A PostgreSQL Database for the PicoPak Clock Measurement Module

W.J. Riley
Hamilton Technical Services
Beaufort, SC 29907 USA
bill@wriley.com

• Introduction

A relational database can be a useful part of a clock measuring system, particularly if the system has multiple channels and is used for measuring several sources at various times. This paper describes a PostgreSQL database [1] for the PicoPak clock measurement modules [2].

• PostgreSQL

PostgreSQL is a freely-available, ANSI-compliant, well-documented open-source client-server SQL relational database system of very high quality that is capable of handling the storage, access and backup of large sets of clock data. For those not familiar with relational databases, SQL and PostgreSQL, getting started can be rather intimidating. A good place to start is at the PostgreSQL web site (www.postgresql.org) and the many books describing it, particularly Reference [3]. Setting up and using a PostgreSQL database is not a trivial task, but if large quantities of clock data need to be stored and managed, it will pay off in the long run, particularly in ease of access to a portion of a particular run for a certain clock.

Downloading and installing PostgreSQL from the postgresql.org web site is relatively straightforward. Creating the empty database is not particularly difficult if its structure ("schema") is already defined. Data storage is accomplished by the PicoPak clock measurment system user interface, and data access is supported by several means including manual `psql` command line queries and PostgreSQL graphical tools (`pgAdmin`). Backups can be made with `pg_dump` and restored with `pg_restore`.

A key step in devising any relational database is designing the structure of its tables, and that is the main subject of this paper.

• Database Structure

The PostgreSQL database structure for storing PicoPak clock data is shown below. It implements a single system, treating each PicoPak module as a measurement channel. Because PicoPaks are often used with different references, those are identified the same way as the signal sources. PicoPaks can make measurements at any frequency between 5 MHz to 15 MHz, and the database includes an item for their nominal frequency. The PicoPak database also includes additional items to map the module S/N and its COM port number.

• Database Schema

The PicoPak PostgreSQL database schema is shown below in the form of a `.sql` file that can be used to create the tables. The `measurements` table holds the clock data. The `measurement_list` table contains information about each measurement run. The `measurement_modules` table has an entry for each PicoPak, and the `clock_names` table has information about each clock under test or reference source.

```
CREATE TABLE measurements
(
     mjd       NUMERIC(12,6)       NOT NULL,
     sn        INTEGER             NOT NULL,
     meas      DOUBLE PRECISION    NOT NULL
);

CREATE TABLE measurement_list
(
     meas_id        INTEGER                        NOT NULL,
     sn             INTEGER                        NOT NULL,
     sig_id         INTEGER                        NOT NULL,
     ref_id         INTEGER                        NOT NULL,
     frequency      REAL                           NOT NULL,
     description    CHARACTER VARYING(256),
     begin_mjd      NUMERIC(12,6)                  NOT NULL,
     end_mjd        NUMERIC(12,6),
     tau            REAL                           NOT NULL
);

CREATE TABLE measurement_modules
(
     sn        INTEGER                  NOT NULL,
     com       INTEGER,
     computer  CHARACTER VARYING(256),
     active    BOOLEAN
);

CREATE TABLE clock_names
(
     clock_name     CHARACTER VARYING(256),
     clock_id       INTEGER                   NOT NULL,
     clock_type     CHARACTER VARYING(256),
     description    CHARACTER VARYING(256)
);
```

The `sn` is the PicoPak module S/N number. The PicoPak clock `measurements` data table contains that, along with the `meas` value and its `mdj` timetag. The `measurement_list` table has an entry for each measurement run with its serial `meas_id` number, the `sn`, `sig_id` for the signal source and `ref_id` for the reference source, the nominal signal `frequency`, a `description` of the run, its data `tau`, and its start and end times, `begin_mjd` and `end_mjd`. The `measurement_modules` table contains the module `sn`, its `com` port number, `computer` name, and a flag to indicate whether it is currently `active`. The `clock_names` table contains a list of every clock that has been connected to the system, with their serial `clock_id`, `clock_name`, `clock_type` and `description`. No default values are included, there are no explicit `SERIAL` items, nor are there any `UNIQUE` or `PRIMARY KEY` constraints, so it is up to the user and/or user interface to enforce those, particularly the unique `meas_id` and `clock_id`. The module `sn`s are also unique, and serve as de facto channel numbers.

