An Automated Vacuum System*

W. H. Atkins**, G. D. Vaughn, C. Bridgman***
MS-H820, Los Alamos National Laboratory, Los Alamos, NM 87545

Abstract

Software tools available with the Ground Test Accelerator (GTA) control system provide the capability to express a control problem as a finite state machine. System states and transitions are expressed in terms of accelerator parameters and actions are taken based on state transitions. This is particularly useful for sequencing operations which are modal in nature or are unwieldy when implemented with conventional programming. State diagrams are automatically translated into code which is executed by the control system. These tools have been applied to the vacuum system for the GTA accelerator to implement automatic sequencing of operations. With a single request, the operator may initiate a complete pump-down sequence. He can monitor the progress and is notified if an anomaly occurs requiring intervention. The operator is not required to have detailed knowledge of the vacuum system and is protected from taking inappropriate actions.

I. INTRODUCTION

The approach of the GTA control system (EPICS, for Experimental Physics and Industrial Control System) is to provide a collection of tools which can be used to implement computerized controls. These tools can be utilized without intimate familiarity of the system. Reducing the programming expertise required of the implementer and the amount of code required increases the system's reliability and reduces its long-term cost.

II. THE SEQUENCER

One of the EPICS software tools, the *Sequencer*, allows a controls problem to be expressed in an accelerator-user's terms and nearly automatically produces programming code to implement the algorithm. The Sequencer tool is comprised of a *State Notation Language* (SNL) compiler which uses a Finite State Machine paradigm, and run-time code to assist in implementation of the program.

A state diagram is often more useful for expressing control problems, as the description is in terms of the problem and is not constrained by programming-language conventions. An example state diagram is shown in Figure 1.

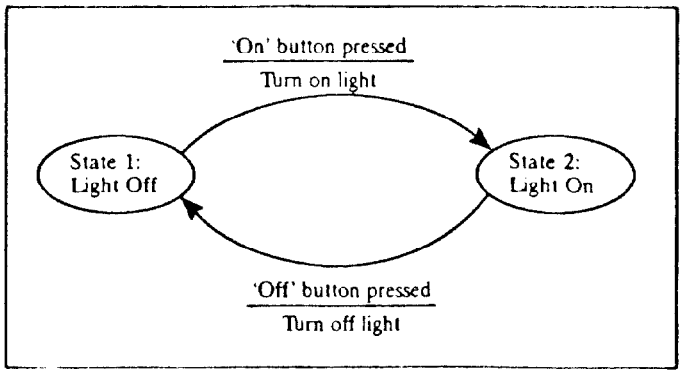

Figure 1. Example State Diagram.

In this example there are two states defined, corresponding to two states of a light. The connecting lines show that the 'system' can transition from one state to the other. The notation next to the state-transition lines describes (above the line) the condition for the transition occurring, and (below the line) actions that occur as a result of the transition. In the example, the condition for transitioning from state 1 to state 2 is that the 'On' button is pressed. The condition for the opposite transition is that the 'Off' button is pressed. No action is taken if the system is in state 2 and the 'On' button is pressed, as there is no transition describing that case.

A state diagram translates directly (not yet automatically) to a State Notation Language program. Figure 2 shows an excerpt of the corresponding SNL program for the example in figure 1.

```
state LIGHT_OFF
        {
        when (on_button_pressed = TRUE)
                {
                turn on light
                } state LIGHT_ON
        }

state LIGHT_ON
        {
        when (off_button_pressed = TRUE)
                {
                turn off light
                } state LIGHT_OFF
        }
```

Figure 2. Excerpt from State Notation Language Program.

The statements in italics are not part of the program syntax. The outer-most sets of braces delimit states. Within a state, the transitions are described as *when* clauses, read *"when some condition occurs, take some action(s), and transition to another state."* The condition is specified as the argument to the *when* statement. The actions to take when

*Work supported and funded by the United States Department of Defense, Army Strategic Defense Command, under the auspices of the United States Department of Energy.

