4.1 - 20160218
```

Get Inverter name and serial:

```
# ./Piko.py --host=192.168.1.10 -n -r
Inverter Name   : Piko_name
Inverter SN     : 90xxxKBN00xxx
Inverter Ref    : 10012345
```

Get Inverter model:

```
# ./Piko.py --host=192.168.1.10 -m
Inverter Model  : PIKO 8.3
Inverter String : 2
Inverter Phase  : 3
```

Get Inverter timers:

```
# ./Piko.py --host=192.168.1.10 -e
Total Time      : 17419h57m08s (725 j)
Running Time    : 8411h36m49s
Last Port. upld : 01h46m19s
Last Hist. updt : 00h05m05s
Hist. updt step : 00h15m00s
```

Get the inverter real time data (status, power, indexes and technical data):

```
# ./Piko.py --host=192.168.1.10 -s -p -t -i -d
Inverter Status : 3 (Running)
Inverter Error  : 0
Total energy    : 801296 Wh
Today energy    : 4841 Wh
DC Power        : 2613 W
AC Power        : 2414 W
Efficiency      : 92.4%
DC String 1     : 640.4 V   3.24 A   2075 W   T=a660 (41.21 C)  S=4009
DC String 2     : 599.6 V   0.89 A    538 W   T=a680 (41.14 C)  S=c00a
DC String 3     :   0.0 V   0.00 A      0 W   T=a660 (41.21 C)  S=0003
AC Phase 1      : 234.8 V   3.45 A    791 W   T=9a20 (48.21 C)
AC Phase 2      : 235.3 V   3.43 A    792 W   T=9a20 (48.21 C)
AC Phase 3      : 241.6 V   3.54 A    831 W   T=9a20 (48.21 C)
AC Status       : 28 (001c ---L123)
```

Get value only (e.g. to get result in other script):

```
# ./Piko.py --host=192.168.1.10 -q -i
801296

# ./Piko.py --host=192.168.1.10 -q -d
4841
```

Get csv data (e.g. to import on another app (wsl, ...)):

```
# ./Piko.py --host=192.168.1.10 --csv


PRO,Piko,1,1.3.0,20130730
TIM,2013-07-30T13:22:35.801768,17419h59m39s,8411h39m19s
INF,90xxxKBNxxxxx,Piko_name,192.168.1.10,81,1,PIKO 8.3,2,3
STA,3,Running-MPP,28,---L123,0
ENE,11629195,13803
PWR,4760,4531,95.2
DC1,596.2,3.97,2370,51.21,94e0,4009
DC2,614.9,3.88,2390,51.21,94e0,c00a
DC3,0.0,0.00,0,51.29,94c0,0003
AC1,241.6,6.31,1528,60.14,8540
AC2,236.4,6.16,1466,60.14,8540
AC3,243.5,6.27,1537,60.07,8560
PRT,PIKO-Portal,01h48m49s
HST,00h07m36s,00h15m00s
```

Database management (SQLite3 or MySQL):

Get help:

```
# ./Piko_db.py --help
Usage: Piko_db.py --db dbname [options]

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -v, --verbose         Verbose mode
  -q, --quiet           Quiet mode, print only values
  --db=DB NAME          DB Name

  SQL connection management:
    Management functions to connect to an SQL Server

    --my                Connect to a MySQL server
    --my-srv=SERVER     Mysql server IP or hostname
    --my-usr=USERNAME   Mysql username
    --my-pwd=PASSWORD   Mysql password

  History DB Management:
    Management functions for history db

    --create-history    Drop and recreate history table
    --dump-history      Dump history table

  Realtime DB Management:
    Management functions for realtime db

    --create-realtime   Drop and recreate realtime table
    --dump-realtime     Dump realtime table

  Stat DB Management:
    Management functions for stat db

    --create-stat       Drop and recreate stat table
    --calc-stat         Calculate stats
    --dump-stat         Dump stat table

