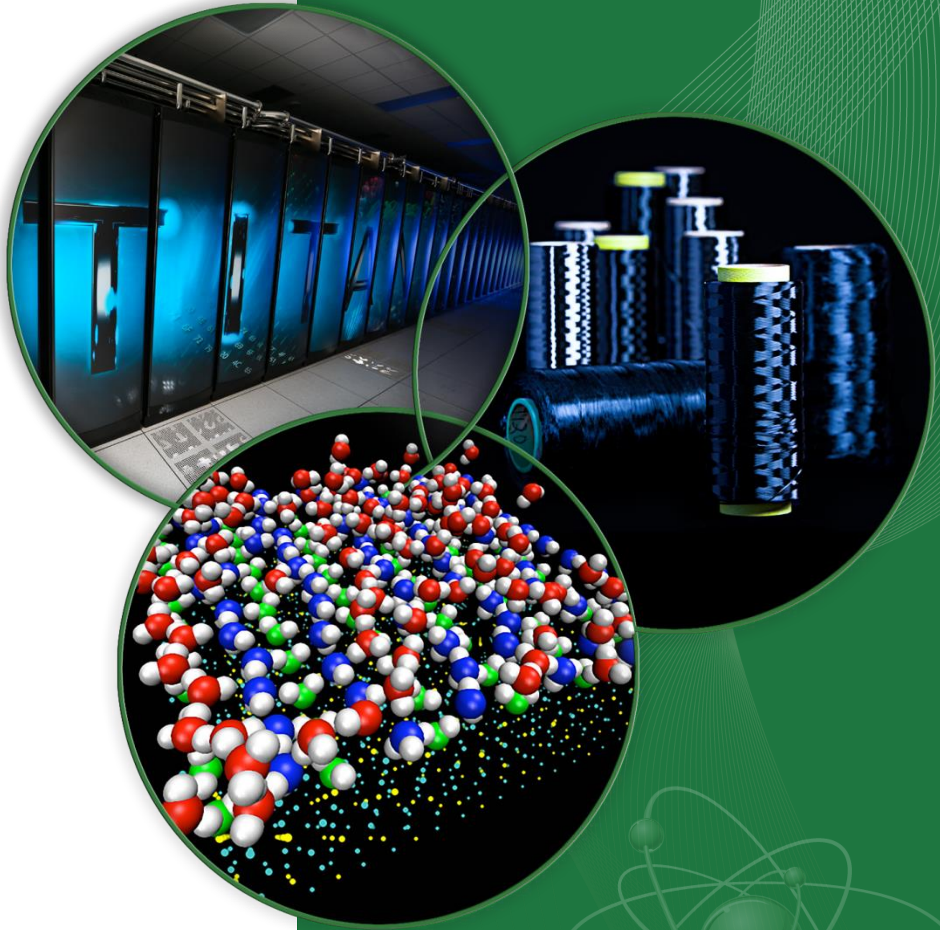
EPICS
State Notation Language (SNL), "Sequencer"



