For: wpm
Printed on: Tue, Sep 14, 1993 15:37:28
From book: AR_ARR_UIM
Document: cover
Last saved on: Mon, Dec 9, 1991 08:16:57
Document: toc
Last saved on: Mon, Dec 9, 1991 08:16:58
Document: preamble
Last saved on: Mon, Dec 9, 1991 08:17:08
Document: AR_kbd
Last saved on: Mon, Dec 9, 1991 08:16:58
Document: ARR
Last saved on: Mon, Dec 9, 1991 08:17:05
Document: ARR_kbd
Last saved on: Mon, Dec 9, 1991 08:17:06
Document: misc
Last saved on: Mon, Dec 9, 1991 08:17:07

AR and ARR User Information Manual

GTA Library Cabinet: Library/Cntl System/User Documents

Document File: AR ARR UIM

Author: Roger Cole

Date Written: December 18, 1989

Primary AT−8 Contact: Roger Cole

AT−8 Approval:

Date:

Revision History:
23 Oct 90 − rac − major re-write for version 2.0 AR and ARR, draft, in progress
16 Aug 90 − rac − editorial changes in UIM
20 Feb 90 − rac − (02/20/90) version, additional capabilities, change directory structure
22 Jan 90 − rac − (01/17/90) version, additional prototype capabilities
18 Dec 89 − rac − initial version, for AR prototype snapshot save/restore

Not to be distributed outside Los Alamos National Laboratory without permission of the author.

This is an AT Division internal report and as such is a privileged communication. There is no warranty as to the accuracy of the contents. Should you wish to refer to the information contained herein, you should first ask the author. Acknowledgment should be made to the author, with the notation “Private communication.”

Controls and Automation Group
Accelerator Technology Division
Los Alamos National Laboratory
Los Alamos, NM 87545

1. DISCLAIMER 1

2. INTRODUCTION AND OVERVIEW 1
2.1. Conventions Used In This Document 1
2.2. Limitations 1

3. AR_kbd—ARCHIVER 2
3.1. Overview of AR_kbd 2
3.2. Starting and Stopping AR_kbd 4
3.3. AR_kbd Commands 5
3.4. Archive Requests 5
3.5. Management Of Archive Files 8

4. ARR—ARCHIVE RETRIEVAL UTILITY 9
4.1. Overview of ARR 9
4.2. Some Notes On the Mouse and OPEN LOOK Windows 11
4.3. Starting and Stopping ARR 13
4.4. Retrieving Data From Archive Files 13
4.5. Plotting Retrieved Data 15
4.6. Printing Retrieved Data 18
4.7. Exporting Retrieved Data 19

5. ARR_kbd—ARCHIVE RETRIEVAL UTILITY, keyboard version 21
5.1. Quick Reference 21
5.2. General Overview 22
5.3. ARR_kbd Commands 22
5.4. Using ARR_kbd 26
5.5. export—writing synchronous samples in “foreign” formats 30
5.6. tsTest—time stamp test 30

6. TIME STAMPS 31

7. arAccessLib—ARCHIVE DATA FILE ACCESS LIBRARY 32

8. INSTALLATION OF SOFTWARE 32
8.1. Installing the Software 32
8.2. Effects Of Changing Versions 32
8.3. Platforms Supported 32

1. DISCLAIMER

The features described in this draft manual are subject to change until this manual is issued in a non-draft version. As of April 25, 1991, many of the features of ARR are regarded as being relatively stable, but not immune to changes.

2. INTRODUCTION AND OVERVIEW

This manual documents the EPICS data archiver, AR, and the archive retrieval program ARR. The two programs are documented together because they are so closely related, and most users of one program will also need to know about the other. In addition, ARR_kbd is documented here; this program contains much of the functionality of ARR, but with a keyboard interface.

2.1. Conventions Used In This Document

In the examples, underlined text indicated characters which were typed at the keyboard. Other characters were printed by ARR or the operating system, or are explanatory notes.

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

{ } indicate that one of the items enclosed must be chosen. (The braces must not be typed as part of the command.)

Some special characters are used by AR and ARR and are mentioned in this manual:

^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 then typing a ‘d’. (Don’t hold down the shift key, though.)

^Z is the “control Z” character, used under UNIX for interrupting a program so it can be moved into the background. ^Z is obtained on the keyboard by holding down the Control key and then typing a ‘z’. (Don’t hold down the shift key, though.)

2.2. Limitations

The following limitations apply to AR and ARR functionality as of April 25, 1991. These limitations will be removed in the future, as determined by priorities and available time. The descriptions for both programs apply to the versions which operate under version 2 of EPICS. ARR can be used with the “sample set” data files produced by the prototype AR, with some restrictions in functionality. THE PART OF THIS MANUAL WHICH DESCRIBES AR SHOULD NOT BE USED FOR OPERATING WITH THE PROTOTYPE ARCHIVER!

1. AR doesn’t run on Sun4 workstations.
2. ARR plots with color on color X11 displays; for PostScript (and EPS files) color isn’t utilized. Only linear scaling is provided for plotting.
3. Selective retrieval of data (based on the value of a channel, for example) isn’t supported by ARR.
4. ARR doesn’t support retrieval directly from Channel Access.
5. AR doesn’t handle variable length waveforms, such as compression records whose “size” is changed by an operator.
6. AR “sample set” data files don’t contain graphics information for the channels.
7. Neither global nor selective purging of data in “by channel” data files is supported.
8. ARR runs only under X11.

3. AR_kbd—ARCHIVER

AR_kbd, the archiver, acquires real-time data and stores the data in disk files. AR_kbd is a keyboard based program, using text commands. AR_kbd can run in either an interactive mode or in a server mode. (AR is a mouse and menu based program which can interact with AR_kbd running as a server; AR also provides additional features, such as creating and editing archive requests.)

This main section discusses the archiving software in general, but with emphasis on AR_kbd. Other main subsets of archiving software are discussed in depth later in this document. Topics covered in this section are:

section 3.1. page 2 general overview of archiving
section 3.2. page 4 how to use AR
section 3.3. page 5 commands for use with AR
section 3.4. page 6 archive requests—what they do, and how to build them
section 3.5. page 9 creating, copying, purging, and deleting archive files

3.1. Overview of AR_kbd

AR_kbd gets time-stamped data from Channel Access and stores the data on disk. Two different forms of storage exist. A ‘by channel’ data file stores the data for each channel as a stream of time-stamped values; a single file contains all the channels being archived. A ‘sample set’ data file stores the data as synchronous sample sets, where a sample set consists of a time-stamp and the values for the channels.

There are several ways to retrieve data from archiver data files. ARR and ARR_kbd allow interactive retrieval of archive data. Several libraries of routines are also available to allow programs to retrieve archive data.

