Databases for Clock Data

W.J. Riley
Hamilton Technical Services
Beaufort, SC 29907 USA

• Introduction

While clock data can be stored simply as a sequence of phase values, significant advantages accrue if it they entered into a database along with supporting information about the measurement. Doing so requires some initial study, design and setup work, and somewhat complicates the data acquisition process, but this effort pays significant dividends during subsequent data access and analysis. This paper describes how a formal relational database can be used to store and retrieve clock data, makes suggestions as to how the database is organized, and illustrates the advantages of that approach, especially when a large amount of clock data needs to be collected, stored, archived, retrieved and processed.

• Databases

A database is a structured means for storing data; a relational database is organized so that different aspects of the data can be queried to extract information about it. Herein we discuss databases based on an associated structured query language (SQL). Such databases are comprised of a set of tables each having several columns containing a certain type of data along with one or more indices that identify a particular row of data. Data is stored and accessed using these indices within the overall structure (the schema) so that values can be stored and accessed flexibly with minimal duplication. Database software implements its operation, usually in a client-server manner, and supports database entries, queries, editing, archiving, backup, maintenance and other such operations, while client software, using the SQL query language, implements user interactions. The details of the database operations can be hidden within a graphical user interface.

• Clock Data

The most essential clock measurement data is, of course, a series of phase readings, preferably with associated timetags. Herein we assume that phase rather than frequency data are measured and stored, that those values have units of seconds with sufficient range to avoid ambiguity and sufficient resolution to show the significant variations. In most cases the phase data are equally-spaced in time and they include MJD timetags that uniquely identify each datum.

• Supporting Data

A principal advantage of using a database is that supporting information can be stored along with the measurement data. For example, database tables can store a list of the measurement channels with entries such an active flag, a list of the clock sources with entries such as their S/N, and a list of the measurements with entries such as a test description. These supporting data are organized according to the data base schema in such a way as to allow them to be accessed by queries according to their relationships.

Other information can be added via additional tables as needed for a particular application. For example clock monitor and environmental readings can be recorded and correlated with the phase data either within the same database or from another measuring system and its compatible database.

• Database Usage

Because of these advantages of storing clock data together in a formal relational database, it is much easier to utilize it via flexible queries to gain insights into both individual and mutual clock performance and to observe their correlations with both internal dependencies and external environmental sensitivities. One can more easily detect similarities in clock behavior and compare their variations over time or against monitor records. The database also imposes discipline on clock data labeling, formatting, storage and archiving that enhancing the accessibility and value of those records. Once established, clock measurements become easier and more consistent, and the resulting data becomes easier to access and utilize.

• SQL Databases

There are several popular SQL databases, both commercial and free, available on Windows, Linux and other platforms. Among the former are Windows Access and SQL Server, and Oracle Database. Among the later are PostgreSQL and MySQL. Herein we will mainly discuss PostgreSQL [*].

• 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 [1] and the many books describing it, particularly Reference [2]. 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, perhaps along with associated monitor readings.

The PostgreSQL database system includes the database server, command line and graphical utilities, and interfaces for programming languages such as C and Python.

Downloading and installing PostgreSQL from the postgresql.org web site is relatively straightforward. Creating the empty database is not particularly difficult if its schema is already defined. Managing the database does require occasional effort regarding making backups and performing software updates.

* PostgreSQL has some slight advantages regarding large systems, speed, complex queries, SQL compliance, security, programming language support and robustness, but either is satisfactory for a clock database system.

• Database Server

The database server can be collocated on the same computer that controls the clock measuring system or set up on a LAN-connected dedicated computer. The later is preferred for reasons of robustness and access by multiple measuring systems and users. Attention should be paid to issues like uninterruptable power and security, and redundant RAID hard drives could be used. The PostgreSQL database server offers adequate security for most clock data requirements. Measurement system, database server and analysis workstation computer clocks should be closely synchronized. Multiple databases can be supported in the same cluster on the server.

There are rather frequent updates to the PostgreSQL database package, and it can be problematic to keep the server, utilities and all related custom software up to date. The most important thing seems to be that all such software use compatible versions.

• Database Schema

A key step in devising any relational database is designing the structure of its tables. Herein we describe a fairly minimal schema for storing clock measurements that has been proven effective for that purpose. In particular, as an example, it was simplified and adapted from a database used with PicoPak clock measurement modules [3], which was in turn based on one used by the Timing Solutions/Symmetricom/Microsemi MMS clock measurement system [4].

