Experimental Physics and Industrial Control System (EPICS)

Archiving Reference Manual

DRAFT

Roger Cole

January 25, 1993

This software was produced under U.S. Goverment contracts: (W-7405-ENG-36) at the Los Alamos National Laboratory, and (W-31-109-ENG-38) at Argonne National Laboratory.

Initial development by:
The Controls and Automation Group (AT-8)
Ground Test Accelerator
Accelerator Technology Division
Los Alamos National Laboratory

Co-developed with
The Controls and Computing Group
Accelerator Systems Division
Advanced Photon Source
Argonne National Laboratory

~gta_epics/Frame/ar.drw/refMan.fdr/quick.book

TABLE OF CONTENTS

TABLE OF CONTENTS

1 INTRODUCTION

1.1 purpose

This manual describes the EPICS archiving software. It is intended for those who need to archive real-time data or to retrieve data from archive data files.

1.2 EPICS archiving software

EPICS archiving software supports acquiring real-time data, saving it on disk, and retrieving the data from disk for plotting, printing, and exporting for use by spreadsheet programs. The software consists of four primary programs. Archiving can be done interactively with AR or with AR_cmd; AR_cmd can also be used in scripts and can run in 'server' mode. Retrieving archived data is done with ARR or with ARR_cmd; the latter can be used in scripts.

1.3 using this manual

This manual provides both quick reference information and detailed documentation for various features of the EPICS archiving software. Often, assistance for performing routine tasks will be adequately provided by the 'pictures'—generally a picture has a set of numbered steps for performing a task. Detailed information about those steps can be found in the paragraphs following the 'picture'.

A companion AR and ARR pocket-sized quick reference summarizes much of the information in this manual. It is useful as a 'memory jogger' for those who are already familiar with using the software.

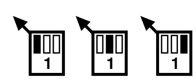
1.4 document conventions

Bold italic typeface is used to indicate operations with menus. For example, selecting a 'by channel' file for retrieval uses the 'setup...' menu item on the 'Retrieve' button on the main ARR screen to bring up the 'retrieval setup' window. This is indicated by ***Retrieve/setup...***.

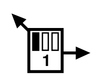
Bold italic is also used to indicate mouse operations, as in ***hold right*** and ***click left***.

In many figures, special symbols are used to show what operations are to be done. The order of the operations is indicated by the sequence of numbers in the symbols.

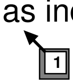
The following indicate a mouse click, using one of the 3 mouse buttons:



Text is selected by holding down the left mouse button and dragging it to the right (or left), as indicated by:



Typing into a text field appears as:



[ ] enclose optional parameters. (The brackets must not be typed as part of the input.)

{ } enclose parameters, one of which must be chosen. (The braces must not be typed as part of the input.)

`^D` is the 'control D' character, used as a synonym for the `quit` command. `^D` is obtained on the keyboard by holding down the 'Control' key and typing a 'd' at the same time. (Don't hold down the 'Shift' key, though.)

1.5 terminology

channel a channel is either 1) a record name or 2) a record name followed by a field name. When the field name is omitted, the `.VAL` field is used. There are several types of channels referred to in this document.

A numeric channel is one that has a numeric value, such as the value field for an analog input record. Some numeric channels have integer values (such as a `.RVAL` channel); many others have floating values.

A discrete channel's value is taken from a set of 'state strings', so that the value is a text string rather than a number. The `.VAL` channel for a binary input record is a discrete channel, as is the `.SEVR` channel for any record.

A channel whose value is a single number is referred to as a scalar channel. An array channel has a value composed of several numbers. Waveform records and compression records are array channels.

filled channel a filled channel is one that isn't beam-synchronous. For a synchronous snapshot, the value for a filled channel is 'filled in' by using the most recent value.

graphics information for a channel is information from the record about high and low operating range, engineering units, and alarm limits.

synchronous channel a synchronous channel is one that sends a value to the archiver on each system trigger. Generally, the `.ADEL` field for a record must be set to `0.` to ensure that the `.VAL` field behaves as a synchronous channel.

synchronous sample a synchronous sample for a set of channels contains the values for all the channels *for the exact same time stamp*. If a filled channel didn't change at that time stamp, it is filled in. If a synchronous channel didn't change at that time stamp, it is marked as 'missing'.

1.6 restrictions

The following restrictions apply to the EPICS archiving software as of Aug 24, 1992.

The `GRAB` sample feature isn't implemented.

Support for color PostScript printers and EPS files isn't implemented.

The AR, ARR, and ARR_cmd programs run only under the X Window System.

Printing of plots requires a PostScript printer.

Archiving doesn't properly handle variable length arrays, such as compression records whose size is changed by an operator.

'sample set' archive data files don't contain graphics information for the channels.

2 OVERVIEW OF ARCHIVING

Archiving acquires real-time data and saves it in disk files. Archiving is done with either AR or AR_cmd. Section 2 provides a general discussion of archiving. Section 3 covers using AR for archiving and section 4 covers AR_cmd.

2.1 programs for archiving

AR is an interactive program that provides a graphical user interface. It is used for interactive archiving activities. AR is usually used for creating and changing archive request files, since it has friendly tools for verifying proper operation.

AR_cmd is a program that uses text commands. It can be used in interactive mode or in client/server mode. The latter is used for 'unattended archiving'. Since AR_cmd uses text commands, it can be used either from the keyboard or from scripts.

AR_kbd was the predecessor program to AR_cmd. The expanded and altered content of archive request files used by AR and AR_cmd has rendered AR_kbd obsolete. Although AR_kbd will continue to be available, it is no longer supported and users should convert to using AR_cmd.

2.2 acquiring and saving real-time data

Archiving activities are specified with commands in 'archive request files'. An archive request file specifies a set of channels to archive and specifies how the archiving is to be done for those channels.

In order to use an archive request, the archiver must be told to 'open' it. When it opens an archive request, the archiver reads the file, makes connections to the channels in IOCs, and begins performing the processing specified by the request.

When an archive request is opened, it is initially in the 'inactive' state. Usually, the archive request file contains a command to activate it immediately and to keep it activated until either the request is closed or the archiver is stopped. In special circumstances, additional flexibility is gained by using special conditions for activating and inactivating an archive request.

Sampling of real-time data occurs only for requests that have been activated. Sampling doesn't actually start, however, until a 'trigger' occurs. Once sampling begins, it continues as specified in the archive request.

Sampling is based on monitoring the channels for changes in their values. When a channel changes more than its archive deadband (the `.ADEL` field), the new value is sent by the IOC to the archiver. For 'beam synchronous' channels, the `.ADEL` field should be set to 0. The samples can be acquired either as streams of values or as snapshots containing synchronous samples.

Time stamps play a critical role in sampling. A data value will be discarded in several cases:

1. if its time stamp is zero;
2. if its time stamp is the same as, or earlier than, the time stamp for the previous value received from the channel; or
3. if its time stamp is the same as, or earlier than, the time stamp of the most recent sample written to the file.

(An alternative method of sampling is available with the `GRAB` command in an archive request. This command gets the present value of the channel, without regard to deadband issues.)

Two different forms of archive data files are available. A ‘by channel’ data file allows optimized use of disk space and can handle high data rates. A ‘sample set’ data file is less efficient in its use of disk space and is appropriate for lower data rates

2.3 archive request files

An archive request file contains a list of channels to archive and a set of commands for how to archive. In addition, archive request files can be used to guide retrievals of data. Information and comments about an archive data file can be stored in the archive request file that corresponds to it.

Archive request files are kept in the `arReq` subdirectory. They have `.req` as an extension. Typically, AR is used to create, modify, and test archive requests. AR supports checking syntax, verifying that the channels exist, and determining the behavior of the data. The latter is especially useful in making sure that channels that are intended to be beam-synchronous actually produce a value on each system trigger.

When archiving is actually in progress with AR or AR_cmd, several archive requests can be open at the same time, each saving data to its own data file. Two archive requests can’t store data in the same archive data file.

2.4 ‘by channel’ archive data files

A ‘by channel’ archive data file is a special binary format file that stores data as time-stamped streams of data, one stream per channel. The format of these files conserves disk space and allows higher-speed, higher-volume archiving. Because of their special format and compact nature, they are also faster to use when retrieving data from the file. ‘By channel’ files can be used for either ‘monitor’ or ‘snapshot’ sampling.

The first time a channel is stored in a ‘by channel’ file, its graphics information is also stored. (If the graphics information changes later, those changes aren’t stored.)

Because their format is complicated, ‘by channel’ files are somewhat vulnerable to corruption if the computer crashes. This risk is especially important for ‘around the clock’ archiving. In many cases, the corruption simply prevents continued archiving and requires saving the corrupted file and creating a new one; in some cases, however, none of the data in the file can be accessed.

A maximum size is associated with ‘by channel archive data files when they are created. A file will begin small and grow until its maximum size is reached.

Choose ‘by channel’ files when:

1. using `MONITOR` as the sampling type;
2. the expected volume of data is large and it is desired to optimize the use of disk space and the speed of retrievals;
3. the number of channels is large and the data rate is high;
4. the speed of retrievals is important;
5. the risk of having the file be corrupted is acceptable.

2.5 ‘sample set’ archive data files

‘Sample set’ archive data files are in a readable text format that stores data as synchronous snapshots. These files are much less efficient in their use of disk space. They support relatively low data rates for archiving and they are slower when retrieving data.

‘Sample set’ files can be used only for ‘snapshot’ sampling. At the time a snapshot is stored, the file is opened, the snapshot is written, and the file is closed.

Because of their simple format and the way they are handled, ‘sample set’ files are extremely reliable. It is also easy to manipulate ‘sample set’ files, even while archiving is in progress. For example, a ‘sample set’ file can be deleted or renamed at any time—the next time the archiver tries to write a snapshot, the file will be recreated.

There is no limit placed by the archiver on the size of ‘sample set’ archive data files.

Choose ‘sample set’ files when:

1. the number of channels and the volume of data are relatively low;
2. speed of retrieval isn’t a high priority;
3. reliability must be extremely high.

3 ARCHIVING REAL-TIME DATA WITH AR

AR is the interactive version of the EPICS archiver. Most commonly, AR will be started from a button on the 'GTA Operator Window' (see sec 9). When started this way, the archive request for the experiment will automatically appear in the AR window.