3.1.1. archive requests

AR_kbd is guided in its activities by one or more archive requests. In essence, an archive request is a list names of channels to be archived, along with a set of instructions on how and when to archive the values for the channels. The “when” of archiving includes such possibilities as sampling periodically, continuous monitoring of changes, and sampling only when a particular condition is met.

Several conditions must satisfied in order for an archive request to actually do some sampling. First, the request must be enabled. Only when an archive request is enabled does AR_kbd “know about” it. Enabling and disabling requests must be done explicitly—either manually or by including a request’s name in the file which lists the requests to be automatically enabled by AR_kbd.

The second prerequisite for an archive request to do some sampling is that the request be active. A variety of methods are available for activating enabled requests. One common activation method is ‘always’—whenever the request is enabled, it will also be active. Another is ‘event’ activation, which is frequently used in conjunction with soft channels tied to OPI buttons. In this case, the request is set up to be activated when the button is ‘ON’ and inactivated when the button is ‘OFF’.

Finally, the sample method must be satisfied (for enabled, active requests) before any sampling occurs. One common sampling method is ‘periodic’, resulting in samples at fixed time intervals. Another method is ‘event’ sampling, similar to event activation, and often used to take a sample when an OPI button is pressed. A third common method of sampling is ‘monitor’, in which a sample is taken whenever the value for a channel changes.

3.1.2. ‘by channel’ data files

A ‘by channel’ data file stores data as time-stamped streams of values, one stream per channel. All archive requests use the same file. Multiple archive requests are allowed to be simultaneously active.

'By channel' archiving is highly optimized for high data rates and efficient usage of disk space. The only sampling method which is supported is 'monitor'. The idea is to capture all data which might possibly be of interest and then analyze the data in a more focused way after a run. If throughput or disk utilization are important, or if it is important to match up data by time-stamp, then 'by channel' archiving is likely to be the best choice. Retrievals from a 'by channel' file are faster than from 'sample set' files.

However, 'by channel' archiving has some disadvantages. The 'by channel' file is somewhat more cumbersome to manage than 'sample set' files. The sample data which results from an archive request loses its association with the request when 'by channel' archiving is used.

3.1.3. 'sample set' data files

A 'sample set' data file stores data as sample sets, where a sample set consists of a time stamp and a value for each channel. Each active archive request has its own separate 'sample set' file.

'Sample set' archiving is oriented toward pre-selecting the data to be archived, with less emphasis on throughput and space-efficient storage. By definition, 'sample set' archiving preserves the association of the data with the archive request which produced it. Since 'sample set' files are ordinary text files, managing them is easy using ordinary UNIX tools. They can be copied, renamed, and even edited.

Two disadvantages to 'sample set' archiving are the slower data rates and the less efficient use of disk space. Also, the time stamp for the samples is, because of the way sampling is done, only approximate; if precise time stamps are required, then 'by channel' archiving must be used.

3.2. Starting and Stopping AR_kbd

AR_kbd can be started in 3 different ways. The first way allows interacting directly with AR_kbd running in a shell. This approach is appropriate for short-duration archiving sessions, and for situations in which the shell in which AR_kbd runs will not be terminated. This method isn't appropriate if the user's workstation must be logged out but archiving is to continue. (Requests aren't automatically enabled.)

```
% cd directory<nl>                    (start AR_kbd in interactive mode)
% AR_kbd<nl>
  AR: enable all<nl>                  (requests must be explicitly enabled)
      ...                             (AR_kbd prints status information about requests)
  AR: quit<nl>
%
```

The second and third ways of starting AR_kbd are closely related; they allow using AR_kbd in a client/server mode, in which the server runs for extended periods. The AR_kbd server is started on a workstation which will not be logged out; the requests named in "arReq/enabled.dat" are automatically enabled when AR_kbd starts as a server. An AR_kbd client can be started on any workstation to connect to the server and check its status, enable or disable archive requests, etc. The AR_kbd server is stopped by using the "quit" (or ^D) command in the AR_kbd client.

```
% telnet serverHost<nl>              (start AR_kbd as a server; requests are auto enabled)
login: yourName<nl>
Password: yourPassword<nl>
% cd directory<nl>
% AR_kbd server&<nl>
      ...                             (AR_kbd enables all requests and prints status information)
% ^D                                 (logout from server host; AR_kbd keeps running)
%


% cd directory<nl>                    (connect to AR_kbd server for interactions)
% AR_kbd serverHost<nl>
  AR: command<nl>
      ...
  AR: close<nl>                       (disconnect; leaving AR_kbd server running)
%


% cd directory<nl>                    (connect to AR_kbd server for interactions)
% AR_kbd serverHost<nl>
  AR: command<nl>
      ...
  AR: quit<nl>                        (disconnect, shutting AR_kbd server down)
stop the server? y<nl>
%
```

3.3. AR_kbd Commands

3.3.1. quick reference

```
activate    reqName
close
create      nMegaBytes
disable     all
disable     reqName
enable      all
enable      reqName
inactivate  reqName
quit
sample      reqName
status
```

3.3.2. activate reqName

The specified request is activated. The request must already be enabled; its activation type must be ‘demand’.

3.3.3. close

Closes the connection between AR_kbd running as a client and the AR_kbd server. This command is valid only in this context.

3.3.4. create nMegaBytes

Creates “arChan/arChanData.bin”, the ‘by channel’ data file. The file will have a maximum size, in megabytes, as specified.

3.3.5. disable {**all** or **reqName**}

All currently enabled requests, or a specific request, will be disabled.

3.3.6. enable {**all** or **reqName**}

All requests named in “arReq/enabled.dat”, or a specific request, will be enabled.

3.3.7. inactivate reqName

The specified request is inactivated. The request must be active; its inactivation type must be ‘demand’.

3.3.8. quit

Shut down archiving. This command can be used either in interactive or client mode.

3.3.9. sample reqName

A sample is taken for the specified request. The request must be active; its sample type must be ‘demand’.

3.3.10. status

Print some information about the current state of AR_kbd, including currently enabled requests. Each request’s name and status is printed. Request status will be one of `active`, `inactive`, or `partly connected to CA`. The latter status indicates that one or more of the channels in the request doesn’t currently have a Channel Access connection.

3.4. Archive Requests

An archive request is a text file which contains a list of channels whose values are to be archived and a set of instructions which control the actual archiving. The “arReq” subdirectory contains the archive request files; the name of an archive request file is the name of the request followed by “.req”.

Line length in archive requests is limited to 80 characters.

Within an archive request, channel names occur in various contexts. A channel name consists of a database record name optionally followed by a field name. Channel names are case sensitive; e.g., ‘POWER:BASE’ is different from ‘power:base’. If no field name is specified, then the ‘.VAL’ field is assumed.