In designing a database schema one must think about how queries will be written to access the contents efficiently and without duplication. Database tables typically include an index column that serves as its primary access key; other columns can also be designated as keys and indexed by the database to facilitate fast access.

An example of a PostgreSQL clock database schema is shown in Figure 1 in the form of a `.sql` file that can be used to create the tables. The entries define the database tables, their column names and data types, and whether they may not have null values.

The `measurements` table holds the clock data. It is the largest table by far even though it contains the minimum amount of information needed (MJD, channel and phase values) to uniquely hold a measurement.

The `measurement_list` table contains information about each measurement run.

The `measurement_channel` table has an entry for each measurement channel in the system.

The `clock_names` table has information about each clock under test or reference source.

The `notes` table holds user-entered remarks regarding measurement events.

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

CREATE TABLE measurement_list
(
    meas_id INTEGER NOT NULL,
    ch 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_channels
(
    ch INTEGER NOT NULL,
    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)
);

CREATE TABLE notes (
    meas_id integer NOT NULL,
    mjd numeric(12,6) NOT NULL,
    note character varying(255)
);
```

Figure 1. Example of a PostgreSQL Database Schema for Clock Data

In this database schema, `ch` is the measurement system channel number, which could also be the S/N of a measurement module. The clock `measurements` data table contains that number, 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 channel, `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_channels table contains the channel ch and a flag to indicate whether it is currently active. Additional items can be added to this table depending on the measurement system hardware. The clock_names table contains a list of every clock that has been connected to the system, with its serial clock_id, and its clock_name, clock_type and description. . A user can make an entry into the notes table during a measurement to indicate and timetag an event.

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 ch numbers.

Only the essential columns have NOT NULL constraints, which are enforced by the user interface. One can 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 user interface will always enter values for 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 database structure. The following columns serve as the primary key for their respective tables; these are constrained as PRIMARY KEYs and indices are added for them to speed up database access.

Table I Example PostgreSQL Clock Database Keys

Database Table	Primary Key(s)
measurements	ch, mjd
measurements_list	meas_id
measurement_channels	ch
clock_names	clock_id
notes	meas_id, mjd

A 1-line listing of this example's clock database tables is shown in Figure 2.

```
             List of relations
 Schema |        Name         | Type  | Owner
--------+---------------------+-------+-------
 public | clock_names         | table | bill
 public | measurement_list    | table | bill
 public | measurement_channels| table | bill
 public | measurements        | table | bill
 public | notes               | table | bill

clock_name | clock_id | clock_type |        description
-----------+----------+------------+----------------------------
 Rb1       |        1 | LPRO       | Efratom LPRO-101 S/N 29932

meas_id | ch | sig_id | ref_id | frequency |   description    |  begin_mjd   |   end_mjd    | tau
--------+---+--------+--------+-----------+------------------+--------------+--------------+-----
     18 |  4 |     12 |      1 |     1e+07 | GPSDO Comparison | 57531.552089 | 57531.567899 |  20

sn  | com | active | computer |  type
----+-----+--------+----------+---------
105 |   7 | f      | LAB      | PicoPak

     mjd      | sn  |         meas
--------------+-----+----------------------
 57527.791612 | 103 | 6.96539788845214e-11

meas_id |     mjd      |     note
--------+--------------+---------
    105 | 57597.817815 | A/C on
```

Figure 2. Listing of the Example PostgreSQL Clock Database Tables

• Database Processes

One must devise software processes following the logic of the database schema to store and recall all desired items of information, utilizing the database schema keys to compose a sequence of commands and queries. Note that these queries are for illustrative purposes only and, while they resemble actual C or Python code, they should not be taken literally.

System Setup

Enter or append inactive measurement channels (or modules) into the system by executing the following command:

```
INSERT INTO measurement_channels(ch, active)
VALUES(%s, %s), (channel, FALSE)
```

Clock Entry

Determine the next `clock_id` for the `clock_names` table with the command:

```
SELECT clock_id FROM clock_names ORDER BY clock_id DESC
LIMIT 1
```

Enter the corresponding `clock_name`, `clock_type` and `description` into the `clock_names` table with the command:

```
INSERT INTO clock_names(clock_name, clock_type,
description) VALUES(%s, %s, %s), (name, type, desc)
```

Measurement Run Start

Determine the next `meas_id` for the `measurements_list` table with the command:

```
SELECT meas_id FROM measurement_list ORDER BY meas_id DESC
LIMIT 1
```

Enter the measurement parameters into the `measurements_list` table with the command:

```
INSERT INTO measurement_list(meas_id, ch, sig_id,
ref_id,frequency, description, begin_mjd, tau) VALUES(%s,
%s, %s, %s, %s, %s, %s, %s), (meas, chan, sig, ref, hz,
desc, mjd, tau)
```

Set the corresponding measurement_channels active flag to TRUE with the command:

```
UPDATE measurement_channels SET active=TRUE WHERE ch=%s,
(chan)
```

Note Entry

Enter a note for a certain measurement run into the `notes` table with the command:

```
INSERT INTO notes(meas_id, mjd, note) VALUES(%s, %s,
%s), (meas, mjd, text)
```

Measurement Run End

Update the measurements_list table for a certain measurement run by entering the `mjd_end` with the command:

```
UPDATE measurement_list SET end_mjd=%s WHERE meas_id=%s,
(mjd, meas)
```

Set the corresponding measurement_channels active flag to FALSE with the command:

```
UPDATE measurement_channels SET active=FALSE WHERE ch=%s,
(chan)
```

Clock Data Access

Obtain the clock phase data for a particular channel `ch` and date `mjd` by executing the following command over the desired date range:

```
SELECT  meas  FROM  measurements  WHERE  ch=%s  AND  mjd=%s,
(chan, mjd)
```

• Database Storage

Data storage is best accomplished by embedding it into the same user interface that controls and manages the clock measuring system.

• Database Access Tools

Database access can be accomplished manually, with PostgreSQL or 3rd-party command line or graphical tools, or (preferably) with a custom user interface. Manual database access can use the PostgreSQL `psql` program and SQL commands or the `pgAdmin` GUI.

A database server, perhaps Linux based using Apache, can employ a php script [5] with the PostgreSQL support module, the GD graphics package and the PHPlot plotting package [6] to implement a user interface to the database [7]. Those tools are all well-supported and free.

• Clock Database Strategies

The most straightforward situation is a custom in-house clock measuring system, perhaps using National Instruments Lab Windows/CVI [8]. In that case, one can embed the data storage within the system controller user interface, set up a LAN-connected database server, and create the desired access means for analyst workstations.

If the clock measuring system is an in-house product, the same elements can be made part of it, with customer support for its commissioning and operation.