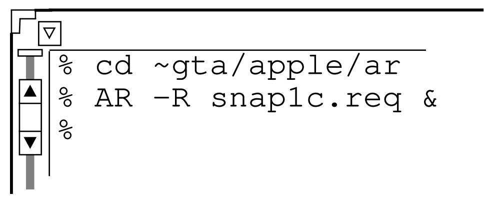
This section gives a brief overview of using AR and also provides some in-depth discussion of the various ways of interacting with AR. For a full discussion on creating, modifying, and checking archive request files, see section 5.

3.1 starting AR for creating or changing an archive request

For developing an archive request, AR will often be started from the keyboard, with no archive requests open. Although `~gta/apple/ar` is shown as the directory in these examples, the directory would normally be chosen by the developer.

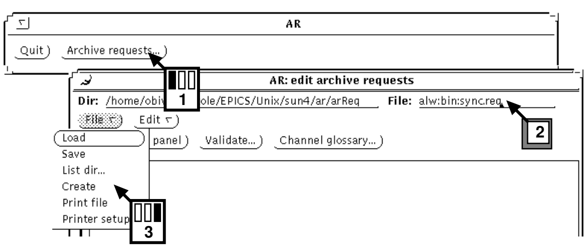
3.2 starting AR with an archive request automatically opened

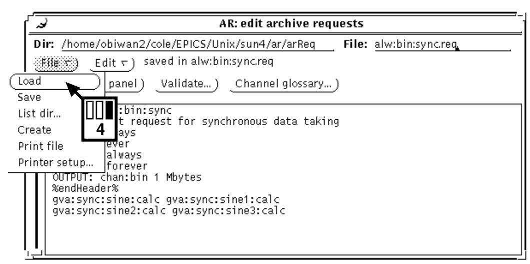
It is often useful, especially during operations, to start AR with an archive request already opened, so that the operator doesn't need to take special action to start AR. The commands shown here would usually be done in a script.



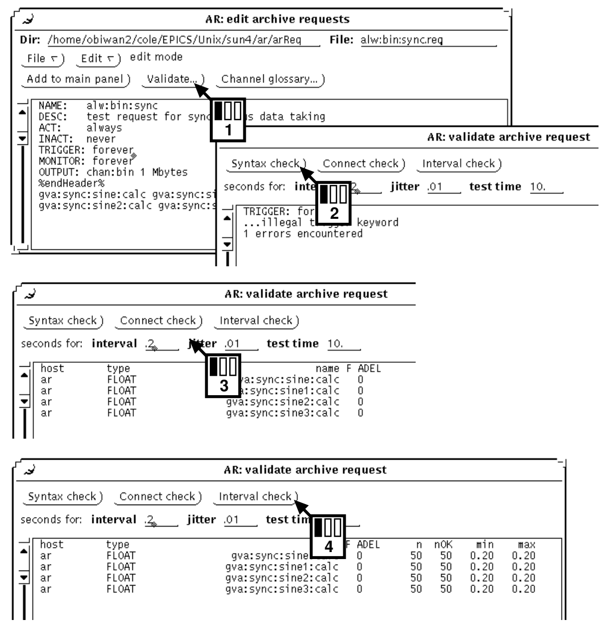
3.3 creating an archive request

To create a new archive request, use steps ***1*** to ***3*** below. Once the lines have been entered, they can be saved with step ***4*** at the right. (See also sec. 5.1.)





3.4 testing an archive request



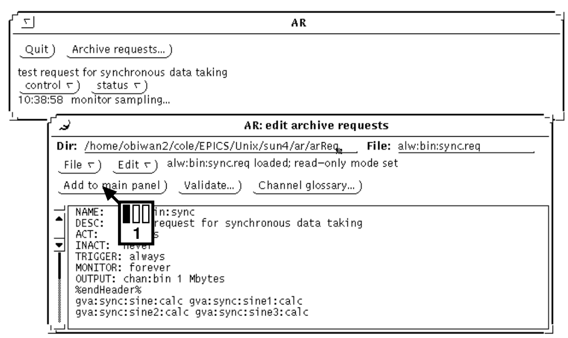
After changing an archive request, test for errors with ***Validate.../Syntax check***. ***Syntax check*** (step ***2***) can be used repeatedly, without needing to ***Save*** between tests.

Connect check (step ***3***) allows testing to see if the channels can be found. It also shows what the archiver deadband is.

Interval check (step ***4***) tests the behavior of synchronous channels, to see if they produce values at the desired rate. Samples are taken for ***test time*** seconds. At the end of the test, the number of samples obtained is printed under ***n***. ***nOK*** shows the number of samples where the time between samples was ***interval*** + or - ***jitter***. ***min*** and ***max*** are the smallest and largest times observed between samples.

See section 5.6 for more details.

3.5 opening an archive request

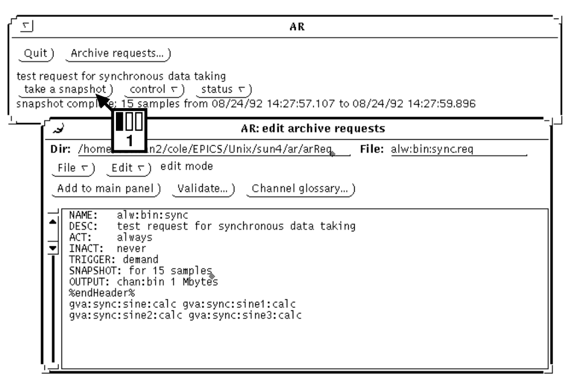


Before any samples can be taken with an archive request, it must be added to the AR main panel, as with step ***1*** at the right. Doing this 'opens' the archive request.

When an archive request is opened, connections are made to the IOC, and the ***ACT:*** condition is checked. Only when the ***ACT:*** condition is satisfied is the ***TRIGGER:*** condition checked.

A 'lock' is placed on an archive request when it is opened, to prevent it from being opened twice and attempting to write multiple copies of the data to the archive data file.

3.6 taking a snapshot



Only after an archive request has been activated can sampling occur. In section 3.5, with ***TRIGGER: always***, sampling began as soon as activation occurred. With ***TRIGGER: demand***, as at right, an additional button appears on the AR front panel. ***click left*** on this button, as in step ***1***, to trigger the sample. When a snapshot is complete, its time range and the number of samples are shown.

3.7 closing an archive request

An archive request is closed automatically when the ***Quit*** button is used. It can also be closed with ***control/close***. This removes it from the AR front panel.

3.8 status and controls on the AR main panel

Quit closes all open archive requests and exits from AR. If an archive request has been modified in the 'edit archive requests' window, but it hasn't been saved, then AR won't exit.

Archive requests... brings up the 'edit archive requests' window

description the text from the ***DESC:*** line is displayed immediately above the ***control*** and ***status*** buttons.

take a snapshot triggers a snapshot. This button appears only if a request has ***TRIGGER: demand*** and uses ***SNAPSHOT*** sampling.

start monitoring triggers the start of monitoring. This button appears only if a request has ***TRIGGER: demand*** and uses ***MONITOR*** sampling.

start stepping starts a stepping sequence. This button appears only if a request has specified ***STEP:***.

control allows closing an archive request and allows interactions for ***demand*** items.

- ***activate*** activates a request that has ***ACT: demand***. ('Grayed out' otherwise.)
- ***inactivate*** inactivates a request that has ***INACT: demand***. ('Grayed out' otherwise.)
- ***start sampling*** triggers sampling for a request that has ***TRIGGER: demand***. ('Grayed out' otherwise.)
- ***close*** closes an archive request.

status allows seeing information associated with an archive request.

- ***show state*** shows whether an archive request is activated, inactivated, or sampling.
- ***show sampling state*** if sampling is in progress, shows how many values have been obtained and the time stamp range of the values.
- ***show IOC connections*** shows how many channels are connected to an IOC and how many aren't connected.
- ***edit...*** brings up the 'edit archive requests' window with the archive request file in it. (If another archive request is presently being edited, then the window will show the file that is being edited.)
- ***printing...*** provides a textual display of the most recent snapshot, similar to the print button in ARR. If the sample type is ***MONITOR***, then this menu item is 'grayed out'. If the archiver is doing any other sampling, then it won't honor the print request.
- ***plotting...*** provides graphical display of the most recent snapshot, with restrictions similar to the ***printing...*** menu item.

status line shows the most recent status message relating to the archive request.

3.9 command line options

AR allows specifying command line options to customize how AR starts running. This is most frequently used to start with one or more archive requests already open, so that operators can be given 'turn key' archiving.

`-I` configFile specifies the path to a configuration file (see sec. 3.10)
`-R` reqFile specifies the path to an archive request to be opened at startup
`-E` reqFile specifies the path to an archive request to be edited; the edit window will be open to the file at startup
`-Dret` dir changes the default directory containing archive requests (the default if this option isn't used is `./arReq`)

3.10 configuration file

The appearance of the AR main window and the 'edit archive requests' window can be configured using commands in a file. This is usually used to add information to the AR main window or the 'edit archive requests' window, or to eliminate unwanted items from the windows.

By default, the main window will have an ***Archive requests...*** button and the 'edit archive requests' window will have a ***Channel glossary...*** button. The latter assumes a glossary file in `../log/channel.glossary`.

If a configuration file is used, then the defaults are ignored and all desired options must be specified in the file. Commands that can be used in a configuration file are:

`MAIN ReqEdit` adds the ***Archive requests...*** button
`MAIN NEWLINE` starts a new line
`MAIN X` pos sets the X position on the line to 'pos'
`MAIN STRING` string places 'string' on the current line
`GLOSSARY` dir file identifies the directory and file containing the channel glossary
`EDIT Glossary` causes a 'glossary' button to be added to the edit window
`Dret` dir sets 'dir' as the directory containing archive requests
`REQ` reqFile causes 'reqFile' to be opened at startup

4 ARCHIVING REAL-TIME DATA WITH AR_cmd

AR_cmd is the command-based version of the EPICS archiver, for use from the keyboard or in a shell script. AR_cmd can run interactively or as a server. The prompt is `AR_cmd:` .

4.1 around-the-clock archiving using AR_cmd in server mode

One frequent reason to use AR_cmd is to provide around-the-clock archiving, especially in situations where the usual operator workstation may be powered down occasionally. In these cases, AR_cmd is started in 'server' mode on a workstation that is always left running.

Only one AR_cmd can be running in server mode on a workstation. Attempting to start a second copy will result in the "address already in use" message.