** Oak Ridge National Laboratory

*** Grumman Corporation, Space Sciences Division.

0-7803-0135-8/91$01.00 ©IEEE

the condition is true are specified inside the next set of braces. Following these braces the destination state is specified. More than one *when* clause would be present in a state if there were more than one transition out of the state.

The expressions shown in the *when* statement are truly as simple as shown. Accelerator parameters such as temperatures, pressures, etc. are made available to the SNL program as program variables. SNL facilities provide this interface with other parts of the control system.

Without detailing the SNL syntax any further, it is clear that a SNL program can be directly produced from a state diagram with no logic or algorithmic translation. This makes for a quick transition from problem definition to implementation.

III. A Vacuum System Example

The vacuum system for the GTA Radio-Frequency Quadrupole (RFQ) utilizes two cryogenic pumps and one turbomolecular pump. Figure 3 shows how one cryo system is configured.

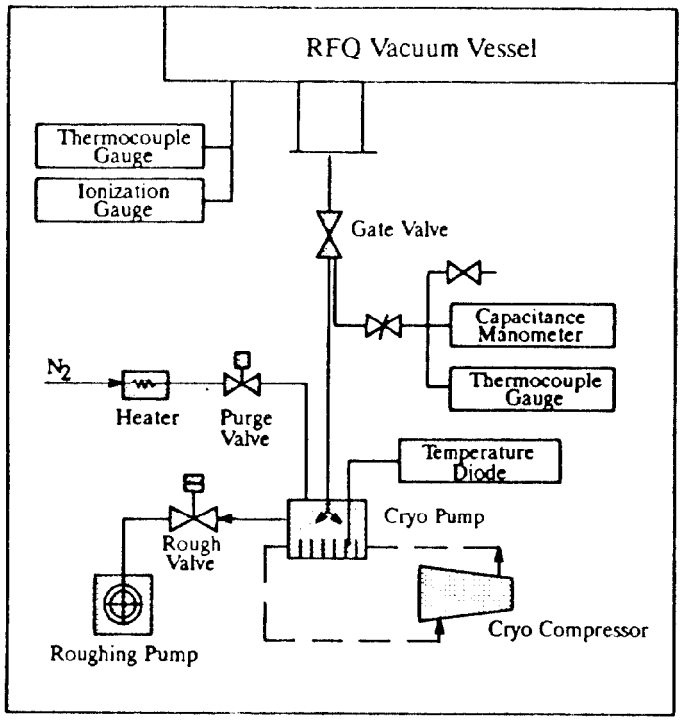

Figure 3. Partial RFQ Vacuum System Schematic.

Each pump and its associated instrumentation can be treated somewhat independently, i.e. a cryo pump can be roughed-out and cooled down independently of what the turbo pump is doing. Coordination of the three pump systems is required when they must cooperate in pumping the vessel itself.

A procedure for preparing the cryo pump for on-line pumping would be as follows:

1. Close gate valve, close purge valve, open rough valve.
2. Start roughing pump.
3. When cryo pump pressure reaches 50 millitorr, close rough valve, start compressor and cold head.
4. If pressure increases to 65 millitorr in less than 1 minute, open rough valve until pressure decreases to less than 50 millitorr, then close rough valve.
5. When cryo pump temperature is below 20°K, the pump is ready to be placed on-line.

Figure 4 shows this procedure expressed as a state diagram.