  Stat options:
    Options for statc calc

    --mins              Calc 10 minutes stats
    --quarter           Calc 15 minutes stats
    --hour              Calc hour stats
    --day               Calc day stats

  Filtering options:
    -f FILTER, --filter-time=FILTER
                        Filter on date/time basis (can include '%')

  Output options:
    -m FORMAT, --format=FORMAT
                        Dump output format [csv|xml|txt|TBL]
    -o FILENAME, --ouput=FILENAME
                        Destination file - Default to stdout
```

Get version:

```
# ./Piko_db.py --version
Piko_db.py v1.4.1 – 20160218
```

Connect to a database:

It's possible to connect to either an SQLite3 file or to a MySQL/MariaDB server (5.0 or higher).

Connect to a SQLite3 database :
--db test.db

Connect to a MySQL server :
--my --my-srv ServerNameOrIP --my-usr piko --my-pwd password --db DatabaseName

Default value for non provided parameters are:
- ServerName : *localhost*
- DatabaseName : *piko*
- Username and password : Database name.

The following exampe use an SQLite3 connection to the test.db file. Same can be done using --my --my-usr 192.168.1.20

Create History and Realtime DB (clear data if already exist in that DB):

```
# ./Piko_db.py --db test.db --create-history --create-realtime
```

Save real time values in Realtime DB record (use also --id if needed):

```
# ./Piko.py --host=192.168.1.10 -t --db test.db
```

Get history from inverter and save it into history DB (merge with current history):

```
# ./Piko.py --host=192.168.1.10 --history --db /mnt/ramdisk/test.db
```

Dump realtime database:

```
# ./Piko_db.py --db /mnt/ramdisk/test.db --dump-realtime --filter-time="%2011-08%" --format csv
...
2011-08-23T13:46:02,613.9,3.61,4252,94e0,71.12,400a,613.8,3.31,0,94e0,71.12,c000,0.0,0.00,0,94e0,71.12,0003,4252,235.3,5.42,1275,8180,90.50,243.1,5.59,1359,8180,90.50,241.1,5.57,1334,81a0,90.38,3968,28,---L123,--.-,93.3,3,RUN, 407922,9815,-,0,-,-,-,-
...
```

```
# ./Piko_db.py --db /mnt/ramdisk/test.db --dump-realtime --filter-time="%2011-08%" --format tbl
...
|2011-08-23T13:46:02|613.9 3.61 4252 94e0 71.12 400a|613.8 3.31    0 94e0 71.12 c000|  0.0 0.00    0 94e0 71.12 0003|4252| |235.3 5.42 1275 8180 90.50|243.1 5.59 1359 8180 90.50|241.1 5.57 1334 81a0 90.38|3968    28 ---L123 --.-| 93.3|  3 RUN|   407922   9815|   -|    0     -     -|  -          -|
...
```

Dump history database:

```
# ./Piko_db.py --db /mnt/ramdisk/test.db --dump-history --filter-time="%2011-08%" -- format txt
...
2011-08-23T13:59:58 577.0 3.91 3819 9421 71.87 400a 577.0 2.71    0 9422 71.87 c000   0.0 0.03    0 9415 71.92 0003 3819 231.0 4.89 1147 85e1 86.12 237.0 5.02 1215 85e8 86.09 236.0 5.01 1204 85ec 86.08 3566    28 ---L123 50.0  93.4 3 RUN 0 0 3 0 0  0
...