Use the following steps to set up for around-the-clock archiving:

1. set up an archive request using AR, and test it;
2. close the archive request in AR;
3. edit `./arReq/open.dat` and add the name of the archive request (see sec. 4.7);
4. start AR_cmd on the desired host, as described below. Or, if AR_cmd is already running in 'server' mode on the desired host, use AR_cmd in 'client' mode to open the request, as described in sec. 4.2;
5. if it is ever desired to stop AR_cmd running in server mode, use AR_cmd in 'client' mode to send it a `quit` command.

When AR_cmd is started as a server, it reads `./arReq/open.dat` and opens all the archive requests named in that file.

Use any editor (vi is shown in this example) to update `./arReq/open.dat` (see sec. 4.7).

```
% rlogin desiredHost
Password:
% cd ~gta/apple/ar
% vi arReq/open.dat
     add lines for requests, as needed
% AR_cmd server &
% logout
```

4.2 using 'client' mode to check status of AR_cmd 'server'

To interact with the archiver server, start your own copy of AR_cmd in client mode. The two will 'converse', and you can control the actions of the server. (You do not need to rlogin to the host that the server is running on.)

```
% cd ~gta/apple/ar
% AR_cmd desiredHost
 AR_cmd: status
 AR_cmd:    any other commands
 AR_cmd: disconnect
```

4.3 using 'client' mode to open a new archive request

To get the archiver server to open a new archive request, start AR_cmd in client mode.

```
% cd ~gta/apple/ar
% AR_cmd desiredHost
 AR_cmd: open snap1c
 AR_cmd: disconnect
```

4.4 using 'client' mode to stop the AR_cmd 'server'

To stop the archiver server, start AR_cmd in client mode and use the `quit` command. All presently open archive requests will be closed and the server will stop running.

```
% cd ~gta/apple/ar
% AR_cmd desiredHost
 AR_cmd: quit
stop the server (y/n)? y
%
```

4.5 using AR_cmd in interactive mode

AR_cmd can also be used in interactive mode. This may be from a script or from the keyboard. Interactive mode can be useful when not running in an X Window System environment and when implementing 'batch' archiving using a script.

```
% cd ~gta/apple/ar
% AR_cmd
 AR_cmd: open snap1c
 AR_cmd: sample
 AR_cmd: ^D
%
```

4.6 AR_cmd commands

`activate` reqName activates an archive request (must be `ACT: demand`)
`close` reqName closes an archive request
`close all` closes all presently open archive requests
`disconnect` when running as a client, closes the connection to the server
`help` [topic] prints some help information
`inactivate` reqName inactivates an archive request (must be `INACT: demand`)
`open` reqName opens an archive request
`open all` opens all archive requests named in `./arReq/open.dat`
`quit` or `^D` exits (interactive mode); or stops the server (client mode)
`sample` reqName triggers sampling (must be `TRIGGER: demand`)
`startStepping` reqName starts a stepping sequence (request must have `STEP:`)
`status` prints the status of all presently open archive requests

4.7 arReq/open.dat

When starting AR_cmd in server mode or when using the `open all` command, `./arReq/open.dat` is read and the archive requests named there are opened. This file contains the names of requests (<u>not</u> file names). E.g.,

```
% cd ~gta/apple/ar/arReq
% cat open.dat
   exp1c
   utilities
%
```

5 HANDLING ARCHIVE REQUESTS

The typical method for modifying and/or creating archive request files is to use AR. Any editor can be used, however. Using AR has the advantage that AR provides not only for editing, but also for checking: syntax; whether channels exist; deadbands for analog channels; and 'behavior' of synchronous channels.

5.1 creating an archive request file using AR

In the main AR window, ***Archive requests...***; type new file name for ***File:***; ***File/Create***; type the lines for the file; ***File/Save***.

5.2 loading an archive request file into AR for editing

In the main AR window, ***Archive requests...***. Method 1: Type file name in ***File:***; ***File/Load***. Method 2: ***File/List dir ...***; ***drag left*** on line for file; ***File/Load***. Method 3: On the command line for bringing up AR, use `-E arReq/name.req` as an option.

5.3 copying an existing archive request file using AR

Load the existing file (see sec. 5.2), then, in the 'edit archive requests' window: type new name in ***File:***; ***File/Create***; change the file; ***File/Save***.

5.4 editing archive request files using AR

Load the existing file (see sec. 5.2), or use ***status/edit...*** if the file is presently opened. Then, in the 'edit archive requests' window: ***Edit/set edit mode***; make the changes; test the changes (sec. 5.6), if desired; ***File/Save***. Revert back to the original file with ***Edit/discard edits***.

5.5 using the channel glossary

The channel glossary that is maintained with the GTA Operator Window is available from the 'edit archive requests' window. It is strongly encouraged, when possible, to use the channel glossary as the basis for channel names in an archive request. Copy lines, including comments, from the channel glossary into an archive request. The glossary also indicates, with an '`F`', channels that aren't beam synchronous.

5.6 testing archive requests using AR

Use ***Validate...*** in the 'edit archive requests' window to bring up the validate window. **NOTE:** testing can be done without having done ***File/Save***.

Syntax check checks for errors in the archive request.
Connect check checks to see that all the channels mentioned in the archive request can be found. Searches are issued for all channels and then ***test time*** seconds later the information prints the type and ADEL for each channel. Type will be `TYPENOTCON` for channels that weren't found.
Interval check checks to see how often values are received from the channels. The channels are monitored for ***test time*** seconds and the interval between time stamps is measured. An interval is 'OK' if it is within ***jitter*** seconds of ***interval***. The number of values received is in the ***n*** column; the count of 'OK' intervals is under ***nOK***; ***min*** lists the smallest interval found; and ***max*** has the largest interval.

5.7 archive request file: commands, syntax, and run-time behavior

An archive request consists of:

1. comment lines, beginning with '#'
2. commands, beginning with a 'KEYWORD:' (A blank must follow the ':'.)
3. an 'end%header' line
4. lines with channel names.

5.7.1 'tests' in archive requests

In order for `when` tests to be satisfied, the condition must first be false, then true—i.e., `when` detects an 'edge'. The `if` and `until` tests just check the present value of the channel.

All of the tests are based on monitoring the the value of the channel using the `.ADEL` field. Thus, if the value of a channel doesn't change more than the `.ADEL` field, then the results of a test won't change.

In the tests where a 'value' is specified and the channel is of type `DBF_ENUM`, then 'value' must be one the state strings for the channel.

`if` chanName cond value
`until` chanName cond value
`when` chanName cond value `[every` n `times]`
`when` chanName `change [every` n `times]`

`cond` can be one of: `==   !=   <   <= > >=`

5.7.2 comments

`# comment` anything following the '#' character is treated as a comment

5.7.3 NAME:

The string from this command is used if no output file name is specified (see sec. 5.7.12).
`NAME:` string
Proper function of various archiving features dictate that the name agree with the name of the file. E.g., if the file is named `exp1c.req`, then the command would be:
`NAME: exp1c`

5.7.4 DESC:

The string from this command appears above the button on the main archive window.
`DESC:` string

5.7.5 ACT:

Controls when an archive request is activated. The action taken when the request is activated depends on the type of sampling. For `SNAPSHOT` sampling, monitors on the channels are established immediately, so that input buffers will generally have some data when a sample trigger occurs. For `MONITOR` sampling, however, monitors aren't established until the sample trigger occurs; this usually results in some missed samples between the trigger and the actual start of archiving.

`ACT: always` activate as soon as the request is opened
`ACT: allow disconnects` allows the archive request to be activated even if some of its channels aren't connected
`ACT: demand` activate when operator clicks ***control/activate*** in AR or when the '`activate` reqname' command is used in AR_cmd
`ACT: every` hh:mm:ss activate periodically
`ACT: if` chanName condition activate any time the test is true
`ACT: when` chanName condition activate when the test changes from false to true

5.7.6 INACT:

Controls when an archive request is inactivated. When the request is inactivated, the action taken depends on the type of sampling. For `SNAPSHOT` sampling, monitors on the channels continue, so that input buffers will generally have some data when a subsequent activation occurs. For `MONITOR` sampling, however, monitors are stopped.

`INACT: after` hh:mm:ss inactivate after a delay following activation
`INACT: demand` inactivate when operator clicks ***control/inactivate*** in AR or when the '`inactivate` reqname' command is used in AR_cmd
`INACT: if` chanName condition inactivate any time the test is true
`INACT: never` inactivate only when request is closed
`INACT: when` chanName condition inactivate when the test goes from false to true

5.7.7 TRIGGER:

Controls when sampling begins. (The request must be activated.)

`TRIGGER: after each step` start sampling after each step of the `STEP:` command
`TRIGGER: always` start sampling as soon as request is activated
`TRIGGER: at least every` hh:mm:ss make sure a sample is taken at minimum intervals. This is useful with `demand`, `if`, and `when` triggering.
`TRIGGER: demand` start sampling when operator clicks ***control/start sampling*** in AR or when the '`sample` reqname' command is used in AR_cmd
`TRIGGER: every` hh:mm:ss start a sample on a periodic basis
`TRIGGER: if` chanName condition start a sample any time the test is true
`TRIGGER: when` chanName condition [`every` n `times`] start a sample when the test changes from false to true (or every nth time it changes from false to true)

5.7.8 MONITOR:

Controls monitoring activities. Archiving by monitoring saves data as time-stamped streams of values, one stream per channel. Either `forever`, `for`, or `until` must be specified. Monitoring of channels doesn't actually begin until a sample trigger occurs, so there is usually a slight delay between the trigger and the actual begin of archiving.

`MONITOR: forever` once sampling begins, continue until request is inactivated
`MONITOR: for` hh:mm:ss continue sampling for the specified time
`MONITOR: round stamps` n `msec` round time stamps to nearest 'n' milli-seconds
`MONITOR: until` chanName condition continue sampling until the test is true

5.7.9 GRAB:

Controls grab sample activities. Archiving by grab samples can be used by itself or in conjunction with `MONITOR:` and `SNAPSHOT:` archiving.

`GRAB: every` hh:mm:ss grabs a sample on a periodic basis

5.7.10 SNAPSHOT:

Controls snapshot activities. A single snapshot is taken in response to a single sample trigger. A snapshot is made up of 1 or more time-stamped synchronous samples. The maximum number of samples in a snapshot must always be specified.

