EPICS PROGRAMMING WITH LUA
What is lua?

• Simple, procedural scripting language
• MIT licensed
• Built for embedding
  - Small footprint
  - Extremely portable
What’s in the lua module?

• luascript Record
  - Record type with scriptable behavior
• luash
  - IOC shell replacement
• Device support for common records
• Helper methods to embed lua into other modules
The luascript record

• Based off the scalcout record
• CALC field replaced with CODE field
  - Can either be inline statement
    ```lua
    field (CODE "return ‘Hello, World’")
    ```
  - Or can reference a function in a script
    ```lua
    field (CODE, "@script.lua function")
    ```
Using Inputs

- 10 double inputs (A-J) and 10 string inputs (AA-JJ)
- On record process, input values are converted to global variables of the same name

field (CODE "return A + B")

field (CODE "return AA .. BB")
The lua shell

• REPL using the lua interpreter
• Can be used as a replacement for or an addition to the IOC or vxWorks shell.
• Provides significant additions to make startup scripts more powerful.
Functions

• Functions can simplify complex tasks

```python
def logInfo(text):
    logfile = io.open("test.log", "a+")
    logfile:write(os.date("%c", os.time()))
    logfile:write(" - " .. text .. "\n")
    logfile:close()
end
```
Arithmetic

• Perform arithmetic to determine macro values

NUM_CHANS = 16
SAMPLES_PER_CHAN = 1200
TOTAL_SAMPLES = NUM_CHANS * SAMPLES_PER_CHAN
Conditionals

• Use conditionals to selectively control execution

```python
COMPUTER = os.getenv("HOSTNAME")
if (COMPUTER =~ "kobold.aps.anl.gov") then
    logInfo("IOC can’t be run on other computers")
    os.exit()
end
```
Flow Control

• Use loops to do repetitive tasks

```
IOC_INFO = {name = "ioc1", engineer = "klang", location = "437b004"}
for key, value in pairs(IOC_INFO) do
    print(key, value)
end
```
Usages

• Can be entered and exited on the shell command line

  iocxxx> luash

• Can be used to call a script

  iocxxx> luash "script.lua" "MACRO=value"

• Can be used as a full replacement for iocsh (soft IOC only)

  luash(argv[1])
‘asyn’ Library

• getXXXParam, setXXXParam
  • Read or write param values on an asyn port

• write/read/writeread
  • Read, write, or both to an asynOctet port

VAL = asyn.getIntegerParam("port", 0, "param")
asyn.setIntegerParam("port", 0, "param", VAL + 1)
asyn.callParamCallbacks("port", 0)

RESPONSE = asyn.writeread("IDN?", "port", 0)
print(RESPONSE)
‘epics’ Library

• **get/put**
  - Does the same as caput or caget

• **pv**
  - Convenience for the previous functions

• **sleep**
  - Causes the execution thread to sleep

```plaintext
if (epics.get("xxx:yyy:value") > 0) then
  epics.put("xxx:yyy:value2", 10)
end

PV1 = epics.pv("xxx:yyy:value")
PV2 = epics.pv("xxx:yyy:value2")
if (PV1["VAL"] > 0) then
  PV2["VAL"] = 10
end

for I = 1, 10 do
  print(I)
  epics.sleep(1.0)
end
```
‘iocsh’ Library

• Only available in 3.15.5 and above

• Takes any function name and arguments

• Searches for any IOC shell function with the same name

```python
VERSION = 0 + os.getenv("EPICS_VERSION_MAJOR")
REVISION = 0 + os.getenv("EPICS_VERSION_MIDDLE")
MINOR = 0 + os.getenv("EPICS_VERSION_MINOR")

VERSION_INT = VERSION << 16 | REVISION << 8 | MINOR

if (VERSION_INT >= (3 << 15 << 8 | 5)) then
    iocsh.dbLoadRecords("./advanced.db", "P=xxx:"
end
```
Adding Custom Libraries

- IOC’s that include the module can extend lua with custom functions
- All platforms support runtime loading of pure lua scripts
- Platforms that support dynamic libraries can load c/c++ libraries at runtime
- Using EPICS’ database registrar, you can build lua extensions into your IOC.
C/C++ Libraries

• A single exported function needs to be available
  - luaopen_xxx, where xxx is the same name as the library.

  - Function must take in a lua_State* and return an integer
    • return value is the number of values pushed onto the lua stack
    • Should only be 1

  - Uses luaL_newlib to give lua a table of function pointers and string names
Example

```c
static int bar (lua_State* state) {
    lua_pushstring(state, "Hello, World");
    return 1;
}

int luaopen_foo (lua_State* state) {
    static const luaL_Reg fooFuncs[] = {
        { "bar", bar },
        { NULL, NULL } /* Sentinel item */
    };
    luaL_newlib(state, fooFuncs);
    return 1;
}
```
Dynamic Libraries

- Set the environment variable LUA_CPATH
  - A set of templates with wildcards
  - Example: "./?.so;/usr/local/?/init.so"

```lua
foo = require("foo")
print(foo.bar())
```
Static Libraries

**foo.cpp**

```c
static void fooRegister(void) {
    luaRegisterLibrary("foo",
    luaopen_foo);
}
extern "C"
{
    epicsExportRegistrar(fooRegister);
}
```

**foo.dbd**

```lua
registrar(fooRegister)
```
More Info

• lua module can be found at: 
  https://github.com/epics-modules.lua
  - Contains example IOC’s and documentation for all functionality

• lua programming help can be found at: 
  https://www.lua.org/start.html