In the same subdirectory with the archive requests is a file indicating which archive requests are to be enabled with the ‘enable all’ command (or when AR_kbd is started as a server). This file is “./arReq/enabled.dat”. Two example files are listed below:

```
./arReq/sine.req
           NAME:       sine
           DESC:       demo archive request gathering sine waves
           ACT:        periodic 0:10:0
           INACT:      timed 0:2:15
           SAMPLE:     periodic 0:0:10
           OUTPUT:     set:txt
           %endHeader%
           POWER:BASE POWER:LAG30
           POWER:LAG60
           POWER:LAG90

./arReq/enabled.dat
           sine
```

The ‘enabled.dat’ file indicates that ‘sine’ should be enabled when AR_kbd is started as a server (or when the ‘enable all’ command is used).

‘sine.req’ indicates that values for 4 channels are to be archived. The list of channels follows the ‘%endHeader%’ line.

The ‘sine’ request is to be activated every ten minutes. It will be inactivated 2 minutes and 15 seconds following activation. During the time that ‘sine’ is active, it will sample its data channels every 10 seconds.

The data acquired by ‘sine’ is to be stored as a ‘sample set’ file, ‘./arSet/sine.txt’.

3.4.1. quick reference

```
NAME:       request name
DESC:       description (several lines can be used; each must start with DESC:)
ACT:        always
ACT:        periodic hh:mm:ss
ACT:        event chanName condition[1] [value] [every n times]
INACT:      never
INACT:      timed hh:mm:ss
INACT:      event chanName condition[1] [value] [every n times]
SAMPLE:     periodic hh:mm:ss
SAMPLE:     event chanName condition[1] [value] [every n times]
SAMPLE:     monitor                (required for chan:bin; illegal for set:txt)
OUTPUT:     set:txt
OUTPUT:     chan:bin
%endHeader%
chanName  chanName
chanName
...
[1] condition can be == != < <= > >= or change  For change, value must be omitted
```

3.4.2. event ...

`event chanName condition value [every n times]` is used to cause an action to take place when the value of a channel satisfies some condition. The default case is that the action occurs each time the condition is specified. If ‘every n times’ is specified, then the action occurs less frequently; for example, if ‘n’ is 3, then the action will occur the 3rd, 6th, 9th, ... times that the condition is satisfied. The conditions which may be specified are: `== != < <= > >=` which have their usual ‘C’ meanings. For channels of type DBF_ENUM, the ‘value’ must be one of the state strings for the channel.

`event chanName change [every n times]` is used to cause an action to take place when the value of a channel changes. The default case is that the action occurs each time the value changes. If ‘every n times’ is specified, then the action occurs less frequently; for example, if ‘n’ is 3, then the action will occur the 3rd, 6th, 9th, ... times that the channel’s value changes.

3.4.3. NAME:

This line isn’t interpreted by AR_kbd. It simply serves as documentation of the name of the archive request.

3.4.4. DESC:

‘DESC:’ lines contain descriptive information about an archive request. They aren’t interpreted by AR_kbd. Several lines of description may be used, each beginning with ‘DESC:’.

3.4.5. ACT:

The ‘ACT:’ line controls when an archive request is activated. When the activation condition is satisfied for a request, there will usually be a small delay before the request is actually activated. Thus, it isn’t possible to activate a request when an event occurs and also sample the actual data associated with the event.

All channels in the archive request must be connected through Channel Access before activation can complete. In an active request, if one or more channels disconnect from Channel Access then the request is inactivated until all channels are connected again.

`ACT: always` is used to keep a request active all the time. The request will be activated when it is enabled and will stay active until it is disabled.

`ACT: event ...` is used to activate a request in response to a condition on a database channel. The request will be activated whenever the condition is satisfied. See section 3.4.2. for details on specifying events.

`ACT: periodic hh:mm:ss` is used to periodically activate a request. The request will be initially activated as soon as it is enabled, and then re-activated at the specified interval. If the request is active when the interval expires, no action is taken and the interval timer is restarted. The period must be between 10 seconds and 24 hours.

3.4.6. INACT:

`INACT: never` is used to keep a request active all the time. The request will not be inactivated until it is disabled.

`INACT: event ...` is used to inactivate a request in response to a condition on a database channel. The request will be inactivated whenever the condition is satisfied. See section 3.4.2. for details on specifying events.

`INACT: timed hh:mm:ss` is used to inactivate a request after a time delay following activation. The time delay must be smaller than 24 hours and greater than 10 seconds. If activation is periodic, then the time delay for inactivation must also be at least 10 seconds less than the period for activation.

3.4.7. SAMPLE:

`SAMPLE: event ...` is used to take a sample in response to a condition on a database channel. A sample will be taken whenever the condition is satisfied. See section 3.4.2. for details on specifying events. There will be a small delay between when the condition is satisfied and when the channels are actually sampled. There also may be some slight differences in the times that the samples are taken for the individual channels. This option is legal only for ‘sample set’ archiving.

`SAMPLE: monitor` is used to sample all changes in the values of the channels. This option must be used with ‘by channel’ archiving; it is illegal for ‘sample set’ archiving. The phrase ‘sample all changes’ requires some elaboration. The monitor deadband in the database record controls what changes are actually sent to AR_kbd. The change in value must exceed the monitor deadband before a change is sent. (If the monitor deadband is 0, then a change is sent to AR_kbd each time the record is scanned, *even if the channel’s value hasn’t actually changed*.)

`SAMPLE: periodic hh:mm:ss` is used to sample periodically. There may be some slight differences in the times that the samples are taken for the individual channels. This option is legal only for ‘sample set’ archiving.

3.4.8. OUTPUT:

`OUTPUT: chan:bin` is used for ‘by channel’ archiving. The data are stored in ‘./arChan/arChanData.bin’. This is a special binary file created by the ‘create’ command. It grows as samples are written to it, until the maximum size specified by the ‘create’ command is reached.

`OUTPUT: set:txt` is used for ‘sample set’ archiving. The data are stored in the ‘arSet’ subdirectory in a file whose name is ‘reqName.txt’, where ‘reqName’ is the name of the archive request. The file is opened by AR_kbd only during the time that a sample is actually being written to disk. If the file doesn’t exist, it is automatically created. If the file does exist, samples are written at the end of the file. There is no checking of file size or enforcement of a size limit by AR_kbd.

3.4.9. %endHeader%

The ‘%endHeader%’ line indicates the end of the header information in the request. The list of channels to be archived is entered on following lines. There may be one or more channels specified per line; blank lines are ignored. If no field name is specified, the ‘.VAL’ field is assumed.

