The DELPHI experiment control.


1) CERN, CH-1211 Geneva 23, Switzerland.
2) Rutherford Appleton Laboratory, Chilton, GB-Didcot OX11 OQX, UK.
3) Nuclear Physics Laboratory, University of Oxford, Keble Road, GB- Oxford OX1 3RH, UK.

The DELPHI detector, which is in operation since the start of LEP in August 1989, consists of 16 different sub-detectors. To provide a high degree of independence to the individual sub-systems, the data acquisition and slow control system has been split into autonomous partitions. The systems which controls the experiment and the subdetector partitions is described.

Introduction

The DELPHI detector [1], a general purpose detector with special emphasis on particle identification, is composed of 16 different sub-detectors, which were built by different teams of laboratories of the DELPHI collaboration. To provide a high degree of independence to the individual sub-detectors, the data acquisition and control of the experiment is split into a set of autonomous partitions. Although each detector has its own specific characteristics and function, the organization of these partitions has been standardized up to a high degree. This standardization is extended towards the central readout and trigger.

The control of all devices associated with each partition, is organized into so called control domains. There are separate control domains to handle the two aspects of each sub-detector: a) Data-acquisition : i.e. the fastbus embedded readout system [3], and the data flow handling in the equipment computers [4], and b) Slow-control: i.e. the monitor and control system for technical aspects of the sub-detectors [2] such as gas, volts, pressure, temperature, ... . The DELPHI control systems consists of 18 domains for data acquisition, 17 domains for slow control and a few central control domains. The control domains and their associated driver processes are distributed over 20 different nodes in the DELPHI online VAX cluster.

The State Manager Concept

The DELPHI experiment control system is characterized by a highly decentralized organization: each embedded processor, the equipment computers, and all the tasks running in these computers have a large extent of autonomy. This organization makes the individual components very flexible and maintainable due to their independence. On the other hand it however puts additional strains on the control system. To cope with the complexity of the control a new concept for the coding of the control logic was developed[5][6], with the following design requirements in mind: The system should 1) be able to deal with the distributed nature of the 'driver processes' (i.e. the processes controlling the external devices); 2) allow for easy modification of the control logic without interference with unrelated driver processes; 3) operate independently from the driver processes and from any interactive run control process. (More specifically, there should...
be no logic built into any of these processes); 4) be able to take automatic actions upon changing conditions in the experiment, independent of any operator or interactive run control interface process; and 5) provide concurrent external access to a control domain to allow central control domains and a local user interface to access concurrently a local domain.

The approach we adopted is based on the State Manager concept. In this concept, the experiment is described in terms of objects, i.e. logical subsystems, for each of which a number of states are defined. An object may correspond directly to a concrete entity in the experiment (a computer controlled device) or any abstraction used in describing the experiment provided it can be identified by a 'noun' (e.g. 'run', 'trigger', 'central detector', etc.). The control system, which is given by the interaction between the various objects, is specified using a formal language, the State Manager Language (SML). The main characteristic of this language are:

- **Finite state logic.** Objects behave as finite state machines. The only 'variables' in this language are the states of the objects (e.g. RUNNING,_PAUSED, ...). An action applied on an object can bring about a change in its state.

- **Instruction sequencing.** Actions on an abstract object may specify a sequence of instructions, mainly consisting of actions and logical tests on other objects. Actions on concrete objects, are send off as messages to an associated driver process which controls the external device.

- **Asynchronous execution.** Several actions may proceed in parallel: an action applied by object-A on object-B, does not suspend the instruction sequence of object-A. Only a test by object-A on the state of object-B suspends the instruction sequence of object-A, if object-B is still in transition.

- **AI like rules.** Each object can specify logical conditions based on the state of other objects, which, when satisfied, will trigger an action of the local object. This provides the mechanism for an object to respond to unsolicited state changes of its environment.

The logic, specified in the State Manager language is translated by a special compiler to create the State manager. The State Manager runs as an independent process and communicates with its environment using DECNET. An interactive control program can be used to control the state manager. This program has access to the specification and states of the objects and may uses this information to dynamically configure its interface.