Kay Kasemir,

SNS/ORNL

Many slides from
Andrew Johnson,
APS/ANL

Feb 2019

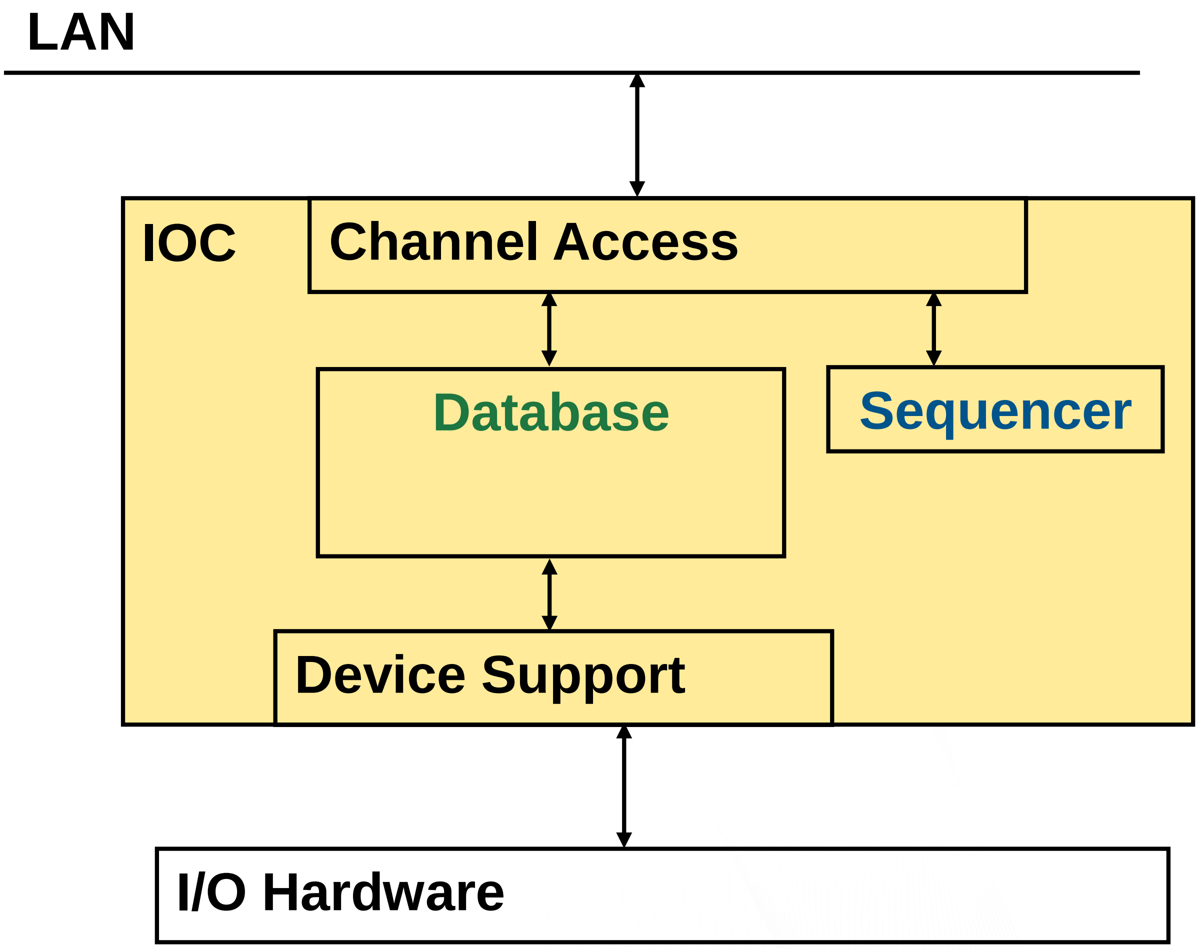
EPICS Sequencer

- Implementation of the state transition control model
- Transparently supports channel access conneciton to external data
  - Read connection state of PVs
  - Get and put values
  - Monitor value changes
- SNL – State Notation Language
  - Produces compiled code
  - Generates C code and supports insertion of manually crafted blocks of code
    - ```
      %%        strcpy( seqg_var->stateName, "init" );
      %{
                // multiple lines of c or c++ code
      }%
      ```

