AtR Controls Evaluation Group
Meeting Summary

Garry Trahern

March 12, 1996

Contents

1 Introduction **3**

2 Vertical Integration Projects **5**

2.1 Beam Loss Monitors 5

2.2 Beam Charge Monitors 6

2.3 Flags 6

2.4 Magnet Power Supplies 7

2.5 Beam Position Monitors 7

3 Front End Computers and Support Systems **9**

3.1 Front End Computers 9

3.2 AdoIf 10

3.3 Pet 10

3.4 Fit and Crate 11

4 Horizontal Integration **12**

4.1 VET 12

4.2 UI Tools 13

4.3 Glish 13

4.4 Code Management and Development Environment 14

5 Database Systems **15**

5.1 RAP data 16

5.2 Controls data 16

5.3 Database access and program interfaces 16

5.4 Naming convention 17

6 System Administration **18**

7 Conclusion **20**

A Action Item Summary **21**

Chapter 1

Introduction

This report summarizes the conversations during the month of January 1996 following the commissioning of the U and W lines of the AGS to RHIC transfer line in the fall of 1995. The scope of the meeting was twofold:

1. To review the commissioning experience and
2. To propose action items to correct problems as well as to begin to think about the sextant test.

The agenda for the meeting was composed of the following topics. The person(s) who lead the discussion are indicated in parentheses.

```
1) Vertical projects.
 a) BLM  (Steve Tepikian)
 b) BCM  (DP Deng)
 c) flags (Jorg Kewisch, Larry Hoff)
 e) magnets (Jorg Kewisch)
 f) BPM's. (Todd Satogata)

2) FEC (Tom Clifford)
 a) core system / Ado / reboot
 b) AdoIf

3) Console Level Computers
```

```
   a) pet (Don Shea.)
   b) fit, crate (Bob Olsen)

4) Configuration databases  (Garry Trahern, Waldo MacKay)
  a) DB++  (Steve Tepikian)
  b) db access tools (Garry Trahern)

5) Development (Ted D'Otavio, Todd Satogata)
 a) release/replicate strategy (Todd Satogata)
 b) styles and rules (Makefiles, etc.) (Todd Satogata)
 c) GUI (Ted D'Otavio)
 d) GLISH/SDS/value class (Todd Satogata, Jorg Kewisch)
 e) what do do about glish? (Todd Satogata)
 f) glish support & development? (Todd Satogata, Jorg Kewisch)

6) Computer systems (Steve Peggs, Rich Cassela)
   admin / computers / file systems

7) Naming Convention (Garry Trahern)

8) Data summary & logging  (Pat Thompson, Jorg Kewisch)
```

The summary report will be organized along the topics above, and I would like to thank the speakers for providing copies of their transparencies. Most of the text in this document was extracted from these sources with notes added from the discussion that accompanied each topic. The detailed list of action items arising from the discussions are collected in an appendix to the report for easier reference.

Chapter 2

Vertical Integration Projects

This chapter covers the following applications:

```
 Vertical projects
a) BLM
b) BCM
c) flag
e) magnet and power supply
f) BPM
```

Some general comments on applications before listing the various specific comments are in order. The BLM, magnet power supply, BPM and flag applications all used the following ideas or tools in their implementations: UItools, glish, the manager concept and of course AdoIf. The implementation of the manager for each application was more a matter of the developer's vision than any devotion to a common interface definition – something which is desired but which does not yet exist. Consequently, there was a strong desire expressed to correct the lack of standards for managers in the next phase of the control system.

The BLM and magnet power supply applications also used online access to database tables to initialize data sets. On the other hand, the BCM application was built entirely around Pet in this first phase.

2.1 Beam Loss Monitors

The beam loss monitors performed as expected. However, during some tests late in the run it was found that the application had a significant impact on the console's cpu. This problem

was traced to a basic setup calculation that was being performed repeatedly but only needed to be done once, and it has been fixed subsequently. Other issues that arose during the discussion included the timing of data acquisition and display. The other major comment was that since all the instrumentation data tied to longitudinal location will eventually be needed by the steering application, better integration of the BLM application with the charge monitors and beam position monitors is needed.