**DELPHI Experiment Control Domains**

To organize the control system of the DELPHI experiment, objects belonging to a specific aspect of the experiment are grouped into an independent State Manager domain (SM-domain). All objects in one SM-domain are managed by their State Manager Process. To coordinate the activities of these individual SM-domains, certain 'abstract' objects of these domains have been made visible as 'concrete' objects in the central domains. The following paragraphs describe the different type of SM-domains participating in the DELPHI experiment control.

Each data acquisition partition is controlled by one State Manager. Because of the high degree of standardization, it was possible to write a single package of SML code for all the 18 partitions. There are two main objects in these domains to orchestrate the running of the partition. The OPERATOR object accepts the top level commands (e.g. start.run) from the local operator user interface. The LC object accepts commands from the central control when the partition operates as part of the whole DELPHI detector.
The slow-control domain handles the monitoring and control of technical aspects of a sub-detector. The most important aspect of the slow-control domain is the control of the High Voltage of the gaseous particle detectors. The raising and lowering of these volts have to be coordinated with the status of the LEP accelerator. Unlike the local data acquisition domains, there is no full standardization in the local slow-control domains. This is mainly due to the differences in the technical aspects of the individual sub-detectors, such as the requirements on High Voltage control. However, seen from the central control, the individual slow control domains are identical, i.e. they all have a set of identical 'top' objects with the same states and actions. Internally these domains are tailored to the environment required by the operation of the specific sub-detector.

The central control logic brings the individual partitions domains together in a coherent system. By controlling the partition SM-domains, it prepares and supervises the DELPHI detector for global running. There are at present two domains which coordinates the experiment as a whole. The central data acquisition domain controls all the detector data acquisition domains participating in the central run, as well as the central readout control supervisor and the central data logger process. The central slow control domain integrates the partition slow control domains and in particular will provide in the future the interlock between the High Voltage of the detectors with the status of the LEP machine mode. An important aspect at present being under study is to introduce in the central control the status of the LEP machine. This will allow the central control to take automatic actions triggered by the change of state in LEP machine, such as the automatic end of run and ramp down of detector high voltages before LEP prepares a new fill, or the ramp up of detector high voltage and start of run when LEP has
finished the preparation of a new fill and the beam conditions are stable.

Run status and run control interface

So far only the non interactive aspects of the experiment control system have been described. In theory, such a non interactive system could handle the automatic control of the run. In practice it is necessary to supplement it with interactive tools to monitor and control the experiment by the operator. These tools can also display information which is not taken into account by the run control logic, e.g. trigger rates, data size, ...), For this purpose, the Delphi User Interface (DUI) tool[7] has been developed. This graphical interface can access all available information of the experiment using the Distributed Information Manager (DIM) system.

Delphi User Interface

The DUI tool allows each operator in DELPHI to set up its own user interface. It merely provides a set of pre-defined displayable blocks (widget-trees), each of which displays one or more items of the experiment status and control. These blocks are combined, following the specification in the configuration file to form a user interface for a particular application. Many of the blocks use parameters to dynamically (re-)configure themselves (e.g. a block could for instance be read-only or provide a command interface). Each block knows by itself what kind of information to display and how to get it.

The configuration file specifies the following standard elements, of which any one can be left out: a menu bar, a pop-up menu, a status area and a work area. The first two are standard MOTIF widgets. The status area shows minimal information of some parts of the experiment control system, either as labels or as pushbuttons. If shown as pushbuttons, a click on them will pop up detailed information. The work area can show one or more detailed blocks. The blocks shown in the status and work area, as well as what pops up if one clicks on a status block, is all defined by the configuration file.

Defining new blocks for the DUI system is a three step process. First, the static part of the block is designed in UIL (User Interface Language) or using VUIT (Visual User Interface Tool). Second, a block 'creation' routine is made to subscribe to the service needed. The same routine can access parameters of this widget, given in the configuration file, and as such change the structure and or behaviour of the block. It can also take advantage of the information provided by the experiment control system (via DIM) to reconfigure itself. Third and last, an update routine is written to make sure DIM can update the block with the correct information. Since each block is self-contained, any routines belonging to that block will not be executed unless the block is specified in the configuration file. Maintenance of current blocks and introduction of new blocks is done on a block by block basis. So, the DUI system is thus NOT in danger, even if faulty blocks or code are introduced, as long as these blocks are not used in the configuration file.