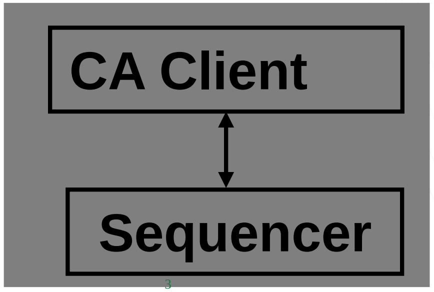
OAK RIDGE National Laboratory
      ```

IOC

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

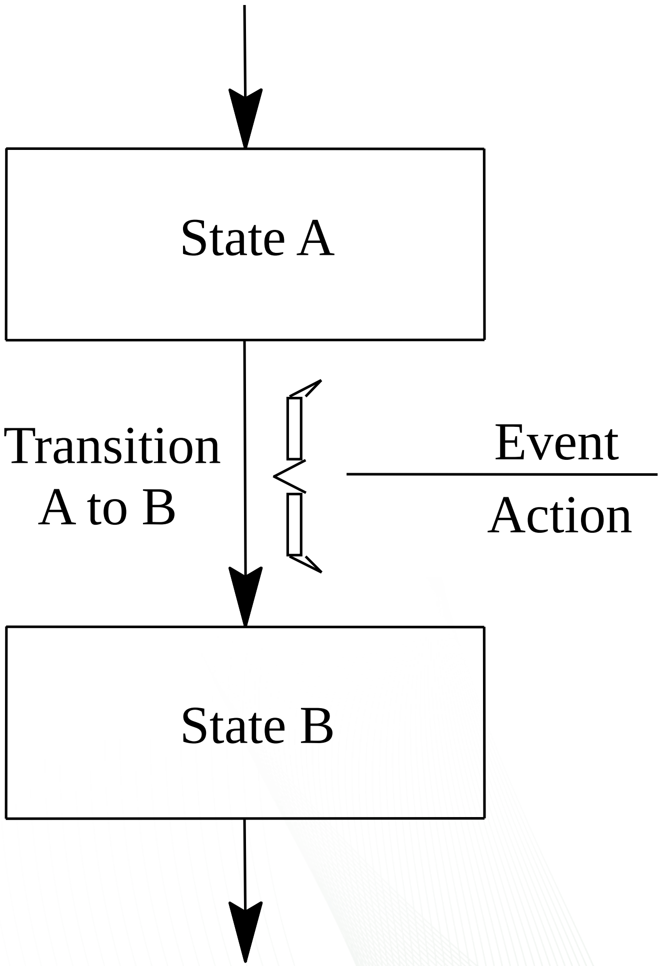


Optional:
Sequencer runs as standalone CA-Client

