THE OBJECT ORIENTED MODEL OF THE AD CYCLE AND ITS IMPLEMENTATION

H. Mulder, CERN, Geneva, Switzerland
G. Segura-Millan, University of Tokyo, Japan

Abstract

Central to the control and operation of the CERN Antiproton Decelerator (AD) is the deceleration cycle which involves accelerator sub-systems such as magnet current, timing, RF systems etc. It is fundamental to AD operation that these sub-system cycles are coherent and an integrated AD Cycle Editor has been proposed to guarantee this coherence. In the object oriented model of the AD, the highest level of abstraction is the class "AD Cycle" which is described in physical terms with an associated set of operations. The accelerator sub-systems inherit from this class thus guaranteeing coherence. The model is implemented in the AD Cycle Editor, which acts on the AD Cycle class and implicitly therefore also on the sub-systems. In this paper the model of the AD Cycle and sub-systems are discussed. The AD Cycle Editor is also presented with comments on the results of the commissioned system.

1 ANTIPROTON DECELERATOR

The CERN Antiproton Decelerator (AD) [1] is a circular accelerator that decelerates protons or antiprotons from 3.57 GeV/c down to 100 MeV/c through intermediate levels at 2.0 GeV/c and 300 MeV/c. At these momenta, beam cooling takes place using stochastic or electron cooling. The nominal AD Cycle is shown in Figure 1.

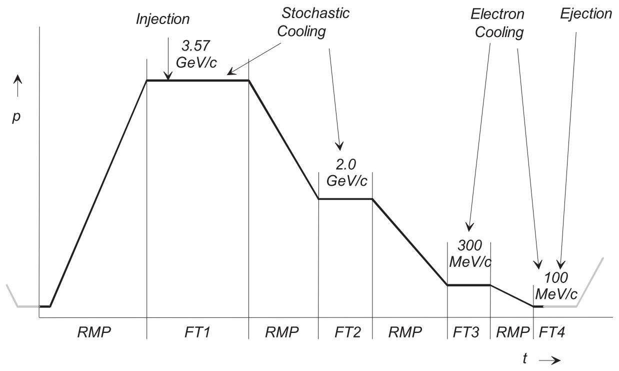


Figure1: AD Cycle Specification

The full deceleration cycle is in the order of minutes and the duration of the different intermediate levels are varied regularly, either programmed or on the fly. Experience with the Low Energy Antiproton Ring (LEAR) which operated in a similar regime had shown the necessity to be able to vary the momentum of the intermediate levels or insert new levels anywhere in the cycle.

The requirement for operational flexibility of the cycle layout with respect to both the time and the momentum axis had important implications with respect to the accelerator systems to be controlled. In particular the synchronization between the CERN PS and AD, the timing system and the decelerating systems: magnet power and RF.

The requirement led to a control system design driven by the operational need for cycle flexibility with the constraint that coherence between all systems subject to cycling – i.e. nearly all systems - be guaranteed.

An object oriented design approach was taken where the AD Cycle was defined as a class with a few fundamental operations. A number of sub-classes were derived to manage different accelerator subsystems.

2 AD CYCLE CLASS AND DERIVATIVES

At the highest level of abstraction of the model, we defined a class "AD Cycle" has been defined, to be compatible with the accelerator operator view. The operator sees the AD Cycle in physical terms of time and momentum.

At the next level, a number of accelerator system classes were defined that inherit the fundamental constraints and operations from the *AD Cycle* class but that have specific functionality to deal with the implemented accelerator systems such as timing or magnet current.

The *AD Cycle* is a base class that is defined by its attributes and operations - in accelerator language:

Attributes:
- Sequence of ramps and flat tops
- Momentum of flat tops
- Duration of ramps and flat tops
- Labelling of flat tops

Operations:
- Change momentum of any flat top
- Insert or delete a flat top
- Change duration of a ramp or a flat top
- Pause the execution of the cycle during a flat top

The *AD Cycle* class definition is the root of a tree of derived classes each of which deals with a specific cycling accelerator subsystem as shown in Figure 2.

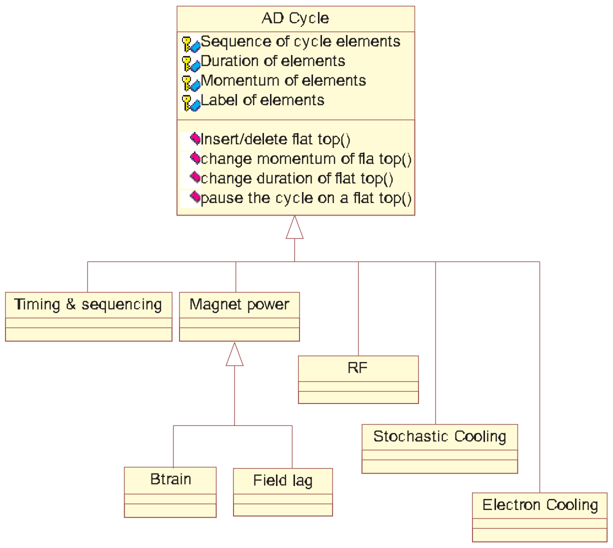


Figure 2: *AD Cycle* class hierarchy

Each of the derived classes inherits the attributes and operations of the *AD Cycle* class and most of them also implement specific attributes and operations. For example the *Magnet Power* class also implements an operation for changing the current level of a specific power supply at a certain flattop.