3.5. Management Of Archive Files

3.5.1. directory layout

3.5.2. archive request files and enabled.dat

3.5.3. “by channel” data files

AR_kbd flushes buffers periodically, so ‘up to the second’ data isn’t available to ARR

3.5.4. “sample set” data files

deleting a file while it is open by ARR may be confusing−−needs to be checked and documented

4. ARR—ARCHIVE RETRIEVAL UTILITY

ARR, the archive retrieval utility, supports retrieving data from AR files, displaying the data in textual or graphical form, and exporting the data for use by other programs. ARR has a mouse and menu based interface which runs under the X Window System. (ARR_kbd is a keyboard based utility which provides similar functions; it runs under either SunView or the X Window System.) The presentation for ARR consists of several sections. Section 4.1. is an overview of the functions provided by ARR. Section 4.2. gives some information about using the mouse and interacting with windows on the screen. Actual operations for ARR are described beginning with section 4.3. on page 15.

4.1. Overview of ARR

ARR retrieves data from files created by AR. Retrieved data is in the form of synchronous samples. Once a group of synchronous samples has been retrieved, it can be repeatedly processed in various ways. When a new group of synchronous samples is retrieved, the prior group is no longer available for processing.

The processing capabilities of ARR, though rudimentary, are adequate for many needs. Where more sophisticated processing is needed, ARR can export data in a form which is easily accessible by spreadsheet programs.

4.1.1. synchronous sample specification

ARR keeps a list of channels for which samples are to be retrieved. Most ARR operations automatically use this list of channels. The list is kept in the synchronous sample specification, which controls retrievals. In addition to the list of channels, the synchronous sample specification also contains information about where to retrieve samples from and information about which synchronous samples to accept or reject.

Initially, ARR doesn't know where to retrieve data from or which channels to use. The **retrieving...** button brings up the "RETRIEVE" window; the **setup...** button in the "RETRIEVE" window brings up the "retrieval setup" window. The "retrieval setup" window allows picking the source for retrieval and also picking the channels to retrieve. Before anything useful can be done, a file must be picked and channels must be selected. Adding channels can be done at any time, even after some samples have been retrieved. The only side effect of adding channels is that if any synchronous samples have been retrieved, they are discarded.

Channels can be removed from the list at any time using the **delete** button. The synchronous samples for deleted channels are discarded, but the synchronous samples for the remaining channels aren't affected.

4.1.2. synchronous sample

Accurately understanding the phrase synchronous sample requires a little background. Each data value has a time stamp associated with it. Consider a sample consisting of a value (with time stamp) for each of several channels. If all the time stamps in the sample are equal, then the sample is called a synchronous sample for the channels. ARR usually handles a group of several synchronous samples.

The example below shows a group of 3 synchronous samples for 2 channels:

```
      time stamp           chan1     chan2
01/11/91 08:13:14.000     1.2 O     3.6 O
01/11/91 08:13:14.100     1.3 O     1.8 O
 01/11/91 08:13:14.200     1.4 O     1.8 O
```

In the example above, a value is shown for each channel every 100 milli-seconds. Each synchronous sample is complete, since both channels contribute a value for each sample. (The letter 'O' following each value indicates that the value is 'okay'.)

Notice that the second and third values for chan2 are the same in the example. Normally, the control system reports a value for a channel only when the value changes. On the surface, this implies that it isn't possible to get a *complete* synchronous sample if one of the channels changes slowly or infrequently. One way to get around this is to set the monitor deadband in the database to 0 for such channels; this forces the control system to report a value each time the time stamp changes.

Another approach is to ask ARR to give special treatment to channels for which a value is available only on an irregular basis. The irregularity may be because the channel changes slowly, as for a thermocouple, or because the channel changes infrequently, as for a power supply on/off relay status. In order to use such channels in synchronous samples, the **Add channel, filled** button must be used. If the channel actually has a value with the synchronous sample's time stamp, then the situation is straightforward. If there is no value at that time stamp, then the most recent value for the channel is used; the underlying assumption is that the channel's value hasn't changed.

When a synchronous sample contains a value which is "filled in" from an earlier sample, ARR flags the value with an 'F'. Consider, for example, that chan3 is a thermocouple for which the only value in the file is at 01/11/91 08:13:14.000. Then a group of synchronous samples containing chan3 might appear as:

```
        time stamp            chan1      chan2      chan3
01/11/91 08:13:14.000      1.2 O      3.6 O     37.7 O
01/11/91 08:13:14.100      1.3 O      1.8 O     37.7 F
 01/11/91 08:13:14.200      1.4 O      1.8 O     37.7 F
```

If, for some reason, a channel declared with the **Add channel, sync** button doesn't have a value with a particular time stamp, then a synchronous sample at that time stamp is partial. For plotting, there will be a gap in the plot for the channel. For text output, the indication is more explicit. For example, suppose that chan4 doesn't have a value at 8:13:14.100. Then the output from the ARR's print command would be:

```
        time stamp            chan1      chan2      chan3       chan4
  01/11/91 08:13:14.000     1.2 O      3.6 O     37.7 O      -1.0 O
* 01/11/91 08:13:14.100     1.3 O      1.8 O     37.7 F   no_data MN
  01/11/91 08:13:14.200     1.4 O      1.8 OR    37.7 F       1.0 O
```

In the above example, the missing value for chan4 had 3 effects. First, 'no_data' was displayed, rather than a numeric value. Then, the flag is 'MN', indicating "missing data", rather than the usual 'O', which indicates "okay". And, finally, the line was preceded by a '*', indicating that the synchronous sample was a partial one. Notice that the third value for chan2 is flagged as having a hardware read alarm ('OR'; despite this, the line isn't flagged with a '*' since such values from the control system have a valid time stamp, and thus don't make a sample partial.

The last variation on the synchronous sample theme arises when the channels aren't "in sync", either with each other and/or with themselves. Consider the following mess:

```
        time stamp              chanA         chanB
   01/11/91 08:13:14.000        12 O          36 O
 * 01/11/91 08:13:15.000        13 O     no_data MN
 * 01/11/91 08:13:15.100   no_data MN         18 O
   01/11/91 08:13:16.000        14 O          9 O
```

In the example above, both chanA and chanB were expected to change once per second. For its middle value, chanB was .1 second late. This example demonstrates the technique of acquiring synchronous samples by choosing the sample time stamp by picking the earliest of the next available value time stamps.

When there is only a single channel being sampled, then a group of synchronous samples consists of the stream of values for the channel.

4.1.3. status codes for values

For the ARR print operations, and for the "export" operations which 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
	`EOF` end of file for that channel

The second character, when present, conveys additional status information. (For plotting with **show** specified for the "value status" item on the "plotting setup" window, 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 violated
`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

4.2. Some Notes On the Mouse and OPEN LOOK Windows

click left position the mouse pointer, depress the left mouse button, and immediately release the button.