2.2 Beam Charge Monitors

The beam charge monitor was a Pet page during the AtR commissioning. There was general agreement that the BCM should be integrated with the BLM application if time permits. Action items emerging from the discussion are listed in the appendix.

2.3 Flags

The beam profile monitors were a good example of vertical integration. The assembly of a small group devoted to the entire system allowed early deployment and debugging of the hardware before the start of commissioning. However, the loss of the system commissioner complicated the development of the manager and console application(s).

The manager made efficient use of AdoIf asynchronous requests and grouping. It also separated scale factors, offsets and rotation parameters from the raw data when images were saved for off-line analysis. However, the manager lacked the ability to control flag settings.

The data display for the flag application used Tk widgets with consequent performance degradation. In addition a memory leak of unknown origin caused the flag display to crash after about 150 flag displays.

The flag control panel was criticized on several fronts. The lack of a complete command set, e.g. to set timing, black level and gains was noted. The panel's initial conditions were often misleading, and there was no way to save and restore these settings once they were established. The lack of error reporting was also noted.

2.4 Magnet Power Supplies

The magnet power supply controller was the most trouble free of the console applications. Its implementation currently has three displays (based on equivalent executables) connected to Glish and the magnet manager. A possible modification is to have one generic magnet display interfaces connected via Glish to beamline specific managers (e.g., uw, x, y and sextant).

Since the wave form generator tables were not used in this implementation of the power supply application, the relevance of the this application's performance for the sextant test and RHIC is unclear. Many other comments on the magnet application are located in the appendix.

2.5 Beam Position Monitors

We begin the discussion of BPMs with some comments on instrumentation hardware and firmware even though these aspects are not explicitly part of the control system. Worries about a possible drift in gain stages was allayed after they were re-measured halfway through the run and found to be small. However, the dependence of the signal on bunch shape is worrisome, especially since the new coalescing schemes may modify the bunch shape. Finally, it was noted that bunch triggered electronics are difficult to setup and time – especially with high gains.

Moving onto controls issues, it was found that attenuator and gain setting interfaces were unnatural and confusing. There was no read-back capability. In addition online calibration capabilities are essentially absent. The cross dependencies of the analog and digitizer board calibrations makes the calibration process complex, but something must be done here since we cannot take systems off-line during operations to calibrate.

Online and off-line analysis of BPM data was enabled by the use of consistent data and class structures. However, the off-line analysis tools were not implemented for routine shift use. Correlated storage of magnet setting and orbits was implemented effectively, but BPM manager should not have to start another magnet manager process in order to generate the magnet data. No high level sequencing (i.e., beam threading) was tested during the run. However, the current system design admits this implementation easily.

Overall performance issues still remain. We are unsure if the system currently imple-

mented via managers and glish can scale to the next test. The "data driven" philosophy we have adopted where the manager collects and controls the data and the interface only displays it does not appear to be robust. The inability of the BPM application to deal with a flood of data from the FEC's is symptomatic of a larger problem.

Chapter 3

Front End Computers and Support Systems

3.1 Front End Computers

Ado software ran well on FEC's. The ability to hide details of the hardware from the application developers was successful. Large and small data packets were handled within the same framework (compared to Epics which still cannot do this effectively at last hearing).

There were bugs in FEC's and Ado's which caused system crashes. While some of these have been fixed, there are still others to work on, and this will be a continuing problem since it is difficult to identify the source without being able to reproduce the situation which causes the crash. One of the features/limitations of VXWorks is the openness to the front end computers it provides, i.e., Ado pointers can overwrite anything in memory.

Some desired (and planned for) functionality in the FEC has not yet been implemented. The boundary between general FEC functionality and functions tied explicitly to a particular Ado is under investigation. This includes the issues of alarms and other system notifications that should be reported from the FEC. Another item to be addressed is the preservation of the FEC configuration setup in static ram. Also mentioned in the discussions was that background diagnostic processes may help predict imminent FEC failures. Finally, issues of general FEC access control (e.g., remote user read-only capability, etc.) will be investigated as well.