# ./Piko_db.py --db /mnt/ramdisk/test.db --dump-history --filter-time="%2011-08%" -- format csv
...
2011-08-23T13:59:58,577.0,3.91,3819,9421,71.87,400a,577.0,2.71,0,9422,71.87,c000,0.0,0.03,0,9415,71.92,0003,3819,231.0,4.89,1147,85e1,86.12,237.0,5.02,1215,85e8,86.09,236.0,5.01,1204,85ec,86.08,3566,28,---L123,50.0,93.4,3,RUN, 0,0,3,0,0,,0,
...
```

As from version 1.2.5 (not shown in sample), power balance (string1 compared to string2) (integer value in % from -100 to 100) is added between effeciency (e.g.:93.4) and running status (e.g.:3 RUN)

Create Stats DB (clear if already exist in that file):

```
# ./Piko_db.py --db test.db --create-stat
```

Calc today Stats (by 10 minutes, by 15 minutes, by hour and by day) :

```
# ./Piko_db.py --db test.db --calc-stat --filter-time=`date +\%Y-\%m-\%d`T\%
--day --hour --quarter --mins
```

Note:Always filter data by day (date until T%) or unexpected result could occur.

Display today Stats :

```
# ./Piko_db.py --db test.db --dump-stat --filter-time "`date +%Y-%m-%d`T%"
--hour
```

```
|2011-09-02T07:00:00|2011-09-02T07:03:00|2011-09-02T07:30:00|   0  100   38    15|   0  129   55    25|   0   65
27    12|   0   64   28    13|   0    0    0     0|19.17 20.14|19.41 20.07|
|2011-09-02T08:00:00|2011-09-02T07:31:00|2011-09-02T08:30:00| 104  221  170   169| 136  264  209   206|  68  133
106   104|  68  131  104   102|   0    0    0     0|21.86 22.98|21.93 23.02|
|2011-09-02T09:00:00|2011-09-02T08:31:00|2011-09-02T09:30:00| 157 1808 1111  1088| 201 1953 1212  1192| 103  965
597   587|  98  988  615   605|   0    0    0     0|30.61 38.57|27.31 31.50|
|2011-09-02T10:00:00|2011-09-02T09:31:00|2011-09-02T10:30:00|1838 2981 2432  2416|1974 3190 2601  2558| 980 1592
1299  1277| 994 1598 1303  1281|   0    0    0     0|46.02 53.50|39.09 46.48|
|2011-09-02T11:00:00|2011-09-02T10:31:00|2011-09-02T11:30:00|2995 3579 3347  3336|3199 3812 3583  3523|1603 2050
1840  1809|1596 1846 1743  1714|   0    0    0     0|59.07 62.57|52.62 57.57|
|2011-09-02T12:00:00|2011-09-02T11:31:00|2011-09-02T12:30:00|3192 3750 3517  3516|3468 4018 3785  3722|2037 2389
2208  2171|1360 1757 1577  1551|   0    0    0     0|60.11 62.50|58.50 62.21|
|2011-09-02T13:00:00|2011-09-02T12:31:00|2011-09-02T13:30:00|3337 3824 3644  3644|3633 4126 3914  3849|2237 2497
2414  2374|1318 1655 1501  1476|   0    0    0     0|59.39 62.38|59.91 62.71|
|2011-09-02T14:00:00|2011-09-02T13:31:00|2011-09-02T14:30:00|1526 3731 3529  3553|1558 4027 3802  3739|1012 2554
2440  2399| 546 1494 1362  1339|   0    0    0     0|57.62 61.36|59.94 62.81|