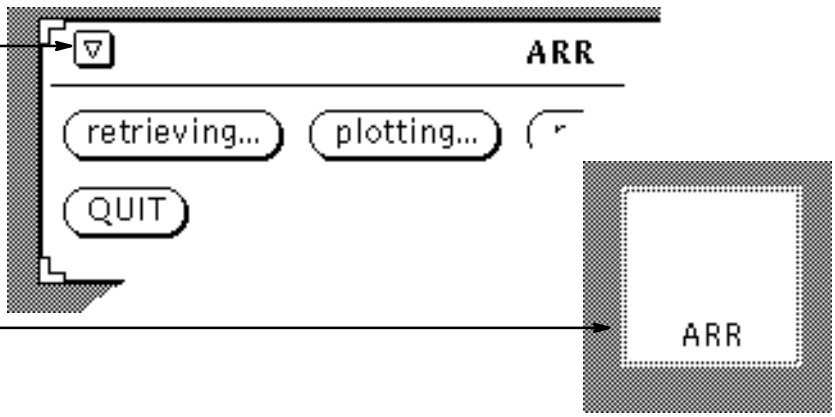
double click left similar to click left, but twice, quickly

hold left position the mouse pointer, depress the left mouse button, and accomplish the desired function while holding down the button

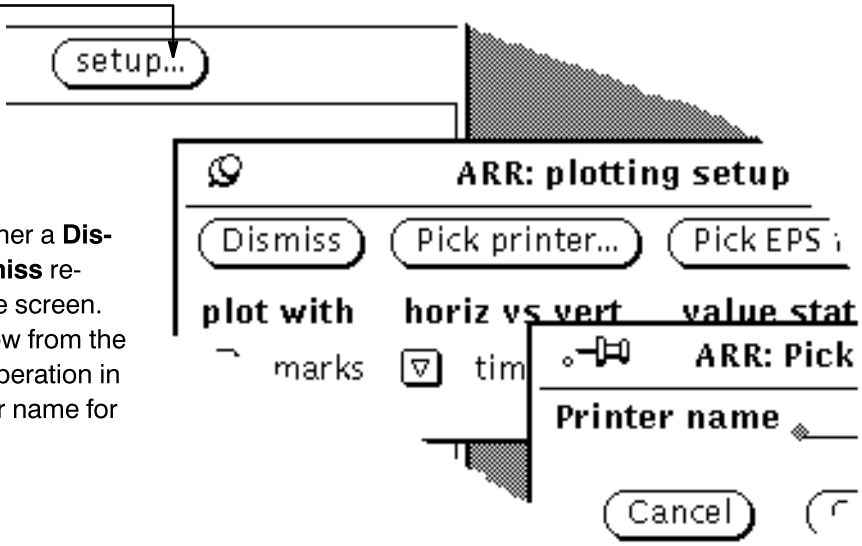
hold right similar to hold left, but using the right mouse button

ARR uses a number of the conventions established for OPEN LOOK user interfaces. Several of these are shown below.

iconify indicator a click left on this makes the window disappear from the screen; an icon will then appear. To see the window again, do a double click left on the icon.



sub-window indicator ... indicates that invoking a button or menu item will result in bringing up another window.



Most sub-windows will have either a **Dismiss** or a **Cancel** button. **Dismiss** removes the sub-window from the screen. **Cancel** removes the sub-window from the screen, and also 'forgets' the operation in progress—e.g., setting a printer name for hard copy output.

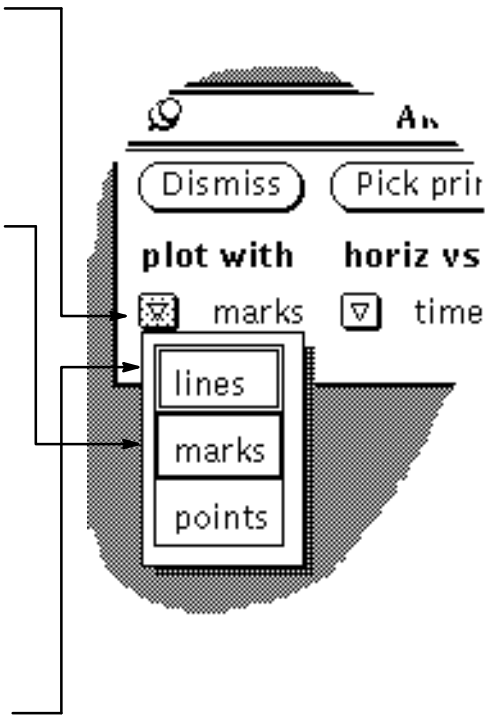
menu indicator on buttons and in square boxes, a downward pointing triangle indicates that a menu is available for the item. Several actions are possible for items with menus:

click left on the menu indicator (either in the box, as shown here, or in a button which contains a menu indicator) will invoke the default menu item. (See the paragraph on the control key, below, for setting the default menu item.) An active menu item is surrounded by a heavy line.

click right will pop up the menu. A menu item can be invoked with either a click left or a click right on the item. The menu can be dismissed without changing the menu selection by either a click left or a click right outside the menu.

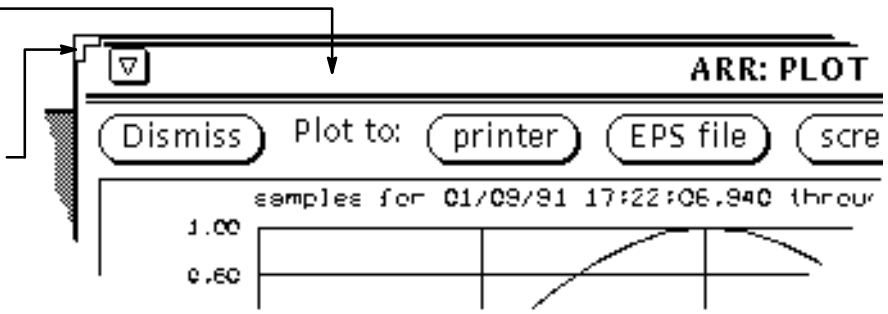
hold right will pop up the menu and allow sliding to the menu item to be invoked. If the mouse pointer is on a menu item when the right button is released, then that menu item will be invoked. If the pointer is moved off the menu before releasing the button, then the menu selection isn't changed.

control key allows setting which menu item is the default. When a menu item is invoked, if the control key is presently depressed then that menu item will become the default. The default menu item is enclosed by an extra box.