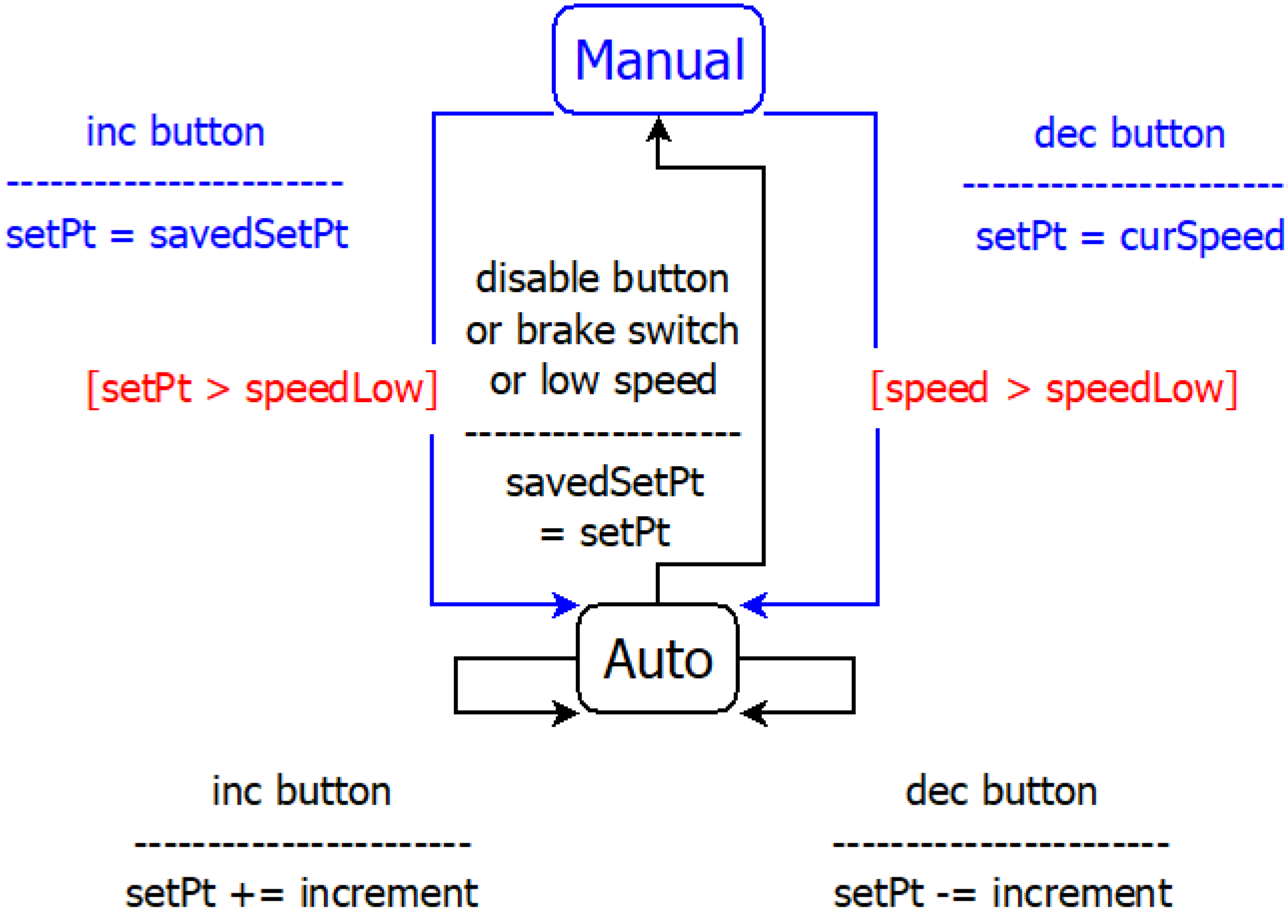


State Machine 101

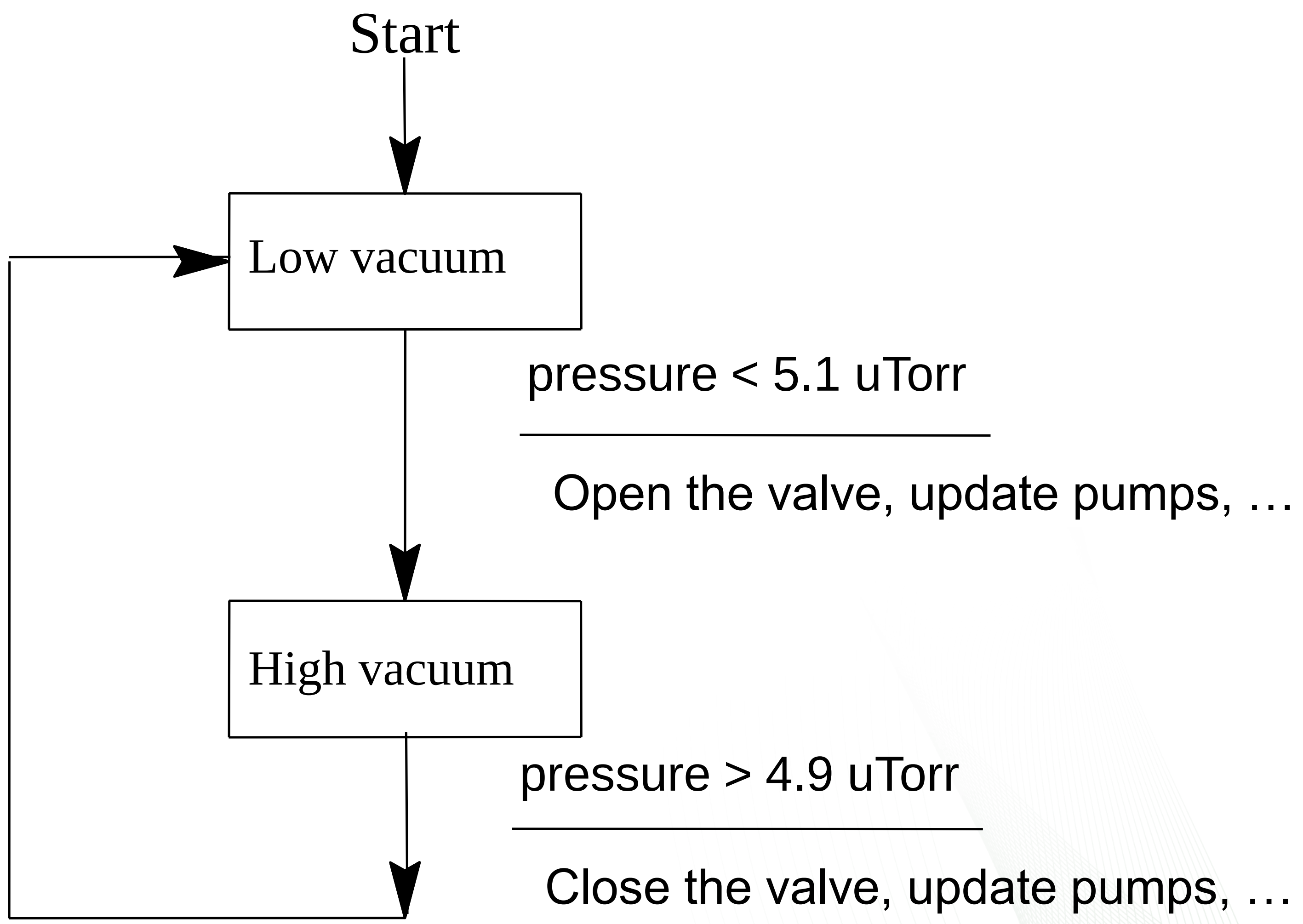
- System is in some state
- Events trigger transitions to other states
- Actions may be performed on transition