Another issue was that startup and reboot of the systems took too much time. In addition during the run there was no uniform startup procedure or philosophy in effect. This and other

FEC issues will be addressed within the task force environment that has begun again.

A general performance report on cpu and memory usage as well as communication band width is desired to gain confidence that FEC systems will scale as the load increases.

3.2 AdoIf

There were some specific comments about AdoIf, the Ado interface class. For example, Ado instance creation is slower than desired. Also, the GetAsync method is complicated when trying to extract measurement values. The magnet manager resorted to polling of the MADC in order to access data uniformly. Some ability to select the polling frequency and event is needed. Generally speaking there was a desire for a way to monitor the status of the Ado instances in an FEC as well as the FEC itself. These issues are under discussion in the working groups.

3.3 Pet

Pet was a good diagnostic and engineering support tool. It worked as advertised (usually) and was robust. Some of the problems identified during the discussions were that new page creation was slow, the widgets used to construct graphs had bugs, some error returns were wrong or absent, the middle mouse button caused Pet to crash and finally that the value editor was cumbersome to use. The latter problem is tied to the structure of glish's value class which is itself already the subject of a re-design investigation.

Many if not all of these problems have already been addressed and fixed. Other improvements currently in the works (or completed as of this writing) include:

1. Save and restore of data on page.
2. Define graphs in the page description file.
3. More styles for graphs.
4. More than one Ado to a row of a Pet page.

3.4 Fit and Crate

Fit and Crate were two diagnostic tools for monitoring the status of FEC's. Fit (for Front End Interaction Tool) was a Perl/Tk program that spawned various programs that would monitor aspects of an FEC. For example, one could spawn an *ECTO* (Event Connection Tool) process which can make, break or list event connections. It could also reset an FEC or spawn the Crate program.

The Crate program periodically tests the FEC utility module and lists its Ado parameters (e.g., chassis power supply status bits, temperature, fan operation, etc.) It represents this data within a graphical picture of a VME chassis. Both of these tools can be the basis for more elaborate diagnostics of the FEC's if there is time to develop them further.

Chapter 4

Horizontal Integration

The conclusion of the evaluation group was that genuine horizontal integration was lacking at the application level. Beyond the common use of various tools whose discussion is included here as part of the horizontal integration efforts, the applications did not incorporate common designs. In particular, managers were implemented without a set of shared interfaces. Communications between one manager and another were incorporated on an informal basis by developers. A general critique of each manager design was not elaborated during the review. Consequently, a working group has been established to study the problem of manager design. Perhaps a consensus for where commonality can be implemented will emerge from the working group.

The remainder of this chapter discusses the software tools in common use.

4.1 VET

VET (for Value Editing Tool) uses the AppPage library in /usr/controls/lib. This toolkit enables user interface building. It supports both glish and the UItool package. The profile monitor application user interface was built (very rapidly) using VET. The following comments about VET in the profile monitor(VPM) are recorded here, but also see the section on flags in the chapter on vertical integration for further comments about the profile monitor interface.

1. VPM manager used adoIf grouping capability.

2. GetAsync used from the beginning.

3. Shared memory Sds used.

4. AppPage library was created during the VPM implementation. This is a general tool for send/receive of display and supports interaction via Glish events and UITools.

4.2 UI Tools

The group made uniformly nice comments about the UItools package. There were kudos for the documentation. UITools is layered on Motif. It currently supports tables, graphing and printing. Future developments are addressed below.

1. Future developments

   (a) Documentation via web browsers.

   (b) XRT widget package for 2D/3D graphing.

   (c) Drawing tools so that beamline or timelines can be shown.

   (d) Improved printing capabilities (postscript, printer selection).

   (e) Guidelines for developers – how to develop a common look for users.

2. Other comments from users

   (a) Standard behavior (button-2 confirm) expected, but not supplied.

   (b) Continuous UI working group is proposed.

4.3 Glish

Glish was used in four of the applications: BLM, BPM, Magnet power supply and Profile monitor. Although each of these applications was successful, there are unanswered questions remaining with regard to performance as well as implementation. For example, whether glish should be used for inter-process communication as well as sequencing is still open. For example, a blocking situation occurred during the development of the power supply application (UI and manager) where the communication buffer from the manager to the UI would fill and would not be released. This problem is inherent in the TCP/IP communication protocols used by glish, and it is not clear how to "solve" it.