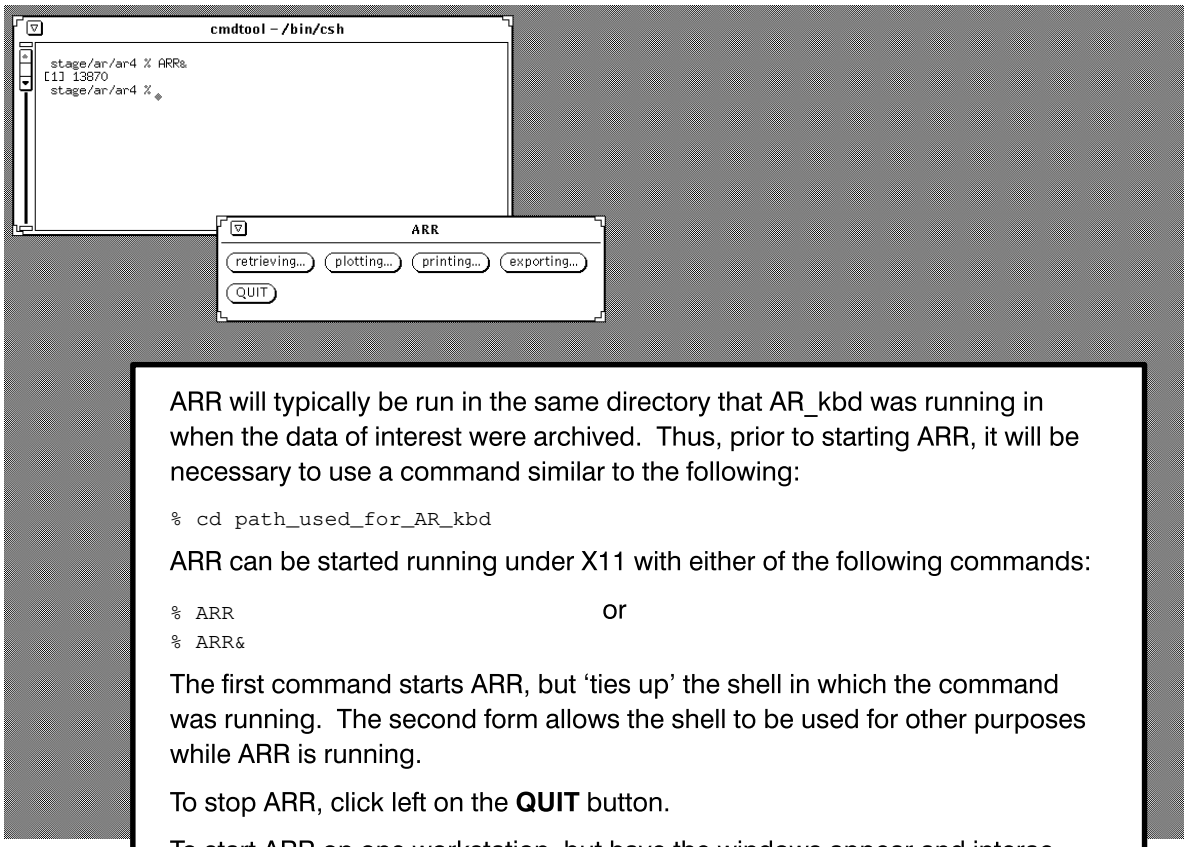


move to move a window, put the mouse pointer on the title bar, do a hold left, and move the window as desired.

resize windows with miter corners can be resized by putting the mouse pointer on one of the corners, doing a hold left, and resizing as desired.



4.3. Starting and Stopping ARR



ARR will typically be run in the same directory that AR_kbd was running in when the data of interest were archived. Thus, prior to starting ARR, it will be necessary to use a command similar to the following:

```
% cd path_used_for_AR_kbd
```

ARR can be started running under X11 with either of the following commands:

```
% ARR
```
or
```
% ARR&
```

The first command starts ARR, but 'ties up' the shell in which the command was running. The second form allows the shell to be used for other purposes while ARR is running.

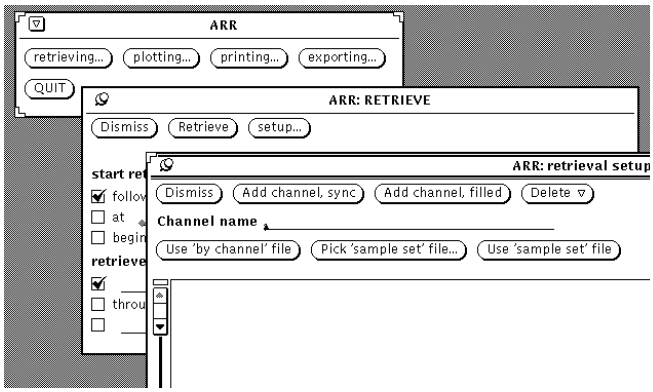
To stop ARR, click left on the **QUIT** button.

To start ARR on one workstation, but have the windows appear and interactions occur on another, use the following commands on the workstation on which ARR is to start:

```
% setenv DISPLAY schwinn:0
```
(replace `schwinn` with the desired name)
```
% ARR&
```
(or omit the `&`, as desired)

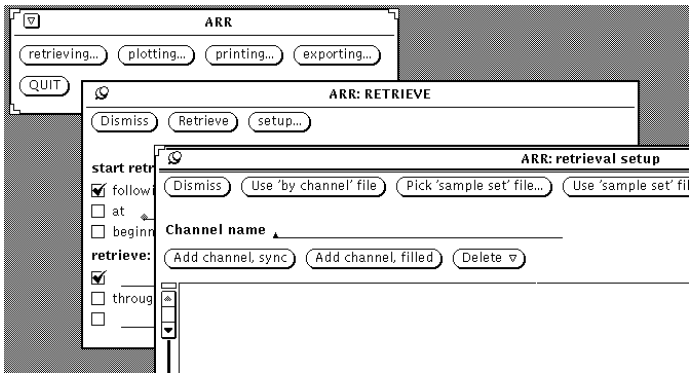
4.4. Retrieving Data From Archive Files

First, a source for retrieval must be selected. Use the **retrieving...** button to bring up the "RETRIEVE" window, and then use the **set-up...** button there to bring up the "retrieval setup" window. In the "retrieval setup" window, the **Use 'by channel' file** and **Use 'sample set' file** buttons allow choosing the source of the data.



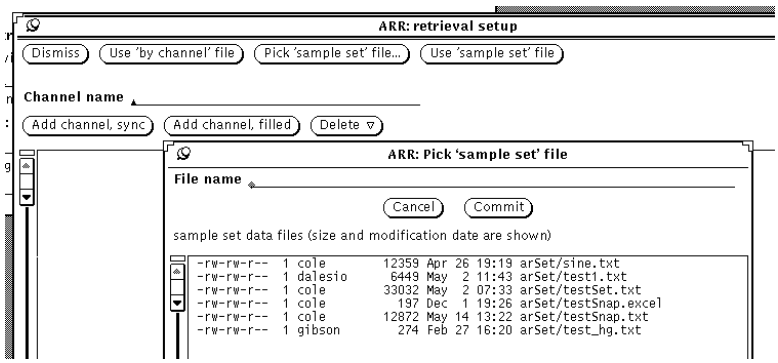
4.4.1. selecting the ‘by channel’ file

A click left on **Use ‘by channel’ file** selects the AR ‘by channel’ file as the source for data retrieval. A list of channels in the file is shown, along with their element count (1 for scalar channels), their type, and the time range of the data available for each channel.