With snapshot sampling, monitors are placed on channels as soon as the archive request is activated and the monitors remain active until the archive request is closed. This eliminates the risk of missing some data between the time a sample trigger occurs and the time archiving actually starts. It also makes it possible to do pre-trigger sampling.

`SNAPSHOT: allow partial samples` allow partial samples in the snapshot; if this command isn't used, partial samples are discarded

`SNAPSHOT: every` hh:mm:ss keep only part of the samples, based on time (e.g., keep the samples at 10 sec intervals)

`SNAPSHOT: every` n `samples` keep only part of the samples, based on count (e.g., keep every 10th sample)

`SNAPSHOT: for` n `samples` snapshot size, based on count

`SNAPSHOT: for` hh:mm:ss`,` n `samples maximum` snapshot duration, with maximum sample count of 'n'

`SNAPSHOT: round stamps` n `msec` round time stamps to nearest 'n' milli-seconds

`SNAPSHOT: start` hh:mm:ss `before trigger` pre-trigger sampling, based on time (e.g., start sampling 5 seconds prior to the sample trigger)

`SNAPSHOT: start` n `samples before trigger` pre-trigger sampling, based on count (e.g., start sampling 10 samples prior to the sample trigger)

`SNAPSHOT: until` chanName condition`,` n `samples maximum` put samples into snapshot until the test is true or the maximum number is obtained

5.7.11 STEP:

Allows changing the value of a channel in a predetermined way. When the operator clicks ***start stepping*** in AR or when the '`startStepping` reqname' command is used in AR_cmd, the channel is initialized to 'startValue'. The channel is then changed by 'stepSize' until the value has passed 'endValue'. After initialization and after each step, the archiver will verify that the change was completed before doing the next step. If a delay is specified, then it will be done after verifying that the change is complete.

If `STEP:` is used with '`TRIGGER: after each step`', then a sample trigger will be issued after verifying that the change is complete (or after the optional delay). The next step won't be done until sampling is complete.

`STEP:` chanName `from` startValue `to` endValue `by` stepSize

`STEP: delay` n `seconds` specifies a timed delay following changing the channel's value

5.7.12 OUTPUT:

Controls where archived data is saved. If no file name is specified, then the path depends on the output type and the string from the `NAME:` command. If the file doesn't exist, it is created. If it does exist, then new data is written at the end.

`OUTPUT: chan:bin` [pathName] n `Mbytes` output goes to a 'by channel' archive file. The default path is `./arChan/`string`.bin`, where 'string' is from the `NAME:` command. 'n' specifies the maximum size for the file, in megabytes. The archiver buffers data in memory when doing 'by channel archiving, and writes the data to the file either when the buffers are full or every 15 seconds. Thus, there will usually be a delay in being able to retrieve data 'on the fly' with ARR. (See Sec. 2.4 for a discussion of 'by channel' archive files.)

`OUTPUT: set:txt` [pathName] output goes to a 'sample set' archive file. The default path is `./arSet/`string`.txt`, where 'string' is from the `NAME:` command. (See Sec. 2.5 for a discussion of 'by channel' archive files.)

NOTE Sample type of `MONITOR` can't be used with output type of `set:txt`.

5.7.13 %endHeader%

Marks the end of the commands and the beginning of the channel names. Comments can be included as separate lines or as information following the last channel name on a line.

An optional '`F`' in front of a channel name marks it as a 'filled' channel—i.e., one that isn't beam-synchronous. If several filled channels are on the same line, each must have its own 'F'.

`%endHeader%`
[`F`] chanName [`F`] chanName
[`F`] chanName

5.8 example archive request files

5.8.1 snapshot on demand

```
NAME:        exampleSyncSnap
DESC:        an example snapshot request for sync snapshots
# This request is always waiting for the operator to 'demand' a
# sample; when the operator presses the button, a sample is taken.
# The sample consists of 30 seconds of data, or 100 samples max.
# Comments can be added later to explain details about the data.
ACT:         always
INACT:       never
TRIGGER:     demand
SNAPSHOT:    allow partial samples
SNAPSHOT:    for 0:0:30, 100 samples maximum
OUTPUT:      chan:bin arChan/92_07_16:syncSnap.bin 10 Mbytes
%endHeader%
F R01:AMP_SET_POINT         # rf amplitude setpoint; RFQ
RQ1_RFL_001                 # rf cavity power, loop 1; RFQ
RQ1_RFL_002                 # rf cavity power, loop 2; RFQ
```

5.8.2 monitor channels for any changes in value

```
NAME:        exampleMon
DESC:        an example monitor request
# Monitor the channels for changes in value.  The ‘F’ on the
# first channel doesn’t affect operations when monitoring, but
# it is useful for validating the archive request, and also when
# using the archive request file as the basis for retrievals.
ACT:         always
INACT:       never
TRIGGER:     always
MONITOR:     forever
OUTPUT:      chan:bin arChan/syncSnap.bin 10 Mbytes
%endHeader%
F R01:AMP_SET_POINT       # rf amplitude setpoint; RFQ
RQ1_RFL_001               # rf cavity power, loop 1; RFQ
RQ1_RFL_002               # rf cavity power, loop 2; RFQ
```

5.8.3 periodic synchronous snapshot

```
NAME:examplePeriodic
DESC:an example periodic snapshotrequest
# Save a synchronous snapshot every 10 seconds.  The snapshot
# will consist of a single sample.
ACT:always
INACT:never
TRIGGER:every 0:0:10
SNAPSHOT:for 1 samples
OUTPUT:set:txt arSet/periodicRf.txt
%endHeader%
F R01:AMP_SET_POINT # rf amplitude setpoint; RFQ
RQ1_RFL_001# rf cavity power, loop 1; RFQ
RQ1_RFL_002# rf cavity power, loop 2; RFQ
```

6 RETRIEVING ARCHIVED DATA USING ARR

ARR is the mouse-and-menu version of the EPICS archive retrieval program.

Most commonly, ARR will be started from a button on the 'GTA Operator Window' (see sec. 9). This eliminates the need for changing directories when using ARR, since each application area can have its own item in the 'Archive retrieval' menu.

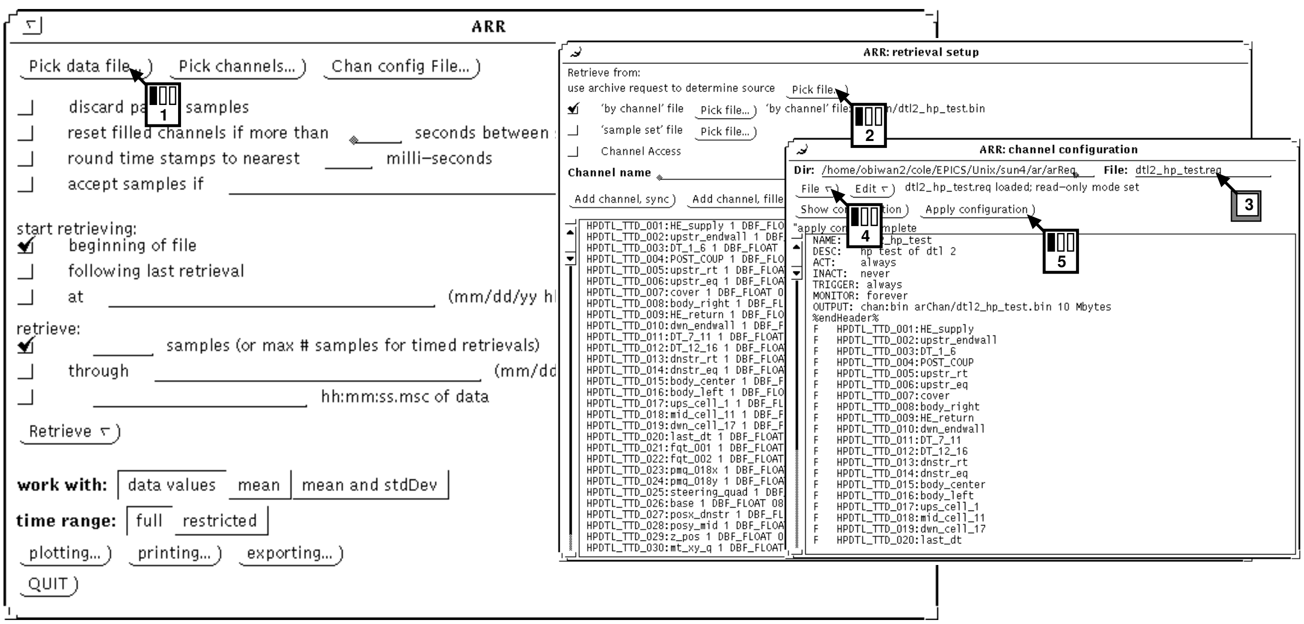
6.1 special cases

`.RVAL` channels are given special handling for printing, plotting, and exporting (except to 'sample set' files). By default, these channels are treated as DVX2502 channels, which are unsigned integers.

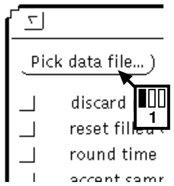
6.2 retrieving data from archive data files

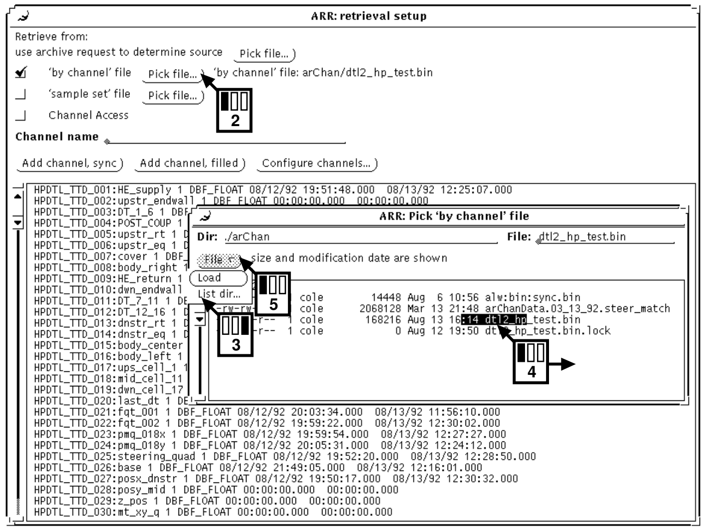
6.2.1 choosing an archive data file

Method 1, uses information from an archive request file to identify both the data file and the channels to retrieve. This is shown in steps ***1*** to ***5*** in the figure below. The archive request identifies the type of file, its name, and the channels.