Glish is flexible but slow taking $\sim$ 1 second from the time of initiation of a request at the user interface and then passing onto glish and the manager and finally coming back to the UI by the reverse path. Whether the delay is occurring in glish or the manager is unclear. Further study is needed. In addition the scalability of glish based applications is not established, and debugging of glish based applications is complicated when the process cannot connect to glish.

Support for and development of glish is a continuing problem. While we need to maintain and develop the existing relationship with the NRAO/AUI development group, we should not become reliant on them. We need to establish local permanent support for ISTK (glish and Sds).

4.4 Code Management and Development Environment

The code development environment for AtR commissioning allowed separate development of applications and front end system code via the /apps and /ride file systems. Having these two development areas provided a clear separation of tasks at the expense of non-uniform maintenance and release coordination. In addition some of the other shared tools and libraries are still scattered throughout the file systems. It was felt that this separation should disappear in the next phase of the control system.

Some tool development for the software environment needs to occur as well. In particular although central (included) Makefiles have streamlined the use of Makefiles, they are not used widely enough, i.e., they have only been used for high-level applications so far. We need a project level Makefile. There has also been a strong desire expressed for a class browser, but previous research on this software has not yet found one that works on SUN architectures. Finally, there is a need for tools to help build a simple framework for documentation. One possibility is **doc++**, a free latex based documentation generator. It analyzes one's code and header files and constructs a html template that can be further edited for detailed explanations. The document can then be posted to the web for general viewing.

Chapter 5

Database Systems

A general comment on AtR (and RHIC) database systems is that successful data management requires that

1. Data owners take primary responsibility for their own data management – i.e. unless the data owner is willing to contribute to the effort, it won't succeed. There are resources available to help groups develop data structures and (to a lesser extent) access and reporting tools. But the responsibility to install and maintain this data must always reside with the owners.

2. The groups agree on a clear statement and description of what data will be managed.

3. Controlled interfaces be constructed.

4. Rate of change of data be adiabatic. Otherwise, application developers will work directly with front end systems for the latest data and will decouple their efforts from electronic database systems.

The online use of the database was restricted during commissioning because both a wider understanding of Sybase by RAP and Controls is needed before applications are strongly tied to it, and a migration from the development server to an operations server is needed with added personnel assigned for support. In any event, at least two of the applications and PlotAtr(the graphical view of the complex) used the database online, and since there were no network or hardware problems with the database server during the run, the online use of the database should be considered a limited success.

The C++ libraries, DB++, that Steve Tepikian constructed were essential in making database access available to the application programs. It is desirable that Controls personnel should learn to use these libraries in pre-operational test situations so that integration of the database with various services becomes easier in the future.

5.1 RAP data

The following databases were managed by the RAP group.

1. atr - the AtR lattice desgin
2. atr_gddb – configuration data for AtR
3. atr_cable – cable data for AtR engineering systems
4. atr_cal – calibration data for magnet transfer functions, BLM constants

5.2 Controls data

The following databases were managed by the Controls group.

1. ado_config - Ado class and instances definition as well as Ado parameter meta data.
2. control_inventory - front end systems and components database. Used to keep track of chassis, crate and cards.

5.3 Database access and program interfaces

The group has access to a variety of tools for database access. Sybase provides two basic tools, isql, a line based SQL interpreter and DWB (data work bench), an X-window based SQL interpreter. We have purchased the PowerBuilder development suite for PC forms, but have yet to use it extensively. (One application was built for the RHIC cable database, but it has not been field tested as yet.) The other access tools have been developed using the web via Perl and the html/cgi-bin API definitions. Waldo MacKay has developed an extensive set of database queries for AtR configuration data, and this path seems to be the preferred one. Sybase has released a product called WebSql which integrates the public domain Sybperl package with html forms.

Steve Tepikian has developed a C++ wrapper for the Sybase Open Client libraries. These libraries were effectively used in the magnet power supply and BLM applications during the run. Steve is currently rewriting part of the code to incorporate more of the functionality of the Sybase libraries.