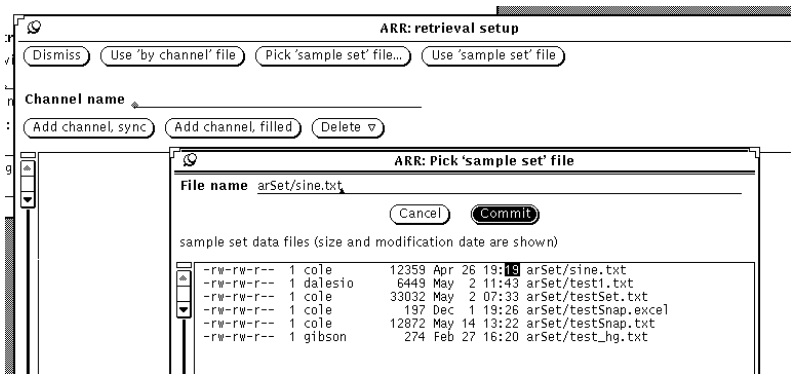


4.4.2. selecting a ‘sample set’ file

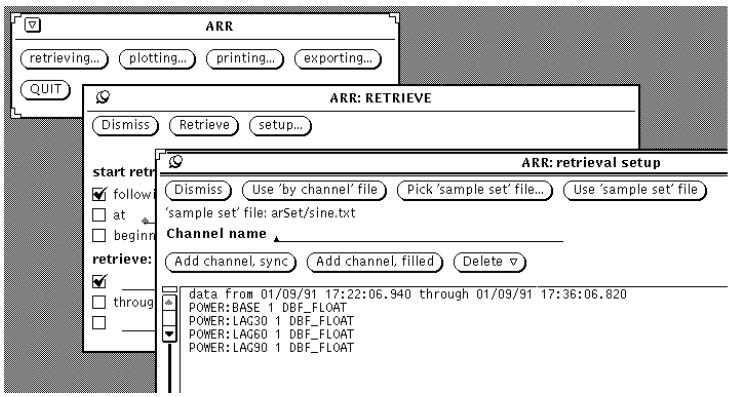
Using a ‘sample set’ file requires an additional step—choosing the file. A click left on **Pick ‘sample set’ file** brings up the “Pick ‘sample set’ file” window and shows a list of the available ‘sample set’ data files. The file sizes, in bytes, and their last modification date are shown. Changing from one ‘sample set’ file to another would involve a similar sequence of steps.



The easiest way to select a file is to double click left *anywhere* on the line for the file; several characters on the line will be highlighted. Then do a click left on the **Commit** button. The path name for the file will be copied onto the “File name” line and the “Pick ‘sample set’ file” window will be dismissed. (The ‘action’ is frozen here for illustrative purposes.)

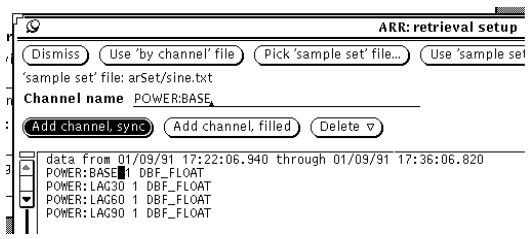


A list of channels in the file is shown, along with their element count and their type. Because of the nature of ‘sample set’ data files, the time range of the data available is the same for all channels in the file.



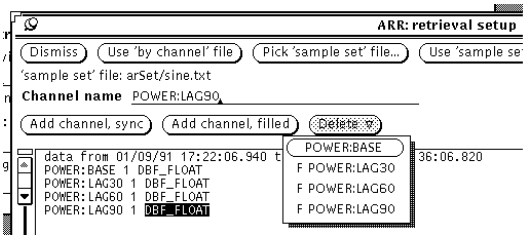
4.4.3. selecting channels for retrieval

The easiest way to select a channel is to double click left *anywhere* on the line for the channel; several characters on the line will be highlighted. Then do a click left on either the **Add channel, sync** or the **Add channel, filled** button. The channel's name will be copied onto the "Channel name" line and it will be added to the list of channels to retrieve. (The difference between the two buttons is discussed in section 4.1.2.)



At any time, a hold right (or click right) on the **Delete** button will show what channels have been selected. An 'F' in front of a channel name indicates that it was added with **Add channel, filled**.

Once the desired channels have been selected, the "retrieval setup" window can be dismissed.

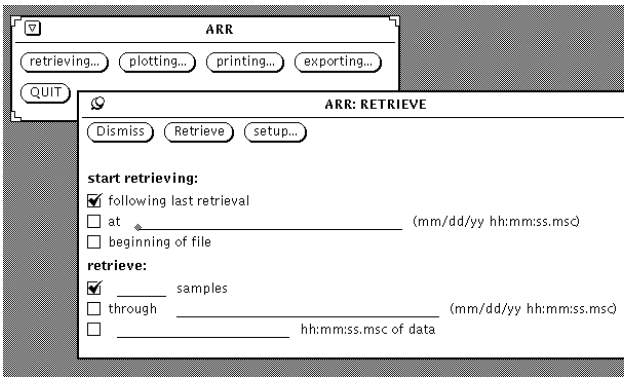


4.4.4. retrieving some data

Retrieving a group of synchronous samples is done with the "RETRIEVE" window. The starting point for the group and the 'size' of the group must be specified. (Immediately after a file has been chosen, **following last retrieval** is the same as **beginning of file**.)

A click left in a box selects an option.

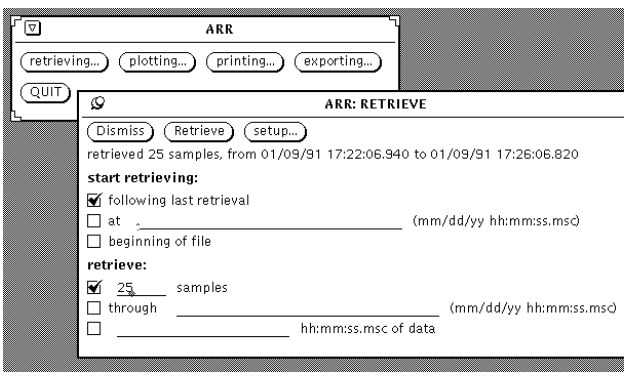
To enter a value, e.g., the number of samples to retrieve, do a click left above the space provided for the value.