If the measuring system is a procured product with its own database support, one can simply use it as-is, modify the database schema as required, and/or write a custom user interface to satisfy specific needs.

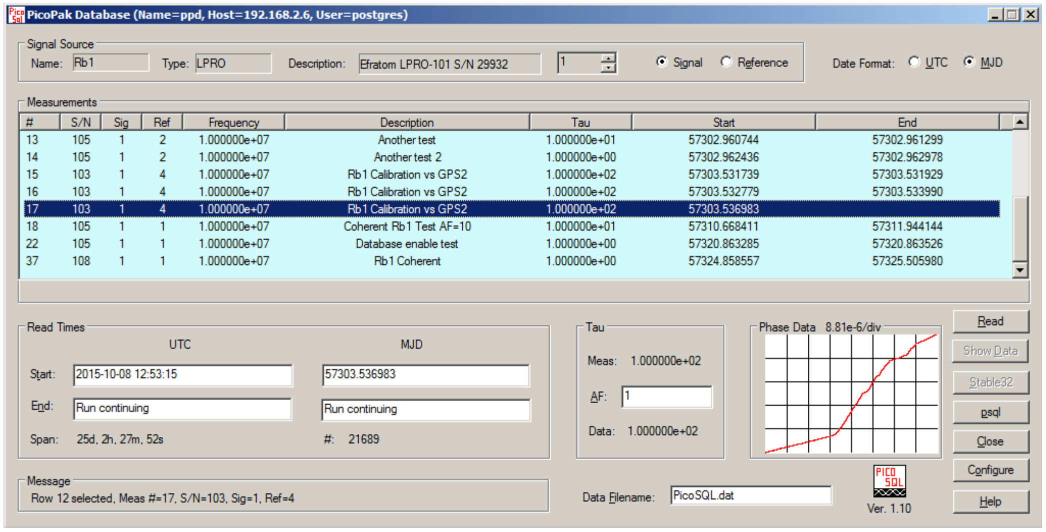
If, however, the measuring system is a procured product without database support, adding that will be more difficult. A possible approach is to write custom software to read the products native data file and store it into a custom database supported by a custom user interface.

• Supporting Applications

A clock measurement and analysis system benefits from a complete suite of supporting applications:

1. A user interface to control the clock measuring system itself that allows clocks to be entered, measurements configured and started, data to be collected and entered into the database, and the measurement run to be ended.
2. A similar user interface, either integrated or separate, to collect monitor data.
3. A user interface, either PC or web based (or both), to monitor the ongoing measurements.
4. A user interface for flexibly extracting previous clock and monitor data from the database. Provisions should be available to access clock and monitor data for a particular clock, reference and measurement over all or a portion of the run. Provisions should be available for plotting these data and writing them to a file for further analysis.
5. General purpose applications for analysis and plotting.
6. Specialized frequency stability analysis applications.
7. Utilities for database backup and maintenance.

A example of a Windows graphical user interface for a PostgreSQL clock database is shown in Figure 3 [3]. It supports connecting to the database, selecting a signal or reference source, displaying a list of the associated measurement runs, selecting one of those, extracting data and plotting from it over a particular date range, and writing it to to a Stable32-compatible data file while applying an optional averaging factor. The extracted data can then be examined with Windows Notebook and/or analyzed with Stable32.



PicoPak Database (Name=ppd, Host=192.168.2.6, User=postgres)

Signal Source
Name: Rb1 Type: LPRO Description: Efratom LPRO-101 S/N 29932 1 [✓ Signal] Reference
Date Format: UTC [✓ MJD]

Measurements

#	S/N	Sig	Ref	Frequency	Description	Tau	Start	End
13	105	1	2	1.000000e+07	Another test	1.000000e+01	57302.960744	57302.961299