Only the essential columns have NOT NULL constraints, which are enforced by the PicoPak user interface. It is best, however, to use the measurement table description and the clock_names table clock_name, clock_type and description to help find the desired clock and measurement during database access. The PicoPak user interface will always enter values for com, computer and active in the measurement_modules table. The measurement data (meas column of the measurements table) is formatted as a DOUBLE PRECISION floating point number, which holds the phase data in units of seconds as are the frequency and tau columns of the measurement_list table.

This is the minimal PicoPak database structure. The following columns serve as the primary key for their respective tables:

Table	Primary Key	Index
measurements	sn, mjd	cs4
measurements_list	meas_id	cs3
measurement_modules	sn	cs2
clock_names	clock_id	cs1

These are constrained as PRIMARY KEYs and indices are added for them to speed up database access.

• Creating the Database Tables

The empty ppd PicoPak database can be created by opening a command window, navigating to the PostgreSQL bin folder and executing the following command:

```
createdb ppd
```

The empty database tables can then be entered into the database by copying the schema above to a file called ppd.sql in that folder and executing the following command:

```
\i ppd.sql
```

The database tables can be listed with by launching the psql program and entering the command:

```
\dt
```

The structure of each of the tables can be examined with the command:

```
\d tablename
```

where tablename is the name of one of the tables.

A psql list of the PicoPak database table descriptions is shown in Figure 1:

```
C:\Program Files\PostgreSQL\9.2\bin\psql.exe

ppd=# \d measurements
      Table "public.measurements"
 Column |       Type       | Modifiers
--------+------------------+-----------
 mjd    | numeric(12,6)    | not null
 sn     | integer          | not null
 meas   | double precision | not null
Indexes:
    "cs4" PRIMARY KEY, btree (sn, mjd)


ppd=# \d measurement_list
         Table "public.measurement_list"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 meas_id     | integer                | not null
 sn          | integer                | not null
 sig_id      | integer                | not null
 ref_id      | integer                | not null
 frequency   | real                   | not null
 description | character varying(256) |
 begin_mjd   | numeric(12,6)          | not null
 end_mjd     | numeric(12,6)          |
 tau         | real                   | not null
Indexes:
    "cs3" PRIMARY KEY, btree (meas_id)


ppd=# \d measurement_modules
      Table "public.measurement_modules"
  Column  |          Type          | Modifiers
----------+------------------------+-----------
 sn       | integer                | not null
 com      | integer                |
 active   | boolean                |
 computer | character varying(256) |
Indexes:
    "cs2" PRIMARY KEY, btree (sn)


ppd=# \d clock_names
            Table "public.clock_names"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 clock_name  | character varying(256) |
 clock_id    | integer                | not null
 clock_type  | character varying(256) |
 description | character varying(256) |
Indexes:
```

Figure 1. PicoPak Database Description

- **Populating the Database Tables**

While the database tables can be populated and edited manually using `psql` using the INSERT INTO and/or UPDATE commands, that is normally handled by the PicoPak Windows® graphical user interface as described in the next sections.

- **Database Section of the PicoPak Configuration File**

Access to the PicoPak Database functionality is controlled by the `[Database]` section of the PicoPak.ini configuration file as shown in Figure 2.

The DB, User, Host and Password items control access to the PostgreSQL database, and the Database flag enables or disables PicoPak user interface database functionality. This content is entered and edited on the PicoPak Configure screen.

```
[Database]
DB=ppd
User=postgres
Host=192.168.2.6
Password=root
Database=1
```

Figure 2. Database Section of PicoPak.ini

• PicoPak Database Screen

The PicoPak Database screen is hidden unless the Database flag is set to 1, which adds a Dbase button to the PicoPak main screen when it is launched, as shown in Figure 3. Pressing that button brings up the PicoPak Database screen shown in Figure 4. That should be done before starting a measurement whose results are to be stored in the database.

PicoPak PostgreSQL database functionality is then activated by checking the Database Active checkbox on the Database screen. The top of the Database screen contains a list of all the clock sources that are present in the PicoPak database. The pair of Signal Source # and Reference Source # controls below the list allow those to be selected as the signal and reference source for the measurement. The signal and reference sources can be the same to perform a coherent (e.g., noise floor) measurement. Signal and Reference Source #0 is the default.

New sources can be entered into the database with the Name, Type and Description edit boxes and the Enter pushbutton. The next source number will be used, and the entry will appear in the list.

The changes can be accepted with the OK button or cancelled with the Cancel button, either of which closes the Database screen.