Example


Manual
inc button
----
setPt = savedSetPt
dec button
----
setPt = curSpeed
disable button
or brake switch
or low speed
----
savedSetPt
= setPt
[setPt > speedLow]
[speed > speedLow]
Auto
inc button
----
setPt += increment
dec button
----
setPt -= increment

Example



Start
Low vacuum
pressure < 5.1 uTorr
Open the valve, update pumps, …
High vacuum
pressure > 4.9 uTorr
Close the valve, update pumps, …

6

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
{
    …
```

State

Event

Action

Transition

OAK RIDGE National Laboratory

7

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
    }
    state high
    {
        when (v <= 5.0)
        {
            printf("sncExample: Changing to low\n");
        } state low
    }
}
```

"snc" Pre-compiler

C Code

```
/* Code for state "low" in state set "ss1" */

/* Delay function for state "low" in state set "ss1" */
static void D_ss1_low(SS_ID ssId, struct UserVar *pVar)
{
# line 15 "../sncExample.stt"
}

/* Event function for state "low" in state set "ss1" */
static long E_ss1_low(SS_ID ssId, struct UserVar *pVar, short *pTransNum, short *pNextState)
{
# line 15 "../sncExample.stt"
        if ((pVar->v) > 5.0)
        {
                *pNextState = 2;
                *pTransNum = 0;
                return TRUE;
        }
        return FALSE;
}

/* Action function for state "low" in state set "ss1" */
static void A_ss1_low(SS_ID ssId, struct UserVar *pVar, short transNum)
{
        switch(transNum)
        {
        case 0:
                {
# line 14 "../sncExample.stt"
                        printf("sncExample: Changing to high\n");
```

C Compiler

Object code

```
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000020 0001 003e 0001 0000 0000 0000 0000 0000
0000040 0000 0000 0000 0000 1738 0000 0000 0000
0000060 0000 0000 0040 0000 0000 0040 001d 001a
0000100 4855 e589 8948 f87d 8948 f075 c3c9 4855
0000120 e589 8348 10ec 8948 f87d 8948 f075 0ff2
0000140 0510 0000 0000 8b48 f845 00be 0000 4800
0000160 c789 00e8 0000 c900 55c3 8948 48e5 ec83
0000200 4820 7d89 48f8 7589 48f0 5589 48e8 4d89
0000220 48e0 458b bef8 0000 0000 8948 e8c7 0000
0000240 0000 8548 74c0 4819 458b 66e0 00c7 0001
0000260 8b48 e845 c766 0000 b800 0001 0000 05eb
0000300 00b8 0000 c900 55c3 8948 48e5 ec83 4820
0000320 7d89 48f8 7589 89f0 66d0 4589 0fec 45bf
0000340 85ec 75c0 480d 3d8d 0000 0000 00e8 0000
0000360 9000 c3c9 4855 e589 8948 f87d 8948 f075
```