14	105	1	2	1.000000e+07	Another test 2	1.000000e+00	57302.962436	57302.962978
15	103	1	4	1.000000e+07	Rb1 Calibration vs GPS2	1.000000e+02	57303.531739	57303.531929
16	103	1	4	1.000000e+07	Rb1 Calibration vs GPS2	1.000000e+02	57303.532779	57303.533990
17	103	1	4	1.000000e+07	Rb1 Calibration vs GPS2	1.000000e+02	57303.536983	
18	105	1	1	1.000000e+07	Coherent Rb1 Test AF=10	1.000000e+01	57310.668411	57311.944144
22	105	1	1	1.000000e+07	Database enable test	1.000000e+00	57320.863285	57320.863526
37	108	1	1	1.000000e+07	Rb1 Coherent	1.000000e+00	57324.858557	57325.505980

Read Times

	UTC	MJD
Start:	2015-10-08 12:53:15	57303.536983
End:	Run continuing	Run continuing
Span:	25d, 2h, 27m, 52s	#: 21689

Tau
Meas: 1.000000e+02
AF: 1
Data: 1.000000e+02

Phase Data 8.81e-6/div

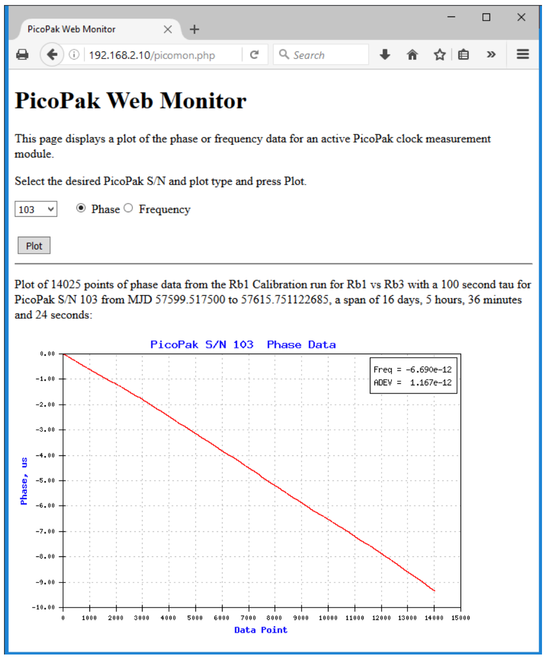
Message
Row 12 selected, Meas #=17, S/N=103, Sig=1, Ref=4

Data Filename: PicoSQL.dat

Figure 3. PicoPak Database Interface Main Screen

Similar Windows, Linux and National Instruments workstation and server interface applications can be written to support clock measurement system operation, database clock entry, clock and monitor data storage and retrieval, archiving, backup, etc. A suite of such custom programs greatly facilitates these activities. They are relatively straightforward to write because underlying tools are available for the database and network processes, as well as for plotting and other user interface elements, particularly with high-level scripting languages like Python and php.

A web-based monitor for a PostgreSQL database in shown in Figure 4 [9]. It is hosted on the same Linux computer as the database server, and can display a plot of the phase or frequency data for a selected PicoPak module during a measurement run.


PicoPak Web Monitor

192.168.2.10/picomon.php

PicoPak Web Monitor

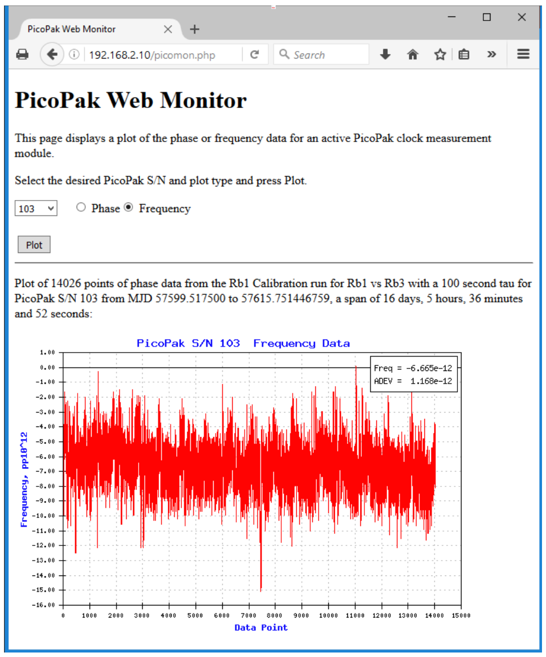
This page displays a plot of the phase or frequency data for an active PicoPak clock measurement module.

Select the desired PicoPak S/N and plot type and press Plot.

103 [✓ Phase] Frequency

Plot

Plot of 14025 points of phase data from the Rb1 Calibration run for Rb1 vs Rb3 with a 100 second tau for PicoPak S/N 103 from MJD 57599.517500 to 57615.751122685, a span of 16 days, 5 hours, 36 minutes and 24 seconds:


PicoPak Web Monitor

192.168.2.10/picomon.php