Method 2 directly selects the data file using steps ***1*** to ***5***. This example assumes the data is in a ‘by channel’ file.





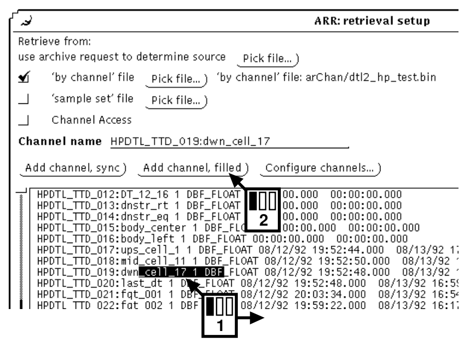
6.2.2 selecting channels to retrieve

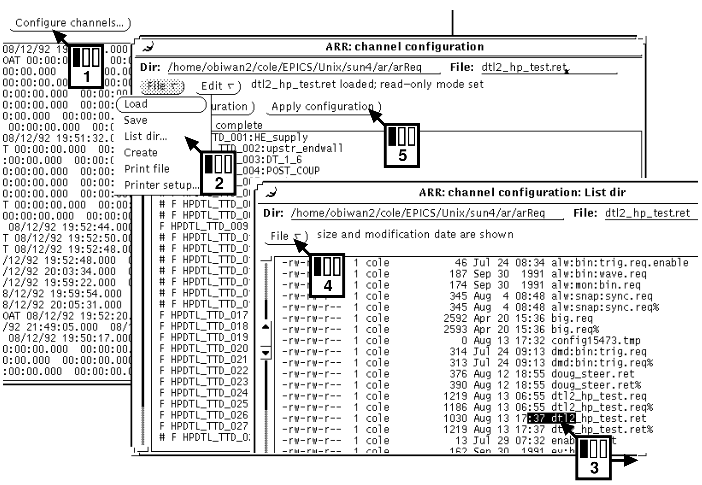
Method 1, using an archive request file, results in the automatic selection of all channels in the file. (See ‘method 1’ in the preceding section.)

Method 2 chooses channels directly from the list of channels in the data file. This method requires knowledge of whether a channel is ‘synchronous’ or ‘filled’.

Repeatedly perform steps ***1*** and ***2*** at the right until all desired channels have been selected. Use the button appropriate to the channel’s type (see sec. 1.5).

As an alternative, a name can be typed directly in the ***Channel name*** field. The name can use ‘`*`’ and ‘`?`’ as wildcard characters. E.g., `HPDTL_TTD_010...` through `HPDTL_TTD_019...` are selected by `*TTD_01*`.





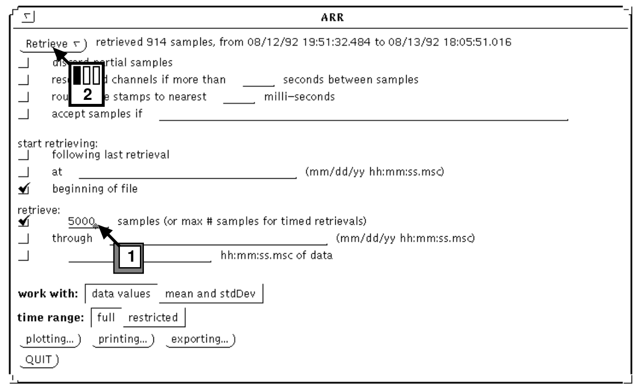
Method 3 uses the list of channels stored in a ‘retrieval configuration file’.

See section 7 for more information on creating and updating retrieval configuration files.

6.2.3 retrieving samples with the main ARR window

The act of retrieving produces a set of synchronous samples. This set of samples can be plotted, printed, and exported in various ways. The working parameters can be changed and the set can be re-plotted, re-printed, etc.

For a simple retrieval, use steps ***1*** and ***2*** at the right. You'll see the time range for the samples that were retrieved. Regardless of which ***retrieve:*** option you use, you must specify the maximum number of samples.



For synchronous data, use ***discard partial samples***. To intermix synchronous and filled channels, use ***round time stamps to nearest***, specifying your system trigger time (e.g., 200 milli-seconds). To 'weed out' marginal data, use ***accept samples if***.

6.2.4 status and controls on the main ARR window

Retrieve causes data to be retrieved and is used to bring up other windows that affect retrievals. The number of samples retrieved and the time range of the samples is shown. When ARR is first started, ***click left*** on ***Retrieve*** brings up the 'retrieval setup' window. Thereafter, ***click left*** retrieves samples.

Retrieve/resync resynchronizes with an active 'by channel' archive data file, so that a retrieval can access the most up-to-date data. This is useful to look at data as it is being archived.

Retrieve/Stop stops the acquiring of samples when Channel Access is the source of data.

Retrieve/setup... brings up the 'retrieval setup' window. This window is used for selecting the source of data to be retrieved and for selecting which channels to use.

Retrieve/configure... brings up the 'channel configuration' window, which allows more versatile ways of selecting channels. It supports saving a list of channel to a file, so that it can easily be used later. The list can be edited. In addition, this window allows using the actual archive request file as the basis for setting up a retrieval. Finally, the archive request file can be edited so that annotations can be added as knowledge is gained about the data.

discard partial samples causes partial samples to be ignored. A partial sample can occur when a synchronous channel doesn't have a value at a particular time stamp. If a non-synchronous channel isn't declared as 'filled', it can cause partial samples. When this option is used, the ***Retrieve*** status message shows how many partial samples were discarded.

reset filled channels is obsolete

round time stamps causes time stamps to be rounded so that there is some leeway in determining time-correlation. This is needed when using 'software synchronized' IOCs, since their time-of-day clocks can drift. The usual practice is to use a value of the system trigger interval. E.g., for a 5 Hz system trigger, use 200 milli-seconds.

accept samples if allows testing the value of one of the channels in the sample to see if the sample should be used. For example,

accept samples if `RQ1_TOR_002:CURRENT > 5`

will use a sample only if 'the beam is on'. This is especially useful for eliminating 'bogus' samples from the computation of mean and standard deviation.

start retrieving: controls where a retrieval begins:

- ***following last retrieval*** causes retrieving to start immediately after the prior retrieval left off.
- ***at*** ____ allows specifying a date and time to start retrieving. (See section 11 for the allowed formats of date and time.)
- ***beginning of file*** causes retrieving to start at the beginning of the file.

retrieve: controls how many samples will be retrieved:

- ____ ***samples*** specifies the maximum number of samples to retrieve. A number must always be specified, since ARR uses this to allocate memory for a sample set. This can be greater than the number of samples in the file. A realistic number should be used; for example, specifying 100 million would usually exceed the computer's memory space.
- ***through*** ____ allows specifying a data and time to stop retrieving. This is most often used in conjunction with ***start retrieving at:***. (See section 11 for the allowed formats of date and time.)
- ____ ***hh:mm:ss*** retrieves data for a time interval. This could be used, for example, to step through a file 15 minutes at a time.

work with: controls processing of data (printing, plotting, and exporting) <u>after</u> it is retrieved:

- ***data values*** use the actual data values.
- ***mean*** use the means of the snapshots in the sample set. This is similar to the following, except that for exporting and printing the standard deviation isn't displayed.
- ***mean and stdDev*** use the means and standard deviations of the snapshots in the sample set. The mean and standard deviation for a channel is computed for each snapshot in the sample set. (If the file isn't composed of snapshots, then each archive request 'activation' begins a new 'pseudo-snapshot'.) The population size is the number of samples in the snapshot. Missing values are treated as 0. (but ***mean and stdDev*** should probably be used only with ***discard partial samples***).

time range: controls the subset of data that is process

- ***full*** use the full set of samples from the retrieval.
- ***restricted*** restricts processing to the time range specified in the 'plot setup' window. This allows 'zooming in' on a particular time period. (This option is automatically set when a time range is selected with ***Apply*** in the 'plot setup' window.)

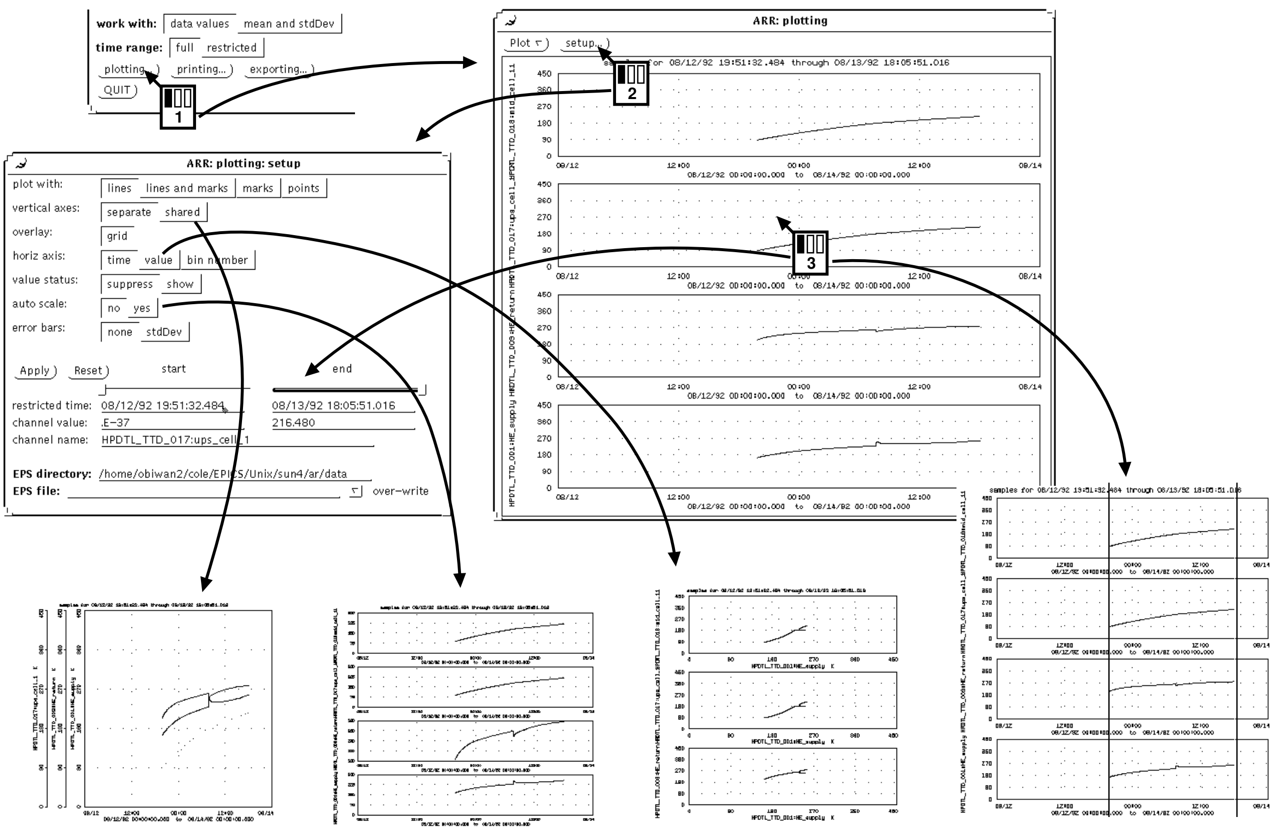
6.3 plotting data following retrieval

ARR plots data in a variety of ways to the screen, to a PostScript printer, and to an Encapsulated PostScript (EPS) file. The ‘plot’ window is brought up with ***plotting...*** on the main ARR window, as shown below with step ***1***.