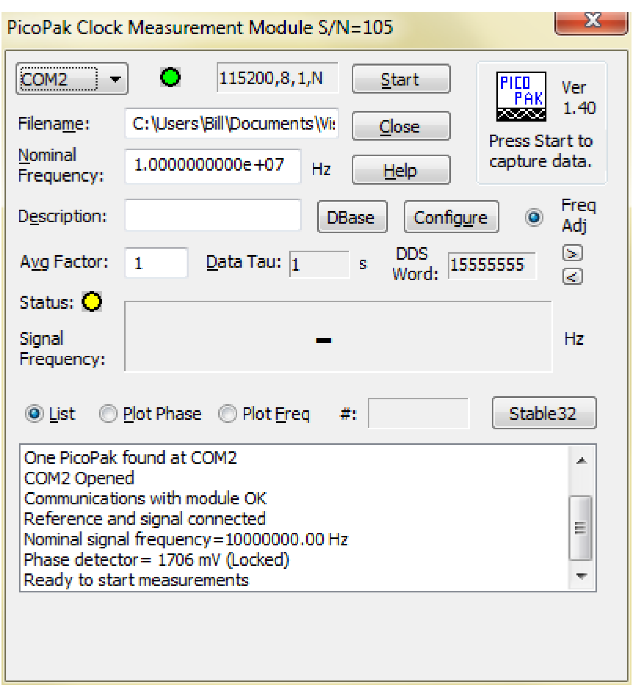

Figure 3. PicoPak Main Screen with Dbase Button

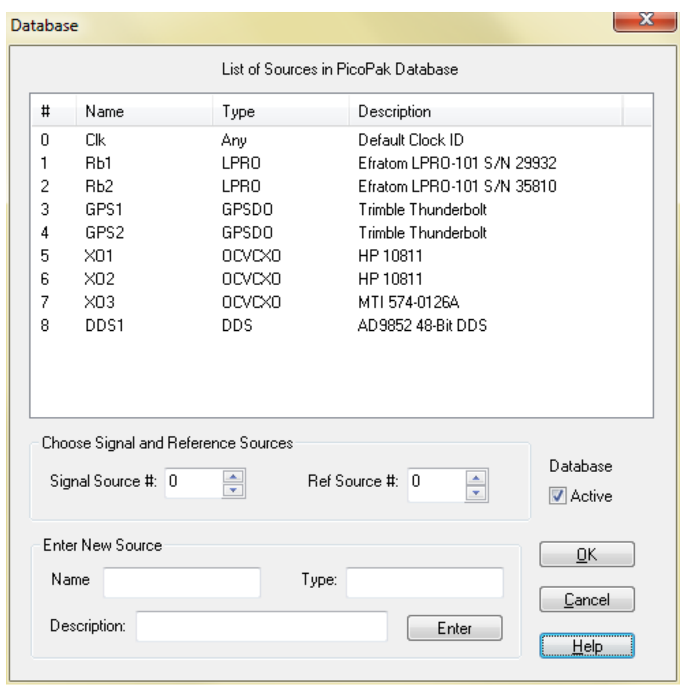

Figure 4. PicoPak Database Screen

Once the source selections have been made, and the database is activated, subsequent measurements will be stored in the PicoPak PostgreSQL database. Each timetagged point will be stored in the measurements table along with the module S/N. The `measurement_list` table will contain a measurement #, the module S/N, the source #s, the nominal frequency and tau, the measurement description, and its start MJD. The measurement end MJD will be added when the run is stopped. Access to the stored data is in the measurements table is generally performed knowing the desired module S/N and MJD range. Those can be obtained from the `measurement_list` table by selecting a particular run from its clock sources and/or description. PicoPak database measurement storage is optional, and does not change the normal data storage to disk.

Please note that the PicoPak database is not compatible with the Stable32 Database function [4], which works with the Timing Solutions\Symmetricom\Microsemi MMS system [5].

• PicoPak Database Access

PicoPak database access can be accomplished manually by using the PostgreSQL `psql` program. For example, with `psql` connected to the `ppd` database, if you want all the data for S/N 103 to be written to the file `foo.txt`, use the command:

```
ppd=# \o foo.txt
```

to redirect the `psql` output to the file, and:

```
ppd=# SELECT meas FROM measurements WHERE sn=103;
```

to write the data to the file, which can be read into Stable32 for analysis. You can restore the screen as the output device with:

```
ppd=# \o
```

More complicated `SELECT`s with a date range are, of course, possible.