PicoPak Web Monitor

This page displays a plot of the phase or frequency data for an active PicoPak clock measurement module.

Select the desired PicoPak S/N and plot type and press Plot.

103 Phase [✓ Frequency]

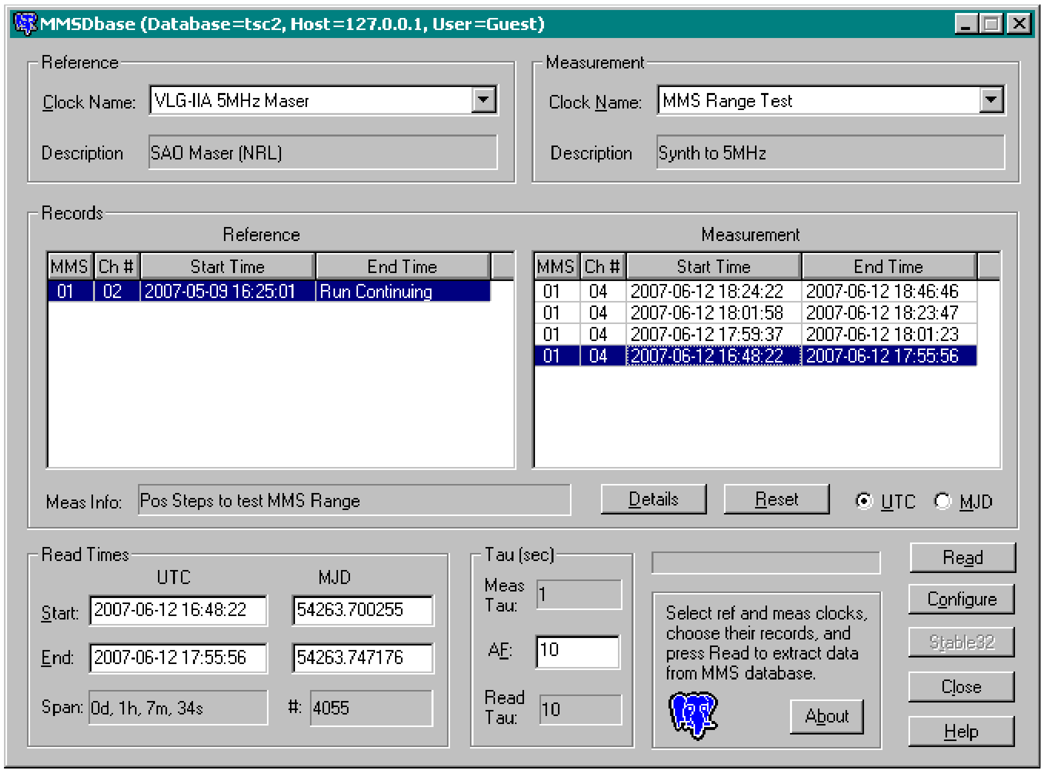
Plot

Plot of 14026 points of phase data from the Rb1 Calibration run for Rb1 vs Rb3 with a 100 second tau for PicoPak S/N 103 from MJD 57599.517500 to 57615.751446759, a span of 16 days, 5 hours, 36 minutes and 52 seconds:

Phase Data Frequency Data

Figure 4. PicoPak Database Web Monitor

Another example of a database interface application is shown in Figure 5. The screen displays a list of the measurement records for the selected reference and measurement clocks, and can read the data for the selected clocks for a selected date range.


MMSDbase (Database=tsc2, Host=127.0.0.1, User=Guest)

Reference — Clock Name: VLG-IIA 5MHz Maser; Description: SAO Maser (NRL)

Measurement — Clock Name: MMS Range Test; Description: Synth to 5MHz

Records

Reference

MMS	Ch #	Start Time	End Time
01	02	2007-05-09 16:25:01	Run Continuing

Measurement

MMS	Ch #	Start Time	End Time
01	04	2007-06-12 18:24:22	2007-06-12 18:46:46
01	04	2007-06-12 18:01:58	2007-06-12 18:23:47
01	04	2007-06-12 17:59:37	2007-06-12 18:01:23
01	04	2007-06-12 16:48:22	2007-06-12 17:55:56

Meas Info: Pos Steps to test MMS Range

Details Reset [✓ UTC] MJD

Read Times

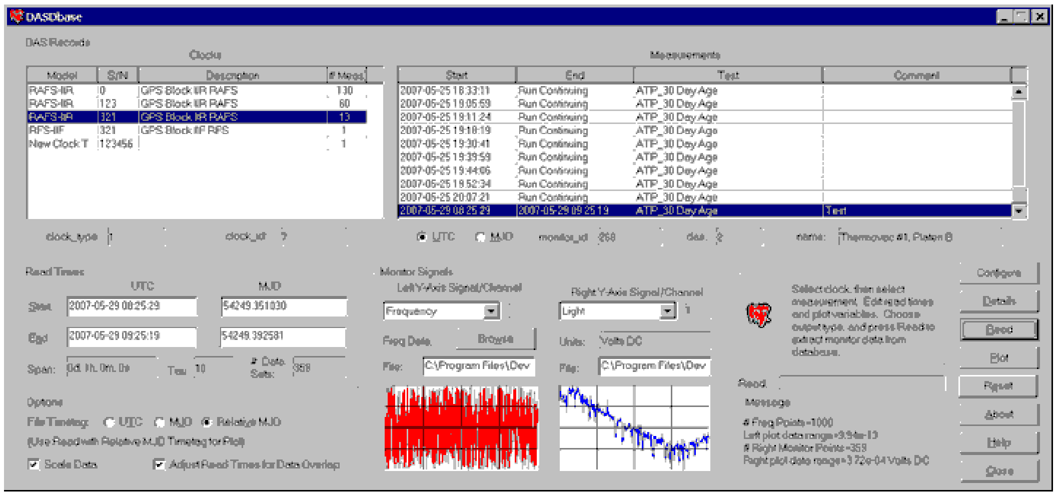
	UTC	MJD
Start:	2007-06-12 16:48:22	54263.700255
End:	2007-06-12 17:55:56	54263.747176
Span:	0d, 1h, 7m, 34s	#: 4055

Tau (sec) — Meas Tau: 1; AF: 10; Read Tau: 10

Select ref and meas clocks, choose their records, and press Read to extract data from MMS database.

Figure 5. MMS Clock Database Interface Main Screen

The interface application of Figure 6 allows clocks and measurements to be selected and supports downloading their phase or converted fractional frequency data along with one selected channel of monitor data.



Clocks

Model	S/N	Description	# Meas.
RAFS-IIR	0	GPS Block IIR RAFS	130
RAFS-IIR	123	GPS Block IIR RAFS	60
RAFS-IIR	321	GPS Block IIR RAFS	13
RFS-IIF	321	GPS Block IIF RFS	1
New Clock T	123456		1

clock_type 1 clock_id 7

Measurements

Start	End	Test	Comment
2007-05-25 18:33:11	Run Continuing	ATP_30 Day Age	
2007-05-25 19:05:59	Run Continuing	ATP_30 Day Age	
2007-05-25 19:11:24	Run Continuing	ATP_30 Day Age	