3 AD CYCLE CLASS VERSUS SYSTEM LAYOUT

The definition and design of the *AD Cycle* system is object oriented. The implementation of the class definitions is actually carried out at different levels in the system and in different environments.

When designing the system layout at a hardware level, it was tried to reflect the *AD Cycle* class properties as much as possible in the systematic use of available control modules. For example, the function generators used to control the magnet power supplies are used in a mode where the programmed function is a sequence of ramp function segments and the flat top generation is achieved by suspending the generation of the function for the duration of the said flat top. By trying to physically implement the systems as close as possible to the top level class definition, the subsequent software implementation of certain operations or attributes was trivial or even non-existent. As in the example above, the flat top duration is not actually programmed in the magnet power function generators and yet the flat tops are generated all the same.

The *Timing & Sequencing* derived class provided a particularly interesting challenge. The timing and sequencing system of the AD is an instance of the Master Timing Generator (MTG) in use at the CERN PS complex [2]. So far this MTG was used to control super cycles composed of fixed length individual cycles, providing a cycle-to-cycle modulation of the CERN PS complex. The MTG is very versatile and could in fact be used for the AD by dividing the AD Cycle into segments of fixed duration and allowing it to dynamically insert "do-nothing" cycles. This then provided the flexibility to stretch the AD Cycle and synchronize it with the CERN PS complex in spite of its different cycle length.

Because of its different origins, the timing system is not a good match to the *AD Cycle* class definition and as a result, the implementation of this class is complex.

4 AD CYCLE CLASS AND TOOLS

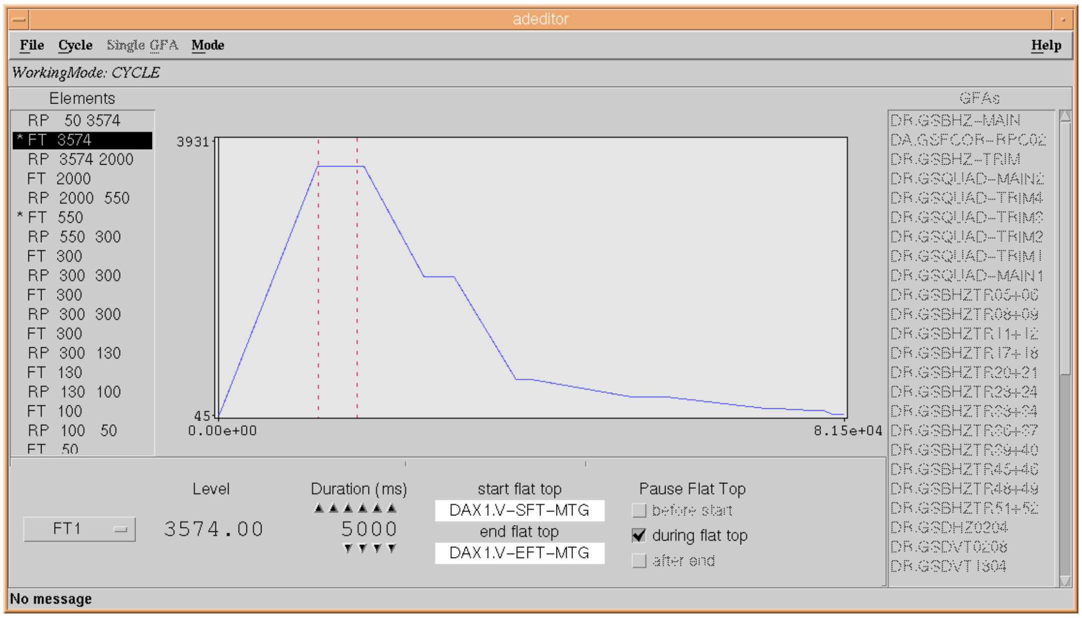
A number of tools have been developed that implement the *AD Cycle* class definition. Here the class hierarchy is found again. At the top level is the *AD Cycle Editor*, a tool that allows the manipulation of all the *AD Cycle* class attributes and operations and at the next level there are tools that allows the manipulation of specific attributes and operations of the derived system classes.

The *AD Cycle Editor* controls the *AD Cycle* object as well as - and this is very powerful - all derived systems. For example, the single action of inserting a flat top in the cycle with the *AD Cycle Editor*, is carried through to all systems. This is transparent to the operator, reducing effort and error.