The plot isn’t automatically redrawn following a new retrieval or a change in set up. Use the ***Plot*** button to draw a new plot.

If it is desired to alter the appearance of a plot, use the ***setup...*** button to bring up the ‘plot setup’ window (step ***2***). After changing setup parameters, use the ***Plot*** button to redraw the plot.

The values of channels at particular times can be viewed in the ‘plot setup’ window with ***click left*** on the plot for a channel, shown as step ***3***.

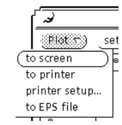


Plotting is primarily oriented to scalar channels. Both numeric and discrete channels are handled. For a discrete channel, annotations use the state strings for the channel.

When an array channel is plotted vs time, only the first element of the array is plotted. Only when using ‘bin number’ for the horizontal axis is the entire array shown.

6.3.1 status and controls on the plot window

Plot/to screen plots the data in the 'plot' window. This window can be resized as desired.



Plot/to printer sends the plot to a PostScript printer. The size of the plot roughly corresponds to the size on the screen.

Plot/printer setup allows selecting the printer to use

Plot/to EPS file sends the plot to the file specified on the 'plot setup' window. The size of the plot roughly corresponds to the size on the screen.

6.3.2 status and controls on the plot setup window

See section 6.3 for a picture of the 'plot setup' window and some examples of variations in plotting.

plot with: controls the plotting of points

- **lines** connects points with lines. (The points are plotted in order by time stamp.) The line isn't drawn between points that have 'missing data' between them. Neither is a line drawn to a point that starts a new run of data, such as the first sample in a snapshot. ***work with: mean and stdDev*** can be used to get lines between snapshots.
- **lines and marks** a combination of **lines** and **marks**
- **marks** plots a geometric symbol at each data point, producing a 'scatter plot'. (The shape that is used for each channel is automatically chosen.) Using **marks** is a good way to see individual data points.
- **points** plots each data point as a pixel-sized dot. This is most useful when the number of data points is large.

vertical axes determines how the channels are plotted

- **separate** each channel is plotted in its own separate plotting area
- **shared** the channels all share the same rectangular plotting area. On color displays, each channel is plotted with its own color; on monochrome media, each is plotted with a dashed line pattern. The choice of color or dashed line pattern is automatic.

overlay: chooses which plotting overlay is used

- **grid** rectangular grid with dotted lines at the major tick intervals.

horiz axis: specifies what is plotted on the horizontal axis

- **time** date and time
- **value** the value of the first channel selected is plotted on the horizontal axis.
- **bin number** the horizontal axis is calibrated from 0 to 'n', where 'n' is the number of samples or the size of the largest waveform. This is the only to plot waveforms. for scalar channels, using this option allows seeing the samples with uniform spacing rather than the spacing indicated by their time stamps.

value status: controls whether the status of individual values is displayed on the plot

- **suppress** the status information is not shown
- **show** the status information is shown by plotting the second character of the status code at each point's coordinate (see section 10)

auto scale: determines how the scaling is done for the channels' axes

- **no** the values at the ends of the axis are taken from LOPR and HOPR for the channel. (For retrievals from 'sample'set files, the minimum and maximum data value are used instead.)
- **yes** scaling and number of intervals is based on minimum and maximum data values
- **yes, shared** scaling and number of intervals is based on the minimum and maximum data values <u>for all channels</u>. This is especially useful for plotting channels on a shared Y axis, so that the relative positions of the lines correspond exactly to relative magnitudes.

error bars: turns off and on the plotting of error bars

- **none** no error bars are plotted
- **stdDev** plots error bars if ***work with: mean and stdDev*** is selected on the main ARR window

Apply applies a new restricted time range for the data, with the current settings for the other options on the 'plot setup' window. The plot is redrawn with the restricted time range. (This also causes the ***time range: restricted*** to be set on the main ARR window, so that printing and exporting will also use the restricted range of samples.)

Reset sets ***time range: full*** on the main ARR window and redraws the plot with the full range of samples.

restricted time: the starting and ending dates and times for the restricted range of processing. These can be set using the sliders, or values can be typed directly in the fields. This time range is used for processing when ***time range: restricted*** is selected on the main ARR window.

channel value: displays the channel's values at the start and end of the restricted time range

channel name: the name of the channel that is displayed. ***click left*** on the plot for a channel selects it for display; the annotation for the selected channel is outlined.

EPS directory: the directory for an Encapsulated PostScript file

EPS file: the name of the EPS file

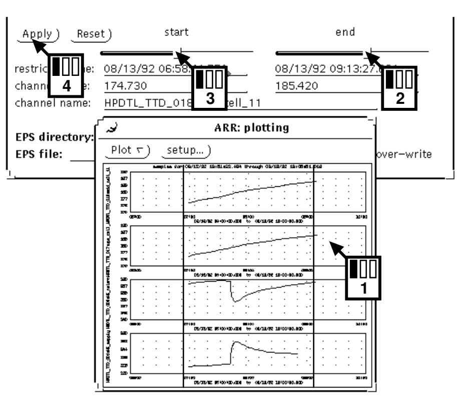
- **append** causes the EPS plot to be added at the end of the file
- **over-write** causes the EPS plot to replace the existing file

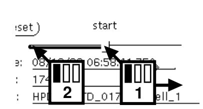
6.4 restricting the time range with the plot setup window

click left on the plot for a channel causes 'time cursors' to appear and the channel's values at the cursors to display.

Move the sliders to restrict the range of samples to be processed.

Use ***Apply*** to see the narrowed range of samples. ***Reset*** goes back to the full range.





drag left on the slider handle (step ***1***) moves the cursor on the plot. ***click left*** on the slider's bar (step ***2***) makes fine adjustments in the slider position.

6.5 printing data following retrieval

ARR prints data to the screen, to a printer, and to a text file. The 'print' window is brought up with ***printing...*** on the main ARR window, as shown below in step ***1***.

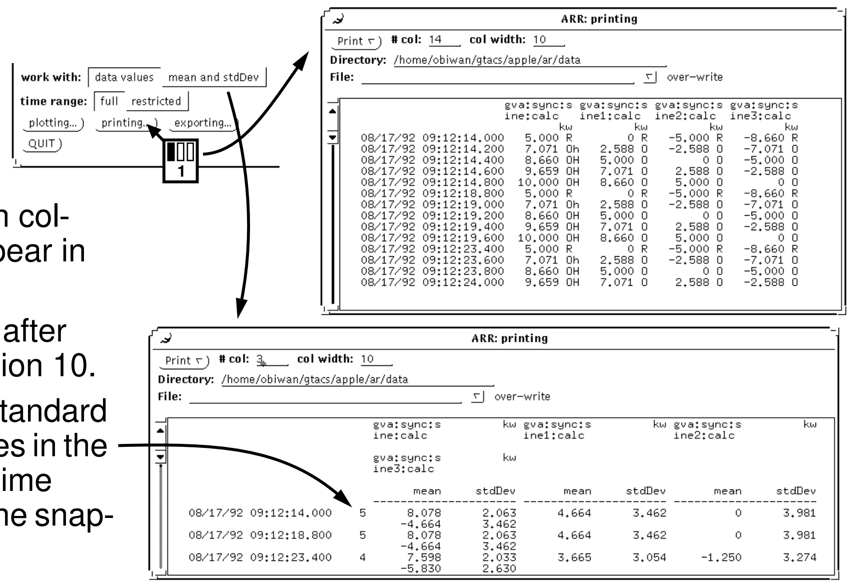
The printout isn't automatically redisplayed following a new retrieval or following a change in ***work with*** or ***time range***. Use the ***Print*** button to print the data again.

The number of columns and the width of a column are controlled by ***#col:*** and ***col width:***. Channel names are 'folded' to fit within a column.

If there are more channels than columns, then additional lines appear in the heading and the data.

The status code that is printed after each value is described in section 10.

When working with mean and standard deviation, the number of samples in the snapshot is also printed. The time stamp is for the first sample in the snapshot.

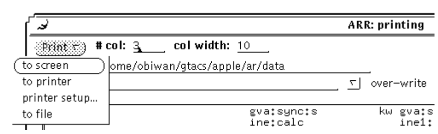


6.5.1 status and controls on the print window

Print/to screen displays the data in the 'print' window. This window can be resized as desired.

Print/to printer sends the text to a printer using the parameters in the 'printer setup' window.

Print/printer setup... brings up the 'printer setup' window to allow selecting the printer to use, paper orientation, and font size.



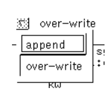
Print/to file sends the text to a file

 # col: sets the number of channels that will appear on a line. If there are more channels than specified, then several lines will be used for a sample.

col width: specifies how wide a column is. A width of 2 is given special treatment—only the status flag is printed; this is useful for diagnosing problems with large numbers of synchronous channels.

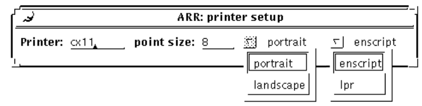
Directory: the directory for the file when using ***Print/to file***

File: the name of the file when using ***Print/to file***



append/over-write controls whether the data is written at the end of the file (preserving its previous contents) or whether the new data replaces the old (which is the default).