5.4 Naming convention

The AtR and RHIC device naming convention was summarized in appendix D of the AtR Commissioning Task Force report (RHIC/AP/53). This naming convention has been used both in the installation and control systems for the AtR. The syntax requires that one construct a unique global name by concatenating the mnemonics from a subset of the following: Machine name, Section name, Device type, Device number and Device multiplicity. The name is case insensitive and has no fixed length although there is a preference for shorter names where possible. For example, in the AtR, **uq1** is the name of the first quadrupole in the U line. For RHIC, with two rings of periodic structure, **bi5-qd15** is the name of the blue ring (inner), sector 5 quadrupole at half cell 15. A list of abbreviations (such as q, qd, qf, etc.) for device types is maintained on the database server and can be viewed from the AtR web page.

These names are referred to generically as SiteWideNames (SWN) since it is hoped that everyone will be conversant with at least this name for a device. There are also other independent (unique) names for some devices which arise from the device's origin in one of the engineering/equipment groups. For example, a magnet cold mass may have a Serial name which effectively labels the device for the purposes of magnetic measurements. In such cases we will maintain a lookup table in the database to correlate the SiteWideName with the Serial (or other) name.

In the discussion it became clear that there is the possibility of confusion between RHIC names and names in the AGS and Booster systems. A committee has been established to address these problems and to complete the naming of devices both on and off the beam lines of RHIC.

Chapter 6

System Administration

The hardware in the AtR controls and development environment was composed of about 50 workstations including 43 Suns of which 14 are Sparc 20 models and 7 SGI Indys. X terminals were located in the equipment buildings. There are also four file servers and one database server. Three persons perform system administration tasks from a common area using secure hosts.

The software for the controls system was developed on the file servers and then released to the console computers via a replication process. Although this process worked successfully during the run with system wide replications done every Tuesday at noon, the process was inefficient in that everything was copied to each console. In the future only changes will be replicated, and the process should go more quickly.

Prior to the run it was agreed that non-operations subnet dependencies should be removed from the environment of the operations accounts and console computers. This did not happen during the run, and we found during a shutdown of one of the development file servers that the mcr account hung trying to resolve path dependencies.

The AtR test was run from several special accounts: mcr, atr and ess. The latter account was to be used mainly by the engineering support staff, for example to run Pet in the equipment buildings during system shake downs. The mcr account is the 'main control room' account used by both AGS and AtR control systems. The use of a common account for the console computers was successful.

It was generally felt that having two monitors and one keyboard per control station was not satisfactory. However, since the current AGS console upgrade plan is to have three

monitors and two keyboards available, this problem will disappear.

PC's running Linux were effective as a development platforms. In order to make them more effective, the motif development libraries need to be available. A test of Pet and other applications should be performed once this occurs.

Documentation for applications was not successful. What documentation exists is available via the web. This should be developed further since it is the only common source of text available from all computer architectures.

Future issues for system administration include the upgrade to Solaris 2.5, the possible switch to the Common Desktop Environment (CDE) as well as the implementation of a common development subnet. The former two issues are more or less straightforward if desired. The move of RAP from subnet 90 to 104 could possibly occur sometime in March. This will probably necessitate some adjustments in the software organization. But this transition should help in making the software collaboration between RAP and the controls groups more transparent and reliable.

Chapter 7

Conclusion

This report has summarized the discussions held in January 1996 regarding the AtR control systems. As the participants of the meetings may have noted during their reading of this document, in most topics I have not supplied extensive commentary. Even so it is entirely possible that my recording of the conversations as well as the context of these discussion may not be accurate. The responsibility obviously lies entirely with the author.

The following appendix lists the action items identified throughout the text.

Appendix A

Action Item Summary

The action items indicated throughout the text are listed here. The most critical topics and items are listed first.

1. Horizontal integration issues

   (a) Support for and development of glish (as well as general ISTK support) should be increased beyond the present ad hoc situation, even if this requires an additional FTE.

   (b) Continuous UI working group is needed.

   (c) Manager design working group is needed. Managers are too eclectic at present. They need an interface standard that all applications should follow.