• Deleting a Measurement Run

A measurement run can be deleted from the PicoPak database with the following psql command:

```
DELETE FROM measurement_list WHERE meas_id=##;
```

where ## is the ID number of the measurement to be removed.

That command will remove the run from the measurement list. A similar command can be used to delete the actual data from the `measurements` table, but may not accomplish much, especially if it was a short aborted run. The disk space of a deleted measurement is not necessarily reused by the operating system.

• The PicoSQL Application

PicoSQL is a Windows® is a graphical user interface for the PicoPak database. It supports connecting to a PicoPak database, selecting a signal or reference source, displaying a list of the associated measurement runs, selecting one of those, and extracting data and plotting from it over a particular date range into a Stable32-compatible data file while applying an optional averaging factor. The extracted data can be examined with Windows Notebook and/or analyzed with Stable32. The PicoSQL main screen is shown in Figure 5.

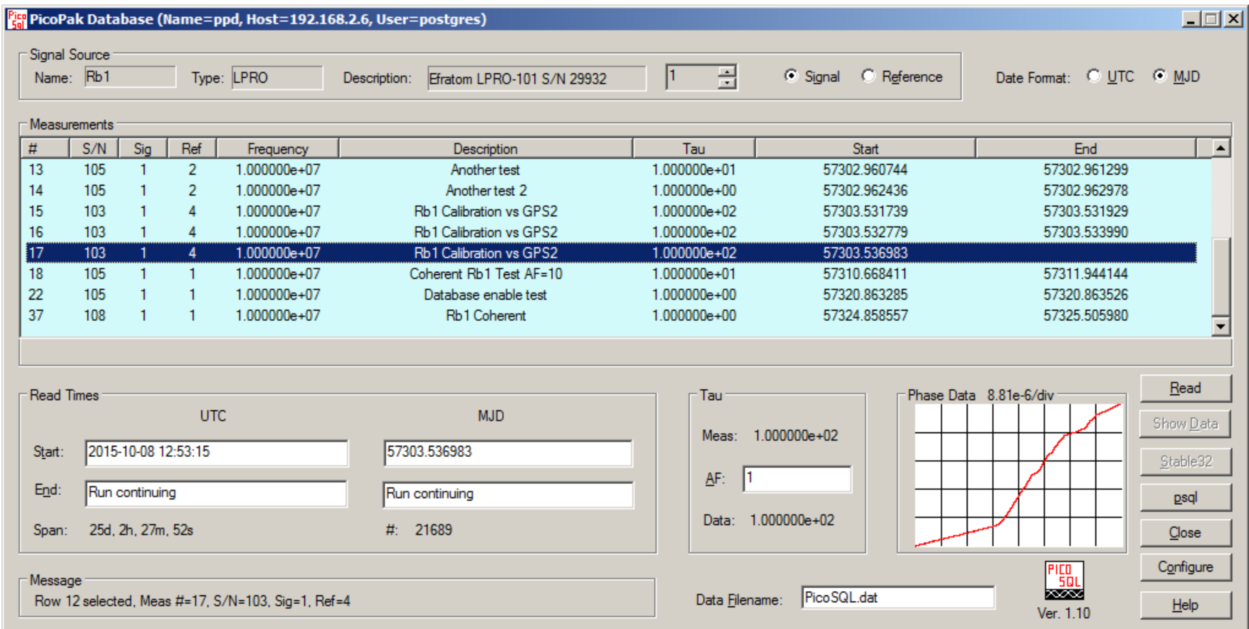

Figure 5. PicoSQL Main Screen

• Conclusion

The capability to store PicoPak clock data in a PostgreSQL relational database is an attractive additional capability, especially along with the PicoSQL user interface application to extract the desired clock data from the database.

• References

1. See the PostgreSQL web site, more specifically: http://www.postgresql.org/about/.
2. W.J. Riley, "The PicoPak Clock Measurement Module", Hamilton Technical Services, April 2015.
3. N. Matthew and R. Stones, *Beginning Databases with PostgreSQL*, ISBN 1-59059-478-9, Apress, 2005.
4. Data Sheet, Stable32 Frequency Stability Analysis, Hamilton Technical Services, July 2008.
5. Data Sheet, Multi-Channel Measurement System, Microsemi, Inc.

File: A PostgreSQL Database for the PicoPak Clock Measurement Module.doc

W.J. Riley
Hamilton Technical Services

October 15, 2015
Rev. A November 3, 2015