6.5.2 status and controls on the printer setup window



Printer: selects the printer to use. Changing the printer name here also changes it for the 'printer setup' window for plotting. The default name is taken from the `PRINTER` environment variable.

point size: sets the font size. This is used both in the 'printer' window and for the printer, when **enscript** is used.

portrait/landscape sets the page orientation for the printer, when **enscript** is used.

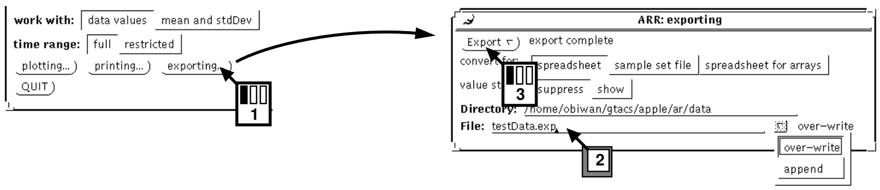
enscript/lpr controls whether `enscript` or `lpr` is used to send output to the printer. Only when `enscript` is used will **point size:** and **landscape** be used.

6.6 exporting data following retrieval

ARR can export data to files in a variety of ways. Exporting is controlled by the ‘export’ window, which is brought up with ***exporting...*** on the main ARR window, as shown in step ***1*** below.

The main purpose for exporting data is to allow further analysis of data by other packages, such as commercial spreadsheet programs.

To export data for processing with a spreadsheet program, use steps ***1*** to ***3*** below.



6.6.1 status and controls on the export window

See section 6.6 for a picture of the ‘export’ window.

Export/to file sends the data to the file.

convert for: selects the type of export

spreadsheet produces a tab-delimited file with one line per time-stamp. This is suitable for use with spreadsheets such as Excel and Lotus 1-2-3. This is intended for scalar channels; if array channels are present, then only the first element in the array is exported.

sample set file writes an archiver ‘sample set’ data file. This allows reading the data back into ARR for further processing.

spreadsheet for arrays is similar to **spreadsheet**, but a ‘count’ column is written indicating how many elements exist for the channel.

value status: controls whether the status of individual values is written to the file

suppress the status information isn’t exported

show an extra column is written for each channel, containing the one or two character status code (see sec. 10)

Directory: the directory for the file

File: the name of the file

append/over-write controls whether the data is written at the end of the file (preserving its previous contents) or whether the new data replaces the old (which is the default).

6.6.2 spreadsheet for data values, without value status

The first line is the ‘reference’ date and time for the data. Lines 2 and 3 are column headings. The actual data starts in the fourth line.

Columns are separated by tabs. The first column is the date and time for the sample. The second column is elapsed time past the ‘reference’ date and time. The following columns contain the values.

```
"08/17/92 09:12:14.000"
"date"                   "time"     "gva:sync:sine:calc"   "gva:sync:sine1:calc"
"stamp"                  "seconds"  "kw"                   "kw"
"08/17/92 09:12:14.000"  0.000      5.000                  0
"08/17/92 09:12:14.200"  0.200      7.071                  2.588
"08/17/92 09:12:14.400"  0.400      8.660                  5.000
"08/17/92 09:12:14.600"  0.600      9.659                  7.071
"08/17/92 09:12:14.800"  0.800      10.000                 8.660
"08/17/92 09:12:18.800"  4.800      5.000                  0
"08/17/92 09:12:19.000"  5.000      7.071                  2.588
"08/17/92 09:12:19.200"  5.200      8.660                  5.000
"08/17/92 09:12:19.400"  5.400      9.659                  7.071
"08/17/92 09:12:19.600"  5.600      10.000                 8.660
"08/17/92 09:12:23.400"  9.400      5.000                  0
"08/17/92 09:12:23.600"  9.600      7.071                  2.588
"08/17/92 09:12:23.800"  9.800      8.660                  5.000
"08/17/92 09:12:24.000"  10.000     9.659                  7.071
```

6.6.3 spreadsheet for mean and standard deviation

Spreadsheet export files for mean and standard deviation are similar to the simple spreadsheet file shown in sec. 6.6.2. The third column contains the number of samples in a snapshot. The following columns contain the mean and standard deviation for each channel.

The column headings can be confusing—the channel’s name and its engineering units appear on the second line in separate columns. The channel ‘owns’ both columns and the engineering units apply to both the mean and the standard deviation.

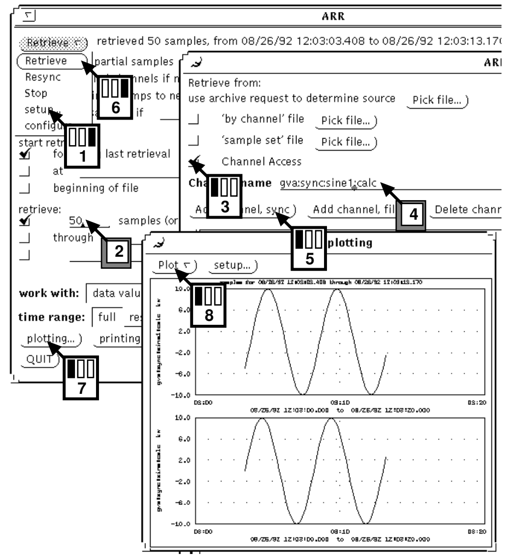
```
"08/17/92 09:12:14.000"
"date"                  "time"    "sample" "gva:sync:sine:calc" "kw"     "gva:sync:sine1:calc" "kw"
"stamp"                 "seconds" "count"  "mean"               "stdDev" "mean"                "stdDev"
"08/17/92 09:12:14.000" 0.000     5        8.078                2.063    4.664                 3.462
"08/17/92 09:12:18.800" 4.800     5        8.078                2.063    4.664                 3.462
"08/17/92 09:12:23.400" 9.400     4        7.598                2.033    3.665                 3.054
```

6.6.4 spreadsheet for data values, with value status

These spreadsheet export files are similar to the simple spreadsheet file shown in sec. 6.6.2. A column containing the status code has been added in front of each channel. (Only the first six lines of the file are shown.)

```
"08/17/92 09:12:14.000"
"date"                   "time"    "stat"  "gva:sync:sine:calc" "stat"   "gva:sync:sine1:calc"
"stamp"                  "seconds" "stat"  "kw"                 "stat"   "kw"
"08/17/92 09:12:14.000"  0.000     "R "    5.000                "R "     0
"08/17/92 09:12:14.200"  0.200     "Oh"    7.071                "O "     2.588
"08/17/92 09:12:14.400"  0.400     "OH"    8.660                "O "     5.000
```

6.7 using ARR to monitor real-time data



ARR can also retrieve real-time data directly from Channel Access, eliminating the need for setting up an archive request, starting the archiver, etc. Most operations are the same as when retrieving from an archive data file.

Steps ***1*** to ***7*** demonstrate setting up for acquiring data from 2 synchronous channels, acquiring the data, and plotting it. Steps ***4*** and ***5*** can be used repeatedly to select channels, but wildcard characters can't be used in the ***Channel name*** field.

If the number of samples is large, the ***Plot*** button can be used repeatedly (step ***8***) to see the data as it builds up. Printing works similarly.

To interrupt a Channel Access retrieve before it's done, use ***Retrieve/Stop***.

6.8 command line options

ARR accepts command line options that can be used to customize the initial state. This is particularly useful when starting ARR from the 'GTA OperatorWindow'.

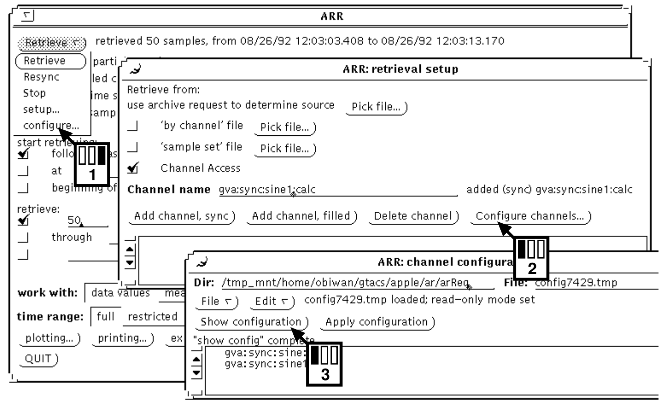
`-Dssf` dir changes the default directory containing 'sample set' archive data files (the default if this option isn't used is `./ar`Set)
`-Dbcf` dir changes the default directory containing 'by channel' archive data files (the default if this option isn't used is `./arChan`)
`-Dret` dir changes the default directory containing 'retrieval setup' files (the default if this option isn't used is `./ar`Req)
`-Dexp` dir changes the default directory containing export data files (the default if this option isn't used is `./`data)

7 WORKING WITH ARR RETRIEVAL CONFIGURATION FILES

Although the ***Add channel*** buttons on the 'ARR retrieval setup' window can be used for selecting channels to retrieve, it is often useful to have a 'canned' list of channels. The 'ARR channel configuration' window supports this need.

The 'channel configuration' window can be brought up with either ***Retrieve/configure...*** on the main ARR window or ***Configure channels...*** on the 'retrieval setup' window (steps ***1*** and ***2*** at the right).

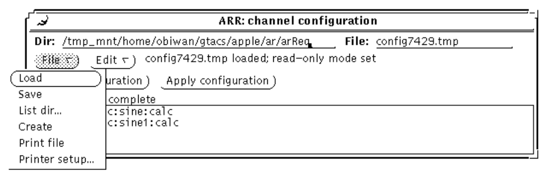
The present configuration can be seen by ***click left*** on ***Show configuration*** (step ***3***). This creates a temporary file that can be edited to alter the configuration. If desired, the information can be stored in a permanent file, as described below.



7.1 status and controls on the ARR channel configuration window

Dir: the directory containing channel configuration files. This is usually `./arReq`.

File: the name of a channel configuration file. By convention, these files have an extension of `.ret`.



File supports several file I/O operations

Load loads the file specified by **Dir:** and **File:**, in read-only mode.

Save saves the edits to disk, changes back to read-only mode, and removes the lock on editing.

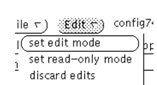
List dir... brings up a separate window with a list of the files in **Dir:**. This window can be used to easily select a channel configuration file. (See section 6.2.2 for an example of using this feature.)

Create creates a new file and stores in it the contents of the window

Print file prints to the printer the file presently being displayed.

Printer setup... allows selecting the printer to use, as well as font size and page orientation. (See section 6.5.2 for more information.)