2007-05-25 19:18:19	Run Continuing	ATP_30 Day Age	
2007-05-25 19:30:41	Run Continuing	ATP_30 Day Age	
2007-05-25 19:39:59	Run Continuing	ATP_30 Day Age	
2007-05-25 19:44:06	Run Continuing	ATP_30 Day Age	
2007-05-25 19:52:34	Run Continuing	ATP_30 Day Age	
2007-05-25 20:07:21	Run Continuing	ATP_30 Day Age	
2007-05-29 08:25:29	2007-05-29 09:25:19	ATP_30 Day Age	Test

[✓ UTC] MJD monitor_id 268 das. 2 name: Thermovac #1, Platen B

Read Times

	UTC	MJD
Start	2007-05-29 08:25:29	54249.351030
End	2007-05-29 09:25:19	54249.392581

Span: 0d. 1h. 0m. 0s Tau 10 # Data Sets: 358

Options

File Timetag: UTC MJD [✓ Relative MJD]

(Use Read with Relative MJD Timetag for Plot)

[✓] Scale Data [✓] Adjust Read Times for Data Overlap

Monitor Signals

Left Y-Axis Signal/Channel: Frequency

Freq Data: Browse

File: C:\Program Files\Dev

Right Y-Axis Signal/Channel: Light 1

Units: Volts DC

File: C:\Program Files\Dev

Select clock, then select measurement. Edit read times and plot variables. Choose output type, and press Read to extract monitor data from database.

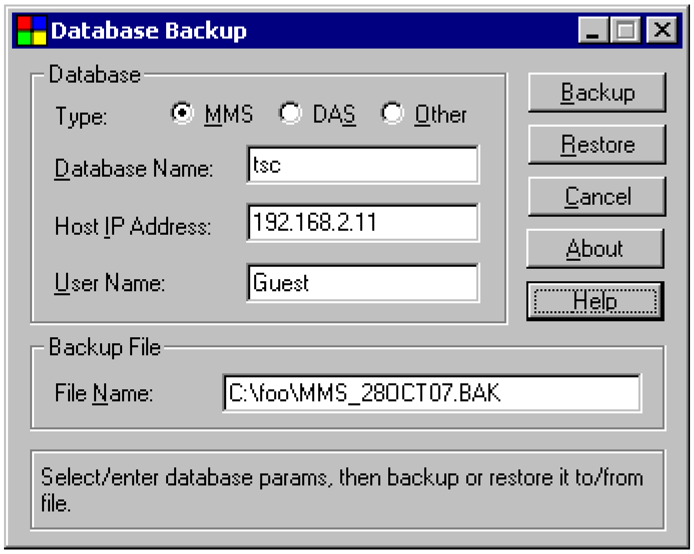
Read

Message

Freq Points =1000
Left plot data range=3.94e-13
Right Monitor Points =359
Right plot data range=3.72e-04 Volts DC

Figure 6. Clock and Monitor Database Interface Main Screen

Figure 7 shows a Windows interface for the `pg_dump` utility to back up a PostgreSQL database.



Database

Type: [✓ MMS] DAS Other

Database Name: tsc

Host IP Address: 192.168.2.11

User Name: Guest

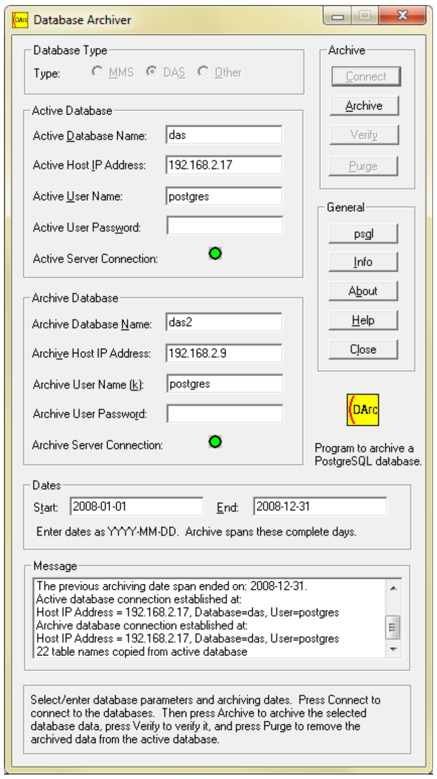
Backup File

File Name: C:\foo\MMS_28OCT07.BAK

Select/enter database params, then backup or restore it to/from file.

Figure 7. Database Backup Interface Screen

The application shown in Figure 8archives and then purges a portion of a PostgreSQL database so as to keep the original one at a more manageable size.



Database Type

Type: MMS [✓ DAS] Other

Active Database

Active Database Name: das

Active Host IP Address: 192.168.2.17

Active User Name: postgres

Active User Password:

Active Server Connection: ●

Archive Database

Archive Database Name: das2

Archive Host IP Address: 192.168.2.9

Archive User Name (k): postgres

Archive User Password:

Archive Server Connection: ●

Program to archive a PostgreSQL database.

Dates

Start: 2008-01-01 End: 2008-12-31

Enter dates as YYYY-MM-DD. Archive spans these complete days.

Message