OAK RIDGE National Laboratory

Advantage

- **Compiled code. Fast.**
- **Can call any C(++) code**
- **Easy connection to Channel Access and thus Records**
  - **Compared to custom CA client, device support, …**
- **Skeleton for event-driven State Machine**
  - **Handles threading, event handling, …**

Disadvantage

- **Limited runtime debugging**
  - See current state, values of variables, but not details of C code within actions
- **Can call any C(++) code**
  - and shoot yourself in the foot
- **Pre-compiler.**
  SNL error
  → SNC creates unreadable 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:

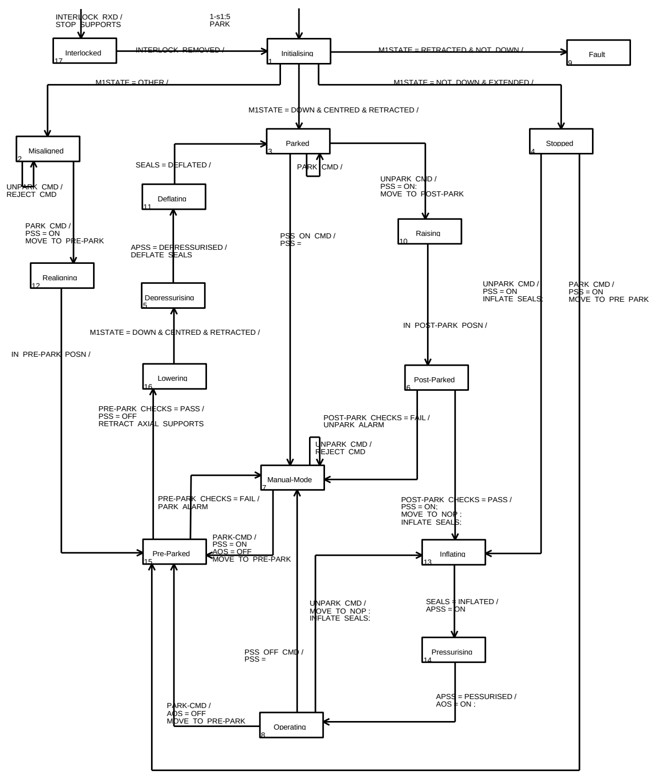
- **Start-up, shut-down, fault recovery, automated calibration**
- **Stateful Problem**
  - My SNL has 20 states, 30 possible transitions,, and little C code for each transition
- **Cannot satisfy system requirements with records**
  - CALC
    CALCOUT
    BO (momentary)
    SEQ
    Subroutine records
- **State machine purpose is to separate control flow and data flow**