Edit supports some editing operations



set edit mode changes to editing mode. If the file is writeable and nobody else is editing it, it is locked for editing. If another user has the file locked, an error message appears.

set read-only mode resets editing mode and removes the lock on editing. This can be used only if the file hasn't been altered.

discard edits discards any changes, resets edit mode, and removes the lock on editing.

Show configuration displays the presently selected channels in the window.

Apply configuration deletes the presently selected channels and then selects the channels named in the window.

8 RETRIEVING ARCHIVED DATA USING ARR_cmd

ARR_cmd is the command-based version of the EPICS archiver retrieval program, for use from the keyboard or in a shell script. The prompt is `ARR_cmd:` . Since the capabilities of ARR_cmd parallel those of ARR, only a brief summary of the commands is given here.

8.1 making ARR_cmd read its commands from a file

`source` filePath causes ARR_cmd to read commands from the file. After the last command in the file, ARR_cmd resumes reading commands from the keyboard. A 'source'd file can also use the `source` command.

8.2 selecting an archive data file

`byChannel` [filePath] selects a 'by channel' data file for retrievals; the default path is `./arChan/arChanData.bin`.

`sampleSet` filePath selects a 'sample set' data file for retrievals

`fileInfo` prints information about the channels and time ranges in the file.

8.3 selecting channels and retrieving samples

Channels are selected with the `channel` command. If a channel name is preceded by an '`F`', it treated as 'filled'. Several `channel` commands can be used.

`channel` [`F`] chanName [[`F`] chanName ...] # optional comment

`delete all` deselects all channels
`delete` chanName [chanName ...] deselects particular channels

Retrieving samples is done by specifying 'nSamples' on the commands that have this parameter. If 'nSamples' is larger than the number available, there is no error.

Several commands are available for positioning and special processing:

`accept if` chanName cond value causes samples to be discarded unless they meet the specified test. ('`accept all`' causes all samples to be accepted.)

`discardPartial` causes partial samples to be discarded

`position` [mm/dd/yy] hh:mm:ss positions the file to the specified time stamp; if the date is omitted, then the present date is assumed

`resetFilled` nSeconds resets filled channels after the specified delay

`rewind` positions the file back to the beginning

`roundStamps` nMillisec rounds times to the nearest number of milliseconds

`workWith` {`dataVal` or `mean` or `meanAndStdDev`} determines whether the output commands work with actual data (the default), with snapshot mean, or with snapshot mean and standard deviations.

8.4 plotting

`plot`[,type] [nSamplesl] plot on the screen; ***click right*** on the plot to dismiss it
`plotEPS`[,type] [filePath] plot to an Encapsulated PostScript file
`plotHC`[,type] [printerName] plot (a hard copy) to a PostScript printer

Plot type is specified by the 'type' parameter. If omitted, the previously specified type is used. '`ty`' is the default.

- `ty` specifies time vs value, with separate grids; `tyy` has shared grid
- `xy` specifies value vs value, with separate grids; `xyy` has shared grid
- `y` specifies 'bin number' vs value, with separate grids; `yy` has shared grid

The plot type can be modified with special prefix and suffix characters.

- `l` as a prefix causes values to be connected by lines (this is the default)
- `p` as a prefix causes a pixel-sized point to be plotted for each value
- `m` as a prefix causes a plot mark to be plotted for each value
- `s` as a suffix causes the status code to be plotted for each value
- `e` as a suffix causes error bars to be plotted for means and standard deviations.

8.5 printing

`print` [[nSamples] [,nCol, colWidth]] prints to the screen
`printHC` [[printerName] [,nCol, colWidth]] prints to the printer

'nCol' specifies the number of channels and 'colWidth' specifies how wide each column is. Default is 4 columns, each 12 characters wide.

To send printed output to a file, use the '`dataOut` [filePath]' command.

8.6 exporting

`export`,type [nSamples] exports to the screen.

Export type is specified by the 'type' parameter. If omitted, the previously specified type is used. '`ss`' is the default.

- `ss` specifies exporting for spreadsheet, with tab-delimited format
- `ssStat` same as '`ss`', except value status code is exported

To send exported output to a file, use the '`dataOut` [filePath]' command.

8.7 an example of using ARR_cmd

This example shows retrieving all the samples for several channels from `arChan/snap1c.bin` and exporting means and standard deviations to `data/snap1c.exp`.

```
% cd ~gta/apple/ar
% ARR_cmd
 ARR_cmd: byChannel arChan/snap1c.bin
 ARR_cmd: channel RQ1_TOR_001:CURRENT
 ARR_cmd: channel F I1A_MSL_001:CURRENT
 ARR_cmd: channel F rq1_prx_down:x F rq1_prx_down:y
 ARR_cmd: discardPartial
 ARR_cmd: roundStamps 200
 ARR_cmd: workWith meanAndStdDev
 ARR_cmd: dataOut data/snap1c.exp
 ARR_cmd: export,ss 10000
 ARR_cmd: ^D
%
```

9 GTA OPERATOR WINDOW (gtaOpr) COMMAND LINE

`gtaOpr` can be used to give operators ‘push button’ access to AR and ARR in interactive mode. To configure `gtaOpr`, edit `~gta/apple/gta_ops/src/config.SECURE` on the L-net (or `..../config.OPEN` on the open-net).

The usual approach is to add a new line by copying and customizing an existing one.

9.1 adding a menu item for starting AR

Locate the set of lines ending with: `MENUBUTTON Archiving`. To start AR running in `~gta/apple/ar` with `arReq/snap1c.req` opened:

```
MENUITEM "exp 1c snapshot"       gtaAr ../ar AR -R snap1c.req
```

9.2 adding a menu item for starting ARR

Locate the set of lines ending with: `MENUBUTTON “Archive retrieval”`. To start ARR running in `~gta/apple/lpctb/ar`:

```
MENUITEM “LPCTB retrieval”     gtaProg ../lpctb/ar ../../ar/ARR
```

10 STATUS CODES FOR VALUES

For the ARR print operations, and for the export operations that show channel status, the status of the values is shown with a one or two character code. (If desired, the second character of two character codes will be shown on plots.) The first character indicates a generic status for the value and will be one of:

- `O` the value is ‘okay’. This means only that the control system provided a value for the channel, with the proper time stamp. It does not mean that there are no alarms for the value.
- `R` the value is okay, but it ‘restarts’ the run of data. This will occur, for example, for the first sample following the activation of an archive request.
- `F` the value is okay, but it is ‘filled in’ from a value with a prior time stamp.
- `M` there is no value for the channel with the proper time stamp. The value will appear as either:
  - `no_data` no value available with sample’s time stamp, or
  - `EOF` end of file for that channel
- `S` the value is okay, but it starts a snapshot
- `I` the value is invalid—i.e., its alarm severity was `INVALID`. This indicates that an actual value was received from the IOC, but that it is meaningless.

The second character, when present, conveys additional status information. (For plotting with **value status: show**, these codes will appear on the plot.)

- `blank` the value is ordinary
- `R` hardware read alarm
- `H` high alarm limit violated
- `h` high warning limit violated
- `l` low warning limit violate
- `L` low alarm limit violated
- `D` the channel disconnected
- `E` end of file on archive file
- `I` archive request inactivated
- `m` the value has a minor alarm (other than one listed above) status from the control system
- `M` the value has a major alarm (other than one listed above) status from the control system
- `N` no data at time stamp, for some other reason than listed above

11 TIME STAMPS

In IOC, the time stamp changes at each system trigger. This time stamp is given to any value stored into the database prior to the next system trigger. Two implications of this technique are important. First, all data obtained during a “system trigger interval” will have the same time stamp. The second implication is that it isn’t possible to determine, just by examining its time stamp, exactly when, relative to the system trigger, a value was obtained. Thus it is that synchronous data, as discussed in this manual, is only approximately synchronous.

(There are actually two ‘flavors’ of IOCs—‘hardware synchronized’ and ‘software synchronized’. The discussion above is for hardware synchronized IOCs, whose time stamps are guaranteed to be identical. Software synchronized IOCs, however, can have slightly different notions of time of day—usually within 10 milliseconds of each other and the hardware synchronized IOCs. If time stamp rounding is done when retrieving data from archive data files (see sec. 6.2.4), then the differences between the two types of timing can be safely ignored.)

When ARR displays a time stamp as text, it is generally in the following form:

`mm/dd/yy hh:mm:ss.msc` where

- `mm` is the numeric month (January=1, ... December =12)
- `dd` is the numeric day of the month
- `yy` is the last two digits of the year
- `hh` is the hour of the day, using a 24 hour clock (00, ... 23)
- `mm` is the minute within the hour
- `ss` is the second within the minute
- `msc` is the milli-second within the second

Special treatment is used during the switch from daylight savings time back to standard time. In the USA, this switch occurred in 1991 on October 27 at 2 a.m., local daylight time—at that time, the time became 1 a.m., local standard time. The result was a 2 hour interval during which numeric clock times repeated. For time stamps within this interval, the display of the time stamp indicates what is happening, as shown in the following sequence of time stamps at 30 minute intervals:

`00:30:00.000`	not in the ‘limbo time’, still local daylight time
`01d00:00.000`	just entered the ‘limbo time’, still local daylight time
`01d30:00.000`	in the ‘limbo time’, still local daylight time
`01s00:00.000`	in the ‘limbo time’, just switched to local standard time
`01s30:00.000`	in the ‘limbo time’, local standard time
`02:00:00.000`	just left the ‘limbo time’, local standard time

When a time stamp is entered as part of an ARR command, a date followed by a time must be used. The forms described above are acceptable. Leading zeros can be omitted. The fractional part of a second can be omitted. If the fractional part of a second is specified, however, it must be a multiple of 3 digits in length. For the ‘limbo time’, freedom is allowed on where the ‘s’ and ‘d’ are placed, so that the exact placement doesn’t have to be memorized. The date can appear in two additional forms, with the name (or the first 3 characters) of the month being used: month dd, yyyy or dd-month-yy. The examples below show some possible forms for entering time stamps, some of which are illegal:

- legal `1/2/91 0:02:3` leading zeroes can be dropped
- illegal `1/2/91 :02:3` must have at least one digit
- illegal `1/2/91 1:2:3.4` multiple of 3 digits needed for decimal part
- legal `10/27/91 01s2:3.759`
- legal `Oct 27, 1991 01s2:3.759`