picoDB ™
a NoSQL database tool for eLua

Lua Workshop
Reston VA US
November 2012
Agenda

- Why picoDB™
- An overview of picoDB™
- Using picoDB™
- Performance characteristics
- Future plans
Why picoDB™

Host Processor

Signals

Grab → Convert

Messages

Decide → Act
Atmel SAM4S

- 160KB RAM
- 2 MB Flash
Why picoDB™

DECIDE

HISTORY

MODEL

ORGANIZE & STORE

- HISTORY
  - Sensor trends
  - Activation traces

- MODEL
  - Rule sets
  - Decision trees
An overview of picoDB™

multiple 1-table in-memory databases
(initially)
• dbSETUP – load all database information.
  • returns: status code

• dbCOMMIT – save all database information.
  • returns: status code

• dbDEFINE – add a database through its metadata.
  • returns: status code

• dbLOCATE – locate data tuples to a database subject to data attribute constraints.
  • returns: a list of matching tuples (empty if no match found or an error code)
• dbBUILD – add or change data tuples to a database subject to data attribute constraints (changes only).
  • returns: a status code

• dbDELETE – remove data tuples to a database subject to data attribute constraints.
  • returns: a status code

• dbERASE – remove both the metadata and data content of a database
  • returns: a status code

• dbSORT – provide a list data tuples of a database sorted by up to 2 data attributes.
  • returns: data tuple list sorted by the data attribute(s) or an error code
stat = dbDEFINE("Meas",{"ID","string", "measure","number"})

stat = dbDEFINE("Coeff",{"row","number", "column", "number", "setting","number"})

stat = dbBUILD("Meas","876",
    {"ID","a0", "measure",45.2})

stat = dbBUILD("Coeff","TempF",
    {"row",2,"column",3, "setting",0.58})

alst = dbSORT("Meas",{"meas","ID"})
• *dbMESSAGE* – format a message to a device or network based on a message exchange protocol.
  • returns: a hexadecimal string representing the message or an error code

• *dbVERIFY* – process a message received from a device or network based on a message processing sequence.
  • returns: a status code
Using picoDB™

- Metadata – Protocols database

```javascript
picoDB.dbDEFINE("Protocols",
    {
        "ProtocolID","string",
        "MsgID","string",
        "ParmID","string",
        "ParmType","string",
        "ParmRange","table",
        "ParmDefault", "string",
        "ParmLoc","number",
        "ParmSize", "number"
    })
```

- Used by dbMESSAGE to create device or network messages
• Metadata – Verifier database used by dbVERIFY

picoDB.dbDEFINE("Verifier",
   {"ProtocolID","string",
    "MsgID","string",
    "ParmID","string",
    "ParmProcess","table"})

• Used to process device or network messages
• ParmProcess uses a stack machine structure
  • example – convert Celsius to Fahrenheit
    • {"P_",9,"*",5,"/",32,"+","=R_Temp_Val"}
Example – using dbMESSAGE with dbVERIFY

```javascript
require "picoDB"
while true do
    msg = picoDB.dbMESSAGE("TempHum","RQHum",{1,"H"})
    if type(msg) ~= "string" then
        -- deal with error condition
    end
    -- request and retrieve data from a humidity sensor
    .......
    stat = picoDB.dbVERIFY("TempHum","RSHum",devresp)
    if stat ~= 0 then
        -- deal with error condition
    else
        -- perform analysis or forward info
    end
end
```

Tom Freund
Performance characteristics

Platform – Futurlec ET-STM32 Stamp

- MCU - ARM Cortex M3 (72 MHz, 90 MIPS)
- Internal RAM – 64 KB
- Internal Flash – 512 KB
- Dim (L X W X H) – 42 mm X 65 mm X 60 mm (1.7” X 2.6” X 2.4”)

Tom Freund
Performance characteristics

- Scenario – ongoing alpha testing
  - Sample temperature-humidity acquisition cycle
    - picoDB + chunk using dbVERIFY and dbMESSAGE
    - no I/O to sensors
    - Protocol and Verifier tables
  - build via eLua Builder – eLua site
    - binary (ROMable) image: 270KB
    - reference eLua footprint
      - Flash – 256KB
      - RAM – 64 KB
  - Preliminary results – 1 millisecond per cycle
Future plans

- Commercial
  - picoChain™ – 1Q 2013
  - development and deployment tools

- Community
  - Sourceforge – 2Q 2013
Questions ?

Lua Workshop 2012