The DUI system is implemented in C using Xlib, Xtoolkit and Motif routines. UIL is used to specify the static part of the blocks. A parameter system is used as an extension to UIL to make widgets configurable at run time. Apart from the standard Motif widget set two extra widget are used; a matrix widget to display tables of numbers and strings, and a dial widget to display information like trigger rates.
can be changed independently of the driver processes, it has become possible to stage the
gradually matured to the complex system described here. Since the experiment control logic
inherent modularity made it possible to commission it in a few partitions first. Since then it
some interesting built in features such as asynchronous execution and simple AI like rules. Its
put into operation in spring 1990. This concept uses a dedicated object oriented language, with
The first version of the DELPHI experiment control, based on the State Manager concept, was

### Distributed Information Manager

The Distributed Information Manager was designed, to access all information related to the
experiment from anywhere. This system runs on top of DECNET and TCP/IP. Any process
in the experiment can publish information (information server) and any client (e.g. DUI) can
subscribe to this information service. The subscriber can be kept up-to-date in an event driven
mode or at regular intervals.

A Name Server[7] keeps track of all servers. The servers register their information
services with the name server. Any client subscribes to an information service by making call
to the name server. The name server checks for authorization and returns the address of the
information server. If the requested service is not yet registered, the client will be contacted by
the name server as soon as the server is available. The name server makes it possible to move
any part of the experiment control system as well as any DUI from one node to another.

### Conclusions

The first version of the DELPHI experiment control, based on the State Manager concept, was
put into operation in spring 1990. This concept uses a dedicated object oriented language, with
some interesting built in features such as asynchronous execution and simple AI like rules. Its
inherent modularity made it possible to commission it in a few partitions first. Since then it
gradually matured to the complex system described here. Since the experiment control logic
can be changed independently of the driver processes, it has become possible to stage the

---

**Figure 2:** DELPHI User Interface configured for LEP status display.

**Table:**

<table>
<thead>
<tr>
<th>Mode</th>
<th>LEP Status</th>
<th>LEP Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Communication**

<table>
<thead>
<tr>
<th>Status</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Lump Status**

<table>
<thead>
<tr>
<th>Status</th>
<th>LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Lump Control**

<table>
<thead>
<tr>
<th>Mode</th>
<th>LEP Status</th>
<th>LEP Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**LEP Experiments**

<table>
<thead>
<tr>
<th>Experiment</th>
<th>L3</th>
<th>LEP</th>
<th>DPAL</th>
<th>DELPHI</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3</td>
<td>0.20</td>
<td>0.07</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>LEP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DPAL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DELPHI</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

---

273
With this tool we have transformed the initialization and setup of the data acquisition system from a sequence of complex operations into a simple operator action which completes in short time. As such, the system has greatly simplified the operator’s task of running the experiment and significantly reduced errors. Steps to enhance the experiment control further, including facilities for automatic error recovery, and automatic run control based on the status of the LEP machine are being introduced at present.

Although the experiment control system handles quite well the run control operation, it is not suited to handle run configuration setup or detailed run status information. Since the SMI based experiment control system is not tightly coupled to any particular user interface, we were able to adapt the user interface to the state of the art techniques and integrate the experiment control system into a configurable OSF/MOTIF based DELPHI user interface.

The distribution of parameters not directly related to the experiment control is handled by a Distributed Information Management system. Together with its intelligent name server, this system will also be used in the future to improve the performance of the information exchange within the current SMI system.

References

[1] The DELPHI Detector at LEP.

[2] The Slow Controls of the DELPHI detector at LEP.
T. Adye et al., This conference.

DELPHI DAS group, W. Adam et al., Computing in High Energy Physics ’91, Editors Y.Watase, F.Abe, Universal Academy Press, P643.