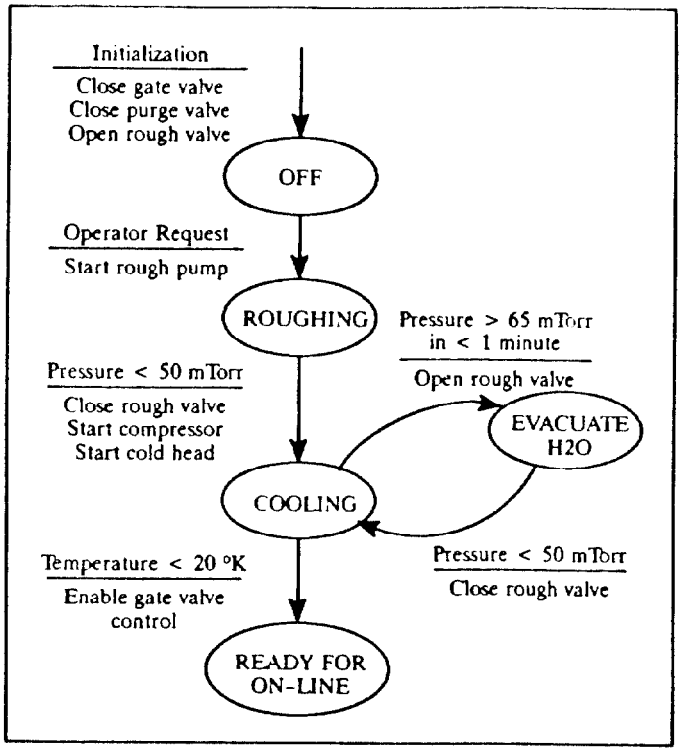

Figure 4. Cryo Procedure as a State Diagram.

The procedure is broken into 5 states. At initialization the system is put into a known state. On an operator request, the system begins evacuating the cryo pump with the roughing pump. This is the ROUGHING state. Note that this means roughing as pertains to the cryo pump, not the RFQ vessel.

Once the pressure in the cryo pump is less than 50 millitorr, the pump is ready to be cooled. The roughing valve is closed, and the cryo compressor and cold head are switched on (state COOLING).

During the first minute of cooling, if water vapor has not been adequately evaporated, it will condense and cause a pressure rise. If this is detected, the roughing valve is opened and the system transitions to state EVACUATE H2O. When the pressure is again low enough, the roughing valve is closed and a transition is made back to state COOLING. Note how transitions in and out of this state implement a hysteresis feature.

Cooling proceeds until the cryo pump temperature is below 20 °K. When this occurs, the state diagram calls for a transition to state READY FOR ON-LINE. The action corresponding to this transition is to enable control of the gate valve. It is not opened automatically because control must be coordinated with the other pumping systems.

This state diagram is not complete. It does not show paths for system shutdown or error handling. Error paths are needed to accommodate problems such as devices not operating as expected, e.g. the roughing valve not opening. Errors that can be anticipated can be handled automatically. Others require operator intervention.

The state diagram is easier to understand because much of the information is visual. It is quick to see what conditions cause state changes without reading through text or program code. A logic error is more likely to be realized through the state diagram than by reading program code.

This state diagram can be converted directly to SNL as described previously. The SNL provides an interface with other EPICS facilities so that conditions and actions specified by the sequence can be monitored and controlled easily. For example, C-language-like subroutine calls are used to test the cryo pump temperature or to operate a valve.

IV. Cooperative Sequences

It was noted that this cryo pump sequence is not completely independent of the rest of the RFQ vacuum system. To treat the RFQ vacuum system as a single entity, a higher-level sequence is appropriate, one specified in terms of the system, not in terms of individual components. Figure 5 shows an example.