```
The previous archiving date span ended on: 2008-12-31.
Active database connection established at:
Host IP Address = 192.168.2.17, Database=das, User=postgres
Archive database connection established at:
Host IP Address = 192.168.2.17, Database=das, User=postgres
22 table names copied from active database
```

Select/enter database parameters and archiving dates. Press Connect to connect to the databases. Then press Archive to archive the selected database data, press Verify to verify it, and press Purge to remove the archived data from the active database.

Figure 8. PostgreSQL Database Archiver Main Screen

• Database Skills

Clock database usage is largely made transparent by user interface software during ordinary operations, and thereby makes measurements and analysis easier by consolidating data storage and retrieval.

Clock database development, however, may require that local technologists (clock designers, test engineers, etc.) learn some new skills. The clock and monitor measurement systems will require user interfaces that incorporate an appropriate database server, schema, and control software. The analysis systems will require user interfaces to flexibly access the clock and monitor data. These software tools can either be bundled with a purchased measurement system, or developed by an in-house programmer or outside consultant. If custom software is required, it is critical that it be devised with database and clock analysis knowledge as well as operating system, networking and programming skills. In any case, because some recurring effort will be needed for administration, maintenance, updates, backups and troubleshooting, local IT personnel and/or test engineers will need some intermediate level of database proficiency. Because long-term usage of the system is likely, thorough documentation is critical, especially to support future software updates. Computer hardware and operating systems evolve quickly as a measuring system is designed, built, commissioned, receives updates and then finally requires legacy support as its database is used to find and retrieve old clock data.

• Database Cost

Although the core database and programming software tools may be practically free, and the necessary computer hardware cost modest, conceiving and commissioning a custom clock database system needs considerable test engineering and/or programming effort. Depending on the scope and complexity, that work can require several full time man months during initial implementation followed by documentation, training, updates and maintenance support. Those costs have to be weighed against its necessity and associated productivity gains over the life of the system.

• Conclusions

Inclusion of a database into a clock measurement and analysis system provides many benefits, especially for a multichannel system where many clocks are involved over a long period of time. These benefits are particularly evident when data must be extracted for selected clocks and references over particular dates and correlated with corresponding monitor and environmental readings and measurement notes. It is arguably the best way to share such data locally over a LAN or even widely via the internet where a database web server can make clock data available to any device with a web browser.

While it would be nice to establish a standard relational database type and schema for all clock data, that is probably not feasible given the many database choices, clock measurement systems and computer hosts. Nevertheless, because these elements share many common attributes, it is reasonable to assume that experience gained with one configuration can be applied to others. Knowledge about and use of database techniques is therefore recommended for all those who work with clock data. Not all clock data belongs in a formal relational database, but a strong case can be made for establishing one for any permanent multichannel clock measuring system.

• References

1. See the PostgreSQL web site.
2. N. Matthew and R. Stones, *Beginning Databases with PostgreSQL*, ISBN 1-59059-478-9, Apress, 2005.
3. W.J. Riley, "A PostgreSQL Database for the PicoPak Clock Measurement Module", Hamilton Technical Services, April 2015.
4. Data Sheet, Multi-Channel Measurement System, Microsemi, Inc., January 2016.
5. PHP web site.
6. PHPlot web site.
7. W.J. Riley, "A Web-Based PicoPak Monitor", Hamilton Technical Services, September 2016.
8. National Instruments Lab Windows/CVI.
9. W.J. Riley, "PicoPak Software Overview", Hamilton Technical Services, August 2016.

File: Databases for Clock Data.doc
W.J. Riley
Hamilton Technical Services
March 5, 2019