   (d) Need correlation of BCM and BLM systems with BPM systems.

   (e) Naming convention for on and off beamline elements should be established as quickly as possible. A committee has been established to address this problem.

   (f) Other horizontal issues

      i. Linux on PC's should be supported. This requires that Motif development libraries be available. A port of Pet and some application to a PC running Linux would be the next step in a proof of principle.

      ii. Determine the best way to alias channels in the MADC to the names of the parameters to be used by applications.

      iii. We need a trouble log for applications so that developers do not need to depend on fortuitous word-of-mouth communications to know about problems.

iv. A uniform standard for data (logging, analysis, plotting) is needed.

v. Standards for database access (via DB++ ?) should be developed.

vi. Improved printing capabilities (postscript, printer selection).

vii. Some functionality in UI tools is missing. For example, button-2 event in text field is desired.

viii. Both the link statement and string variables in the client function of glish do not work and should be fixed.

2. Vertical integration issues

   (a) Beam Loss Monitors

      i. Array bounds were exceeded occasionally in BLM manager. Is this a glish problem - perhaps in sds_resolve?

      ii. Event update on AGS pulse rather than AtR pulse requires a better event to trigger on.

      iii. Calibration of BLM power supply settings and read-back in volts is needed.

      iv. Table display for BLM readings (pop-up menu item) is needed.

      v. Monitor selection list is needed in UI.

      vi. UI display takes "too long" to update.

      vii. Need clear indication when saturation occurs.

      viii. Logarithmic scale didn't work in BLM plots.

      ix. Time indicator on graphics display is needed, and UI should use a circular buffer and reset the buffer counter instead of sticking at 99.

   (b) Beam Charge Monitors

      i. Set fixed Y-axis scale in graphs.

      ii. Save/restore of hardware settings (from PET save/restore?).

      iii. Incorporate BCM into beam loss application or make a manager for BCM's.

      iv. Plot of transformer_2 divided by transformer_1 should be a default graph of the application.

      v. Auto-scale should be disabled.

vi. Implement notification of the status of (and a change in) charge state.

(c) Flags (Manager and User Interface)

i. Some flag control settings were missing in manager implementation.

ii. Incomplete command set – timing, black level and gain were missing from UI control set.

iii. Need capability to save/restore flag settings as in other applications.

(d) Magnet Power Supply

i. Design values for strengths and currents should be displayed.

ii. Error diagnostics using pop-up windows needed.

iii. Connection to orbit steering routines needed.

iv. Knobbing desired (See $\rightarrow$ UI working group).

v. Use pull down menus for off/on/standby.

vi. Fix transfer functions.

vii. Allow for $\frac{dE}{dx}$ loss in flag and foils.

viii. Fix ADO's so polling is unnecessary.

ix. Need to decide on and correctly display the PLC error codes.

(e) Beam Position Monitors

i. Online calibration capabilities are essentially absent. The cross dependencies of the analog and digitizer board calibrations makes the process complex, but something must be done since we cannot take these systems off-line during operations.

ii. Startup of system is cumbersome. User should not have to choose BPM's and planes manually.

iii. Better indicators of "good" and "bad" data are required.

iv. Off-line analysis tools should be available for routine shift use.

v. Need correlation of BCM and BLM systems with BPM systems.

vi. Calibration coefficients as well as instrumentation card information should be stored in a database. Timing information for acquisition gates with respect

to FEBbunch for various RF setups might also be data-based, or baring that, a more consistent procedure for the setup should be established.

3. Front End Computers and AdoIf

   (a) FEC access control should be investigated to prevent remote user conflicts with main control room operations.

   (b) A general performance report on cpu and memory usage as well as communication band width is desired to gain confidence in the FEC systems ability to scale to RHIC.

   (c) Need improvements to GetAsync allowing programmers to specify polling frequency or event.

   (d) FEC's might send values when a change exceeds threshold.

   (e) Monitoring of FEC health is desired.

4. Other

   (a) Need project level Makefile. (See section 4.4)

   (b) Shared tools and libraries are scattered over the file system. They should be brought to a common location.