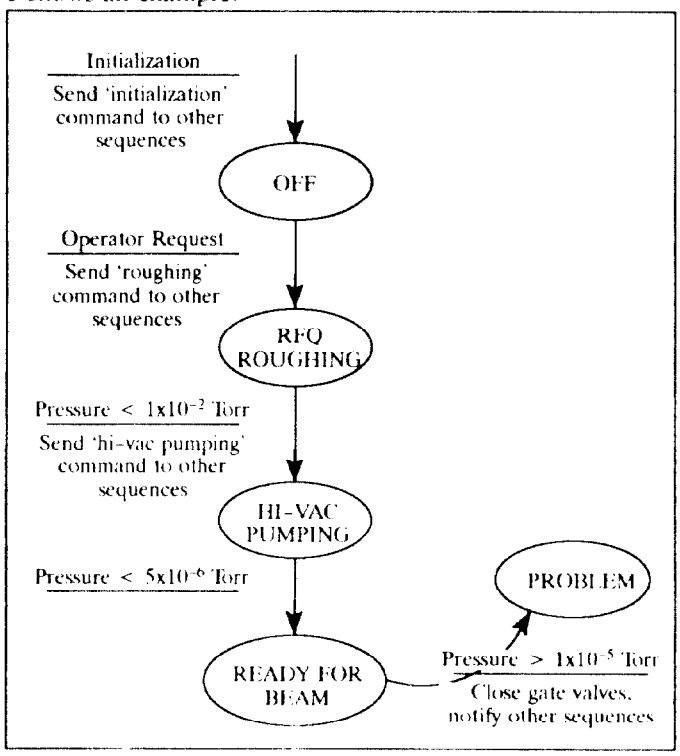

Figure 5. High-level Vacuum System State Diagram.

This diagram is expressed in terms of the RFQ vacuum system, hence 'roughing' means roughing of the RFQ vessel, not one of the pumps. Again, this is not a complete diagram as there is no shutdown path or error paths.

Note the PROBLEM state, which illustrates how an interlock may be implemented as part of the state machine. This interlock is more easily implemented here because the condition of pressure exceeding $1x10^{-5}$ Torr is only a problem if the system was previously READY FOR BEAM.

Actions specified by this diagram are instructions to lower-level sequences, not operations on particular vacuum components. In turn, the lower-level sequences would accept instructions from this sequence instead of directly from an operator. This hierarchy is illustrated in figure 6.

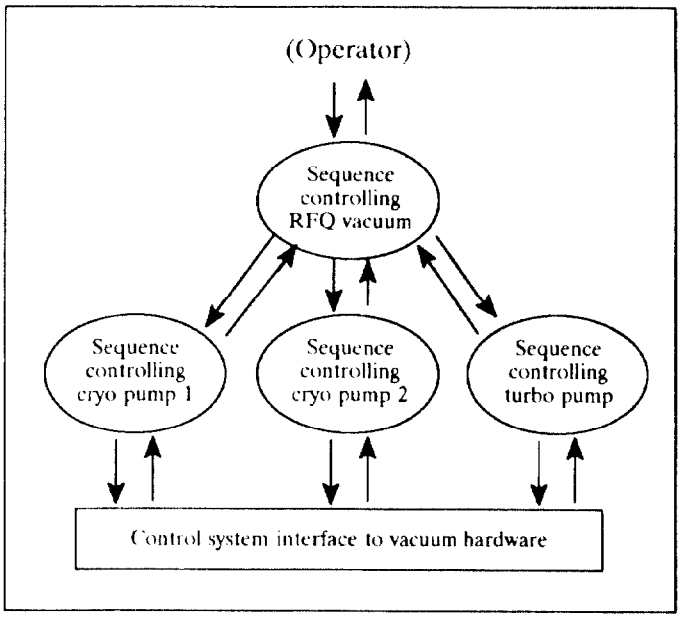

Figure 6. Hierarchical Sequences.

A third-level sequence might later be appropriate to coordinate RFQ vacuum with other upstream and downstream vacuum vessels.

V. Summary

Some controls problems lend themselves better to a state machine representation than to other paradigms. For such problems, the EPICS Sequencer tool provides a convenient and faster implementation method. Hierarchical sequences may be built which provides a basis for higher-level accelerator control.

VI. References

[1] A.J. Kozubal, D.M. Kerstiens, J.O. Hill, and L.R. Dalesio, "Run-time Environment and Applications Tools for the Ground Test Accelerator Control System", Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems, Vancouver, BC, Canada, Oct 30 - Nov 3, 1989.