State Notation Language (SNL) Programming

Andrew Johnson, AES/SSG

Material from many people
IOC

- Database: Data Flow, mostly periodic processing
- Sequencer: State machine, mostly on-demand

Optional:
Sequencer can run as a standalone CA-Client
State Machines 101

- State Machine is in some State
- Events trigger transitions
- Actions are performed on transition
Example

Start

Low vacuum

pressure < 5.1 uTorr

Open the valve, update pumps, …

High vacuum

pressure > 4.9 uTorr

Close the valve, update pumps, …
Example State Notation Language

state low_vacuum
{
    when (pressure <= .0000049)
    {
        RoughPump = 0;
pvPut(RoughPump);
        CryoPump = 1;
pvPut(CryoPump);
        Valve = 1;
pvPut(Valve);
    } state high_vacuum
}
state high_vacuum
{
    ...
}
How it works

State Notation Language

program sncExample
double v;
assign v to "{user}:aiExample";
monitor v;
ss ss1
{
    state low
    {
        when (v > 5.0)
        {
            printf("sncExample: Changing to high\n");
        }
    }
    state high
    {
        when (v <= 5.0)
        {
            printf("sncExample: Changing to low\n");
        }
    }
}

C Code

```
Code for state "low" in state set "snc" /

/* Event function for state "low" in state set "snc" */

static void E_low(int64_t ssid, struct UserVar *pVar)
{
    // Line 15 "./sncExample.c"
}

/* Action function for state "low" in state set "snc" */

static void A_low(int64_t ssid, struct UserVar *pVar, char *transNum)
{
    switch (transNum)
    {
        case 0:
        // Line 14 "./sncExample.c"
            printf("sncExample: Changing to high\n");
    }
}
```

Pre-compiler

```
C Compiler

Object code
```

APS EPICS Training • Andrew Johnson • 2014-11-21 • SNL Programming
Advantage

- Compiled code. Fast.
- Can call any C(++) code
  - Use #define to create macros, ...
- Easy connection to Channel Access, Records
  - Compared to custom CA client, device support, ...
- Skeleton for event-driven State Machine
  - Handles threading, event handling, ...
When to use the sequencer

- For sequencing complex events
- E.g. park and unpark a telescope primary mirror

Photograph courtesy of the Gemini Telescopes project
Disadvantage

- Limited runtime debugging
  - See current state, values of variables, but no details of C code within actions

- Can call any C(++) code
  - and shoot yourself in the foot

- Pre-compiler.
  SNL error
  - SNC creates nonsense C code
  - Totally cryptic C compiler messages

- Risk of writing SNL code
  1. Starts out easy
  2. Evolves
  3. Ends up as a convoluted mess
Should I use the Sequencer?

**Good Reasons:**

- Start-up, shut-down, fault recovery, automated calibration
- Stateful Problem
  - My SNL has 20 states, 30 possible transitions, and little C code for each transition
- Cannot do this with CALC, BO.HIGH, SEQ, subroutine records

**Bad Reasons:**

- PID control, interlocks
- Warning sign:
  - My SNL code has 3 states with 2000 lines of C code
- I don’t want to deal with records, I’m more comfortable with C code
If you really want to use SNL

Good manual:

http://www-csr.bessy.de/control/SoftDist/sequencer/

Implement in small steps

- Code a little
- Compile, test
- Code a little more
- Compile, test
SNL Structure

```plaintext
program SomeName

/* Comments as in C */

/* Options */

/* Variables */

/* State Sets */
```

Program Name Used in DBD file and to start, stop and query it
SNL Options

option +r;

Make “re-entrant”.
Should be the default.
Allows running more than one copy (with different macros).

option -c;

Start right away, do not await connections.
Event with “+c”, the default, PVs may disconnect..
**Variables**

```cpp
double pressure;
assign pressure to "Tank1Coupler1PressureRB";
monitor pressure;
```

**Map to channel**

```cpp
short RoughPump;
assign RoughPump to "Tank1Coupler1RoughPump";
```

**Update with channel**

```cpp
string CurrentState;
assign CurrentState to "\{macro\}:VacuumState";
```

```cpp
string == char[40]
```

**Replaced w/ macro’s value**

*int, short, long, char, float, double*
Array Variables

double pressures[3];
assign pressures to
{
   "Tank1Coupler1PressureRB",
   "Tank1Coupler2PressureRB",
   "Tank1Coupler3PressureRB"
};
monitor pressures;

short waveform[512];
assign waveform to "SomeWaveformPV";
monitor waveform;
Event Flags

a) Communicate events between state sets

b) Trigger on Channel Access updates

Declare like this:

```
evflag event_flag_name;
```

Optionally, synchronize with monitored variable

```
sync var_name event_flag_name;
```
State Sets

ss coupler_control
{
    state initial{
        when (pressure > .0000051){
            } state low_vacuum
        when (pressure <= .0000049){
            } state high_vacuum
    }
    state high_vacuum{
        when (pressure > .0000051){
            } state low_vacuum
    }
    state low_vacuum{
        when (pressure <= .0000049){
            } state high_vacuum
        when (delay(600.0)){
            } state fault
    }
    state fault {
    }
}
Events

- Variables used in events should be ‘monitor’ed!

```c
when (pressure > .0000051)
{
    /* Actions ... */
} state low_vacuum

when (pressure < 0.000051 && whatever > 7)
{
} state high_vacuum

- Note that a delay(10) expression is not a wait for 10 seconds.
  - It means return false (0) until the state set has been in this state for 10 seconds, then return true (1).

```c
when (delay(10.0))
{
} state timeout
```
Events..

Use event Flags:

```c
when (efTestAndClear(some_event_flag))
when (efTest(some_event_flag))

/* Meanwhile, in other state */
when (pressure < 0.000051 && whatever > 7)
{
    efSet(some_event_flag);
} state high_vacuum
```

Check for connections:
```c
when (pvConnectCount() < pvChannelCount())
when (pvConnected(some_variable))
```
Actions

when (pressure > .0000051)
{
    /* Set variable, then write to associated PV */
    RoughPump = 1;
    pvPut(RoughPump);

    /* Can call most other C code */
    printf("Set pump to %d\n", RoughPump);
}

state low_vacuum

Action statements are almost C code. Above, RoughPump is a state machine variable. The SNL is transformed to
printf("Set pump to %d\n", pVar->RoughPump);

SNC will add the “pVar->” to all state machine variables that it recognizes.

Sometimes it will be necessary to
%
/* Escape C code so that it’s not transformed */
static void some_method_that_I_like_to_define(double x);
%

Walk through the SNL from
makeBaseApp -t example

• configure/RELEASE
  SNCSEQ=/path/to/seq

• Generated Makefile:
  .._SRCS += MySource.st

• DBD file entry
  registrar(SomeNameRegistrar)

• IOC st.cmd
  seq SomeName, “macro=value”
Sequencer Commands

- `seq name`
  - Start sequence program

- `seqShow`
  - List all sequence programs with their program `name`

- `seqShow name`
  - Detail of program state

- `seqChanShow name`
  - List variables of `seq`

- `seqStop name`
  - Stop a sequence
There is more

- Support for ‘entry’ and ‘exit’ blocks
- Assign PV names within code: pvAssign(..)
- ‘Get Callback’, ‘Put Callback’
- Checking status & severity of PVs
- ‘syncQ’ to queue received Channel Access updates
Summary

- SNL very useful for State-Machine logic
- Read the SNL manual