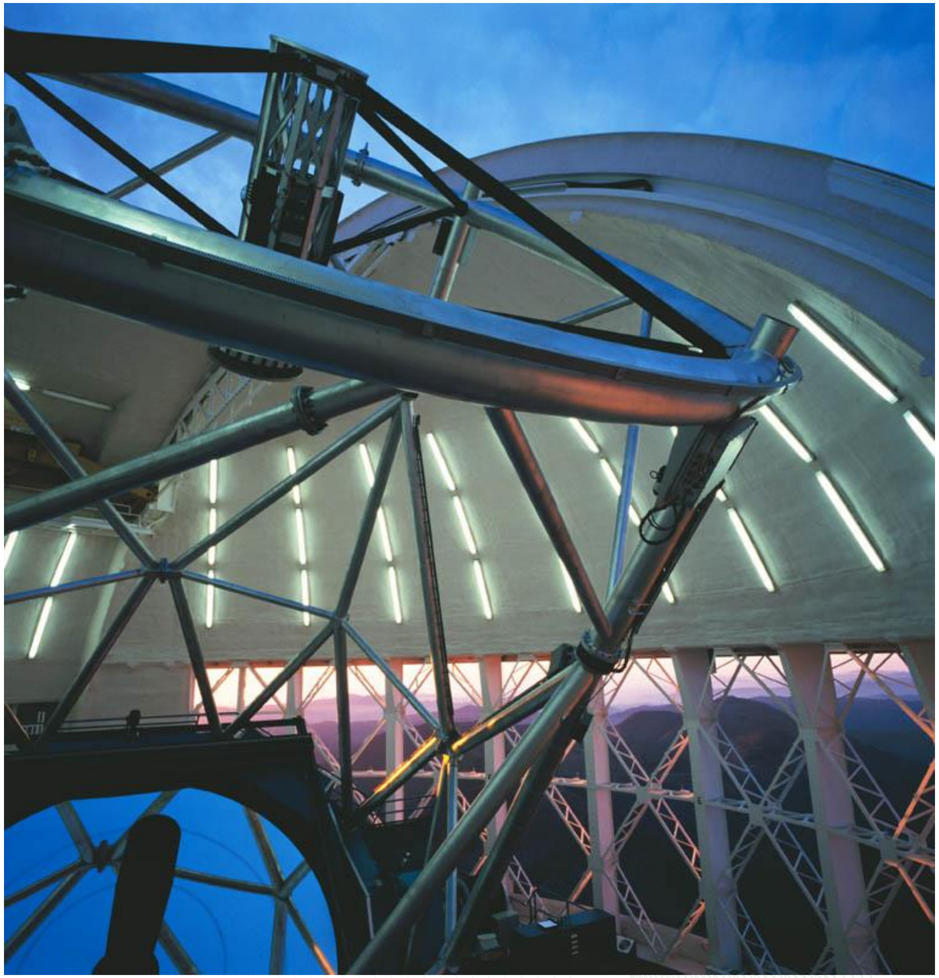
Bad Reasons:

- **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**

Use the sequencer

- For sequencing complex control tasks
- E.g. parking and unparking a telescope mirror





Photograph courtesy of the Gemini Telescopes project

OAK RIDGE
National Laboratory

12

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

– This makes debugging viable

- Bisect new code into successively smaller sections to find offending statements when diagnostic messages are overly mysterious

SNL Structure

Program name!

Used in DBD
&
to launch the sequence.

```
program SomeName("macro=value")

/* Comments as in C */

/* Options */

/* Variables */

/* State Sets */
```

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.

Even with "+c", the default, PVs may disconnect once you're running..

SNL Structure

```
program SomeName("macro=value")
/* Comments as in C */
/* Options */
/* Variables */
/* State Sets */
```

Variables

int, short, long, char, float, double

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

Map to channel

Update with channel

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

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

string == char[40]

Replaced w/macro’s value

Array Variables

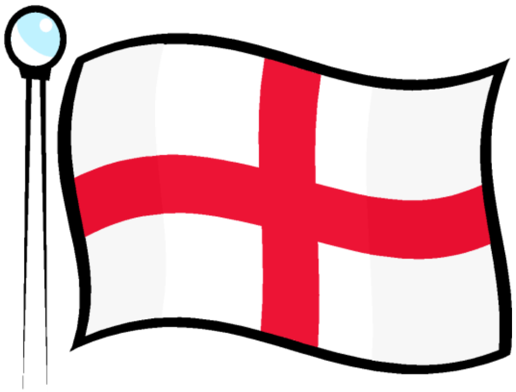
Any but 'string'

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

short waveform[512];
assign waveform to "SomeWaveformPV";
monitor waveform;
```

Map to channel(s!)

Event Flags

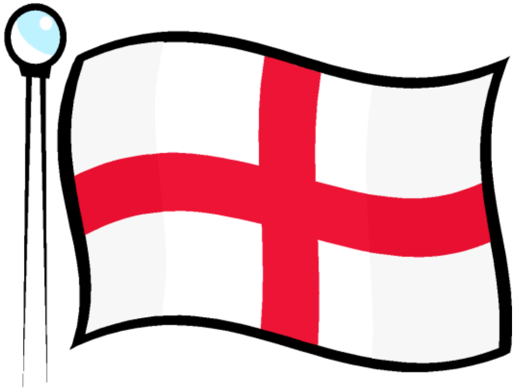


- **Declaration:**
  `evflag` *`event_flag_name`*`;`
- **Trigger on Channel Access updates by synchronizing with <u>monitored</u> variable**
  `sync` *`var_name event_flag_name`*`;`