|2011-09-02T15:00:00|2011-09-02T14:31:00|2011-09-02T15:30:00|1153 3593 3117  3136|1275 3864 3356  3300| 832 2517
2215  2178| 443 1374 1141  1122|   0    0    0     0|57.30 60.26|59.93 62.67|
|2011-09-02T16:00:00|2011-09-02T15:31:00|2011-09-02T16:30:00|1617 3444 2577  2586|1817 3711 2798  2751|1240 2476
1901  1869| 573 1263  897   882|   0    0    0     0|56.79 59.29|59.90 62.69|
|2011-09-02T17:00:00|2011-09-02T16:31:00|2011-09-02T17:30:00|1347 2928 2131  2152|1535 3167 2327  2288| 965 2167
1534  1508| 528 1002  793   780|   0    0    0     0|56.08 59.29|58.77 62.69|
|2011-09-02T18:00:00|2011-09-02T17:31:00|2011-09-02T18:30:00| 574 1928 1350  1360| 676 2187 1506  1481| 408 1362
912   897| 268  846  594   584|   0    0    0     0|54.59 56.64|55.72 59.38|
|2011-09-02T19:00:00|2011-09-02T18:31:00|2011-09-02T19:30:00| 278 1074  743   737| 347 1202  845   831| 183  708
493   485| 160  504  353   347|   0    0    0     0|47.41 49.86|45.39 48.45|
|2011-09-02T20:00:00|2011-09-02T19:31:00|2011-09-02T20:27:00|   0  283   92    92|  19  358  143   133|  12  192
77    72|   7  166   66    62|   0    0    0     0|38.46 42.43|37.96 41.71|
```

The records show

- 3 datetime:
  - 1st is Stat reference datetime ; 2nd is the first sample datetime and 3rd is the last sample datetime.
- 5 groups of power value :
  - **AC Total Power**
  - **DC Total Power**
  - **3 DC String individual power**
  - Each group contain :
    - **Min Power (W)**
    - **Max Power (W)**
    - **Mean Power (W)**
    - **Cumulated energy (Wh)**
- 2 groups of temperature value :
  - **AC Mean temp and AC Max Temp**
  - **DC Mean temp and DC Max Temp**
- 2 values of Efficiency (as of version 1.2.4, xx.xx format (e.g.:92.25), not shown in the provided sample)
  - **Efficiency based on mean power**
  - Efficiency based on Energy value
- 1 value of string balance from -100% to 100% (string1 compared to string2) (as of version 1.2.5, xxxx format (e.g.: -10), not show in the provided sample)

Stats periods :

Stats period are calculated as below :

By 10 mins : H-1:56 to H:05 ; H:06 to H:15 ; H:16 to H:25 ; H:26 to H:35 ; ...
eg: Stats for 10h40 is calculated from data from 10h36 to 10h45

By 15 mins : H-1:53 to H:07 ; H:08 to H:22 ; H:23 to H:37 ; H:37 to H:52
eg: Stats for 10h15 is calculated from data from 10h08 to 10h22

By hour : Stat for hour H is from H-1:30 to H:30
eg: Stats for 10h is calculated from data from 9h30 to 10h30

By day : For the entire day from 00h00 to 23h59

The period is trucated at begin and end by the existence of realtime data.

Eg. stat by day for september 13th start at 7h20 and end at 20:06 in this example:

By day:

```
|2011-09-13T00:00:00|2011-09-13T07:20:00|2011-09-13T20:06:00|   0 4232 1342 17209|   0 4753 1465
18703|   0 3419  897 11452|   0 1565  568  7251|   0    0    0     0|42.60 61.26|37.57 50.21|
```

By hour:

```
....
|2011-09-13T19:00:00|2011-09-13T18:31:00|2011-09-13T19:30:00| 149  759  408   417| 204  872  480
472| 113  641  337   331|  91  231  143   141|   0    0    0     0|35.72 38.29|34.35 36.00|
|2011-09-13T20:00:00|2011-09-13T19:31:00|2011-09-13T20:06:00|   0  167   42    29|  10  220   78
46|   3  129   41    24|   7   91   37    22|   0    0    0     0|30.92 32.79|30.07 31.88|
```

Filter-time :

Filter time is use in all DB query : dump-history, dump-realtime, dump-stat
It's also use in calc-stat as this begin by a query on realtime db.
The format is sqlite syntax to match an ISO date format YYYY-MM-DDTHH:MM:SS
Char % is a wildcard.

Eg: 2011-09-13T% match all time for september 13th

Output format :

All dump output can be done in 4 different format using -m or –format

- txt : space separated
- tbl : | separated and aligned (default)
- csv : commat separated
- xml : xml (self explicit)

How to save data automaticaly :

I use the following structure :

- /opt/solar (HDD or Flash Disk)
  - bin : scripts and binaries
  - etc : configs
  - log : for logs (e.g.:web portal upload log)
  - data : DB
  - graph : for graph output (not yet released)

- /mnt/solar (RAMDISK)
  - Piko_data.db
  - Piko_history.db

A ramdisk is used because I run the server on a flash disk wich don't like to be updated every minute all the time. So, at boot time, a script create a ramdisk, copy the DB files from /opt/solar/data to it. Every 4 hours (cron) and at shutdown, another script copy the ramdisk back to the data directory.
I use 2 SQLite DB files, one for realtime and stats, another for history.

If pushing data to a MySQL DB, ramdisk is not necessary.

From 5:00 to 22:59 :
- Every minute, save the realtime data to the realtime DB
- Every 30 minutes, save the inverter history to history DB
- Every 5 minutes, recalculate the stats for the current day

Every 4 hours : save ramdisk to flash

Crontab file :

```
# PV
SolPath=/opt/solar

# DB
* 5-22 * * * root nice -n 19 $SolPath/bin/Piko.py --host=192.168.1.10 --db /mnt/solar/Piko_Data.db --tech
*/30 5-22 * * * root nice -n 19 $SolPath/bin/Piko.py --host=192.168.1.10 --db /mnt/solar/Piko_History.db --history
*/5 5-23 * * * root nice -n 19 sleep 5 && $SolPath/bin/Piko_db.py --db /mnt/solar/Piko_Data.db --calc-stat
--filter-time=`date +\%Y-\%m-\%d`T\% --day --hour --quarter --mins
0 */4 * * * root nice -n 19 sleep 15 && $SolPath/bin/ramdisk_flush.sh
```

Another script can also be scheduled to upload data on an Internet portal like pvoutput.org or bdpv.fr

MySQL DB:

1.Install and configure a MYSql or MariaDB server. Version 5.0 or higher is required. Code has been tested on MariaDB server 10.0 using a mysql 5.0 client library.

1b.If required, activate TCP/IP on mysql (/etc/my.cnf) to allow connection from another computer.

1c.Define an admin password for MySQL

2.Create a Database for Piko and a user for the connection:

2a.Connect to mysql using the admin account : mysql -p

2b.Create the database (here : db name is piko)

```
create database piko;
```

2c.Create the user (here : username is piko and password is changeme). The user has to be created at localhost and/or at the IP where the script run:

```
  create user 'piko'@'localhost' identified by 'changeme';
  create user 'piko'@'192.168.1.11' identified by 'changeme';
```

2d.Give this user right to the db:

```
  grant all privileges on piko.* to 'piko'@'localhost';
  grant all privileges on piko.* to 'piko'@'192.168.1.11';
  flush privileges;
```

3.Create the tables on the database using Piko_db.py:

```
  ./Piko_db.py --my --my-pwd changeme --create-realtime
  ./Piko_db.py --my --my-pwd changeme --create-history
  ./Piko_db.py --my --my-pwd changeme --create-stat
```

  If needed (non default), provide --my-srv --my-usr and --db parameters

4.If you have data to import from an existing SQLite3 DB:

4a.Be sure to have a functionning MySQL with database and empty tables (run step 3)

4a.Export SQLite data and convert it to MySQL format:

```
  sqlite3 test.db .dump > test.sqlite
  cat test.sqlite | grep INSERT | sed  's/"//g' > test.mysql
```

  It's possible to filter data (e.g. realtime only, or selected year only) by using a second grep in pipe after the INSERT.

4b.Import into MySQL (This can take lot of time):

```
  mysql -h 192.168.1.20 -u piko -p --database=piko <test.mysql
```

5.Test DB query

```
  ./Piko_db.py --my --my-pwd changeme --dump-stat --hour -f 2016-01-%
```