As said, some of this control is implicit in the system implementation. Otherwise the control is explicit either by invoking the derived class implementations directly or by triggering other systems that implement these classes.

Without exception, all of the derived system classes implement specific attributes and/or operations. In the case of the *B-Train* and *Field Lag* classes, the specific attributes vary rarely and they are maintained through start-up files. The other derived system classes all warranted individual editors to manipulate the attributes and operations.

The integrated design approach taken has guaranteed coherence between the derived classes. Ideally this should be reflected in a coherent view of these classes. This has been achieved by integrating the specific editors for the *Magnet Power* and *Timing & Sequencing* classes into the *AD Cycle Editor* as alternative editor modes. The other derived system class editors are not yet integrated. A screen dump of the *AD Cycle Editor* is shown in Figure 3.



WorkingMode: CYCLE

Elements

```
  RP   50 3574
* FT  3574
  RP  3574 2000
  FT  2000
  RP  2000  550
* FT  550
  RP  550  300
  FT  300
  RP  300  300
  FT  300
  RP  300  300
  FT  300
  RP  300  130
  FT  130
  RP  130  100
  FT  100
  RP  100   50
  FT   50
```

GFAs

```
DR.GSBHZ-MAIN
DA.GSFCOR-RPC02
DR.GSBHZ-TRIM
DR.GSQUAD-MAIN2
DR.GSQUAD-TRIM4
DR.GSQUAD-TRIM3
DR.GSQUAD-TRIM2
DR.GSQUAD-TRIM1
DR.GSQUAD-MAIN1
DR.GSBHZTR05+06
DR.GSBHZTR08+09
DR.GSBHZTR11+12
DR.GSBHZTR17+18
DR.GSBHZTR20+21
DR.GSBHZTR23+24
DR.GSBHZTR33+34
DR.GSBHZTR36+37
DR.GSBHZTR39+40
DR.GSBHZTR45+46
DR.GSBHZTR48+49
DR.GSBHZTR51+52
DR.GSDHZ0204
DR.GSDVT0208
DR.GSDVT1304
```

FT1 — Level: 3574.00 — Duration (ms): 5000

start flat top: DAX1.V-SFT-MTG

end flat top: DAX1.V-EFT-MTG

Pause Flat Top: [ ] before start [✓ during flat top] [ ] after end

No message

Figure 3: *AD Cycle Editor*

5 RESULTS

The system has been in production since September '98 and a number of conclusions may be drawn.

The objective to provide an accelerator operator view and control of central part of the AD, the AD Cycle, has been achieved. Operations staff have a quick and intuitive grasp of the AD Cycle Editor and have used it successfully to set up the machine to decelerate protons down to 100 MeV/c. This certainly validates the overall design.

Another very positive aspect is that the design has proved extensible. In two cases, the accelerator commissioning revealed the need for additional accelerator systems. The first was the Synthetic B-Train and the second the flat top Field Drift. Both have a cyclical behavior and were thus by definition submitted to the *AD Cycle* class definition. In both cases the definition was respected and these new derived system classes were easily attached to the existing system.

On the down side, the *AD Cycle Editor* system on occasions lacks speed, it may take several minutes to carry out some of the *AD Cycle* class operations. This is mainly due to the necessity to carry out some of the write actions at specific instants of subsequent AD cycles. Since the cycle rate of the AD is in the order of minutes the loading takes at least multiples thereof.

Giving the operations staff tools with a physics view of the accelerator rather than a view linked to the underlying system created another challenge. By not showing the underlying system, little awareness is cultivated of those underlying systems. In actual fact, the knowledge of the entire system is embedded in the system itself and with the developers - in spite of a documentation effort to spread it. This means that problems related to the system are difficult to analyse for all but a few. This problem is particularly present in the *Timing & Sequencing* class.

If the *AD Cycle Editor* system is powerful in translating physics and operations needs into hardware settings, a similar system is now needed to do the reverse, i.e. diagnose and interpret low level problems at a high level.

6 ACKNOWLEDGEMENTS

The *AD Cycle* system and tools were an integral part of the overall AD project and thanks must go to all concerned with giving input for it's development and feedback to its validation. In particular we thank the PS CO group and the AD operations staff. Most of all we are grateful to have partaken in this exciting and original development.

REFERENCES

[1] S. Maury, “Design Study of the Antiproton Decelerator”, CERN Internal Report PS/AR/Note 96-34, 1996

[2] J.-C. Bau, G. Daems, J. Lewis, J. Philippe, “Managing the Real-Time Behaviour of a Particle Beam Factory: The CERN Proton Synchrotron Complex and its Timing System Principles”, Xth IEEE Real-Time Conference 97, Beaune, France, September 22-26, 1997, p-553