```
assign var1 "pvname1";
monitor var1;
assign var2 "pvname2"
monitor var2;
sync var1 ef;
sync var2 ef;
```

- **Communicate events between state sets with**
  `efSet(), efTestAndClear(), ef*..`

Event Flags



Multiple PVs may be sync'd with a single evflag but a single PV may not be sync'd with more than one evflag

- **Allowed**
  - ```
    sync var1 ef1;
    sync var2 ef1;
    sync var3 ef2;
    ```
- **Not allowed**
  - ```
    sync var1 ef1;
    sync var2 ef1;
    sync var3 ef2;
    sync var1 ef2;
    ```
    # offending statement – attempt to # sync var1 with ef1 and ef2

OAK RIDGE National Laboratory

SNL Structure

```
program SomeName("macro=value")
/* Comments as in C */
/* Options */
/* Variables */
/* State Sets */
```

State Sets

Starts in First state, name does not matter

```
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

- **Variable value test**
  - Variables assigned to PVs and used in events MUST be monitor'ed if their values are changed by external agents alone

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

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

- **Asynchronous pvGet or pvPut completion**

```
when ( pvGetComplete(someVar) ) { …
when ( pvPutComplete(someVar) ) { …
```

Events..

- **Timer expiration**

```
when (delay(10.0))
{
} state timeout
```

This is not an unconditional delay!
It is a timeout that expires only when
other event conditions stay false for
the specified elapsed time

- **Event flags**

```
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
```

- **Connection state changes**

```
when (pvConnectCount() < pvChannelCount())
when (!pvConnected(some_variable))
```

Actions and Transitions

```
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 mostly resemble C code. Above, RoughPump is a state machine variable. The SNL for the printf is pre-compiled into

```
       printf("Set pump to %d\n", pVar->RoughPump);
```

SNC adds *pVar->* to all state machine variables.

Sometimes inserting manually crafted code blocks is necessary

```
%{
 /* Escape C code so that it’s not transformed */
 static void some_method_that_I_need_to_define(double x);
}%
```

Walk through the SNL from makeBaseApp –t example

configure/RELEASE or RELEASE.local

```
MODULES = /home/training/epics-train/tools
SNCSEQ = $(MODULES)/seq-2.2.6
```

Generated Makefile:

```
.._SRCS += sncProgram.st
```

sncExample.dbd

```
registrar(sncExampleRegistrar)
```

IOC st.cmd

```
seq sncExample, “user=me”
```

```
program sncExample
double v;
assign v to "{user}:aiExample";
monitor v;

ss ss1 {
    state init {
      when (delay(10)) {
          printf("sncExample: Startup delay over\n");
      } state low
    }
    state low {
      when (v > 5.0) {
          printf("sncExample: Changing to high\n");
      } state high
    }
    state high {
      when (v <= 5.0) {
          printf("sncExample: Changing to low\n");
      } state low
    }
}
```

Sequencer Management and Diagnostic Commands

- `seq NameOfSequence`
  - Start sequence
- `seqStop <thread id or name>`
  - Stop a sequence
- `seqShow`
  - List all sequences with IDs and names
- `seqShow <thread id or name>`
  - More detail for given thread
- `seqChanShow <thread id or name>`
  - List variables of seq

Sequencer Management and Diagnostic Commands…

- `seqcar <level>`
  - Level 0 - show pv statuses
    - Total programs=1, channels=18, connected=17, disconnected=1
  - Level 1 – show disconnected pvs per program
    - Program "sncExample“
 Variable "highLev" not connected to PV "one:highLevel“
Total programs=1, channels=18, connected=17, disconnected=1
  - Level 2 – show details for each pv by name
    - Program "sncExample“
Variable "systemEnable" connected to PV "one:systemEnable“
Variable "pause" connected to PV "one:pause“
Variable "fillTimeout" connected to PV "one:fillTimeout“
.
.
.

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**
- **and more…**

Summary

- **SNL and the EPICS sequencer is a powerful tool with a rich feature set**
- **Very easy to implement EPICS state machines with SNL**
- **Read the SNL manual**