Once the parameters for retrieval have been set, a click left on **Retrieve** will retrieve the group of samples. The number of samples in the group, and the time range, are shown.

Each time **Retrieve** is used with **following last retrieval**, the *next* group of samples in the file is retrieved.

See section 6. on page 33 for details on allowable forms for entering time stamps.



After a group of samples has been retrieved, ARR allows plotting, printing, and exporting the data. The following sections describe the various operations.

4.5. Plotting Retrieved Data

need to discuss using HOPR and LOPR for scaling

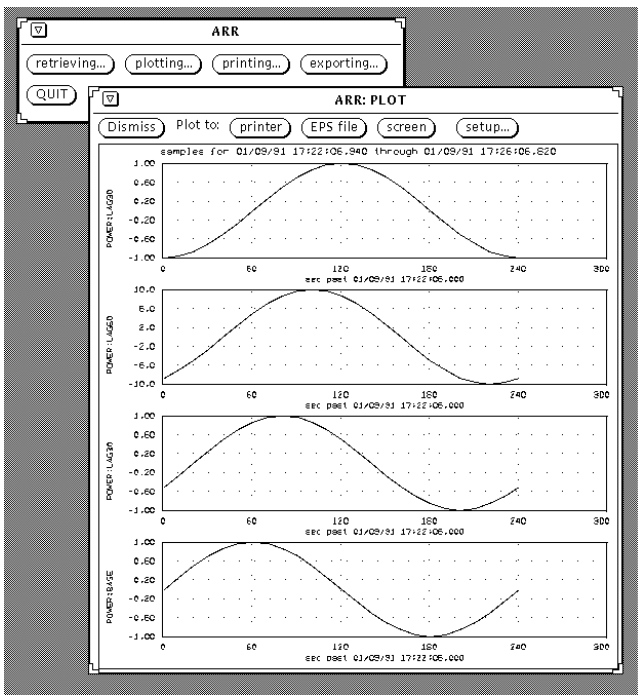
need an 'auto−range' function, for all or individual channels

When the “PLOT” window is brought up with a click left on **plotting...**, the group of samples is plotted on the screen using the current plot setup.

Even if the “PLOT” window is visible when a new retrieve is done, the new group of samples isn’t automatically displayed. Instead, the **screen** button must be used.

The “plotting setup” window allows making changes in the appearance of the plot. (It also is needed for some other setup for plots.) This window can be brought up with the **setup...** button.

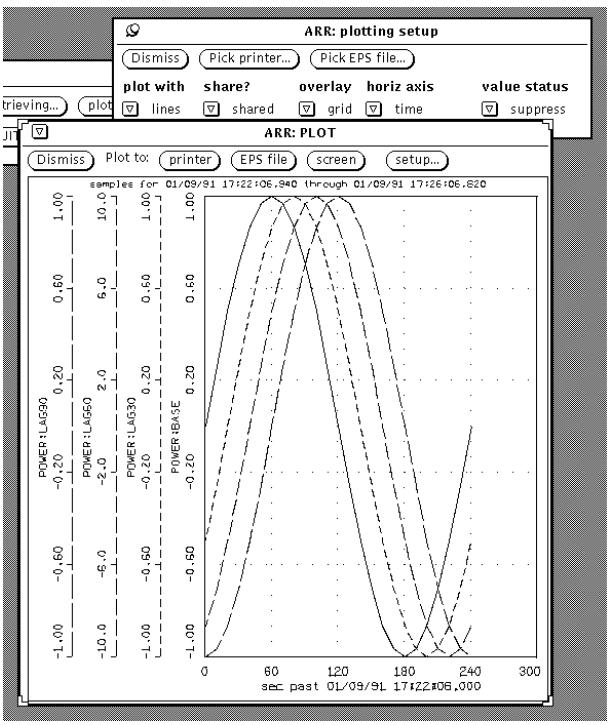
When ARR begins running, the plotting setup produces the type of plot shown here. Each channel is plotted on its own grid, using lines for drawing, with time on the horizontal axis.

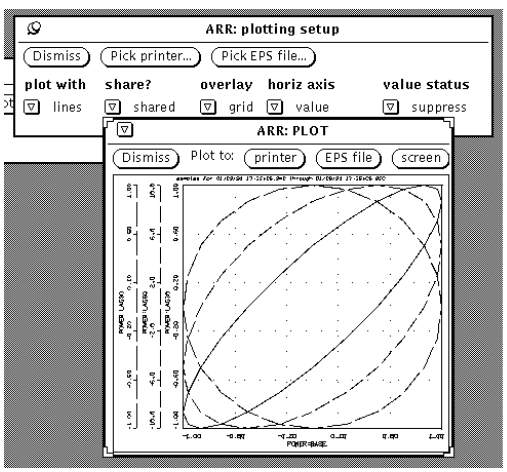


If the plot setup is changed, the samples can be re-plotted by using the **screen** button.

When the **share?** option is set so that all channels share the same space, ARR provides for distinguishing between the various channels. On monochrome media, dashed line patterns are used. On color media, each channel is plotted with a separate color instead of using a dashed line pattern.

Experimenting with the setup parameters will quickly show the possibilities. The options on the “plotting setup” window are explained below; to save space, examples of all possible plots aren’t shown.

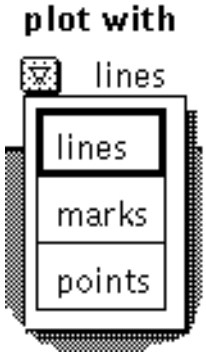


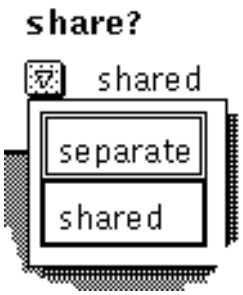


The **lines** option results in using lines for plotting the data. On monochrome media, dashed lines will be used; on color media, the lines will be colored rather than dashed.

When **marks** has been selected, then a small geometric figure will be drawn at each data point. On color media, the marks will be drawn with colors.

The **points** option produces small dots at the data points. If the plotting medium supports color, the dots will be colored.





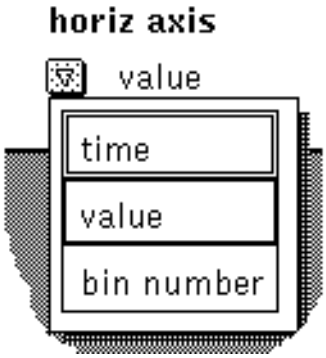
separate results in each channel getting its own grid for plotting. All channels share the same plotting area if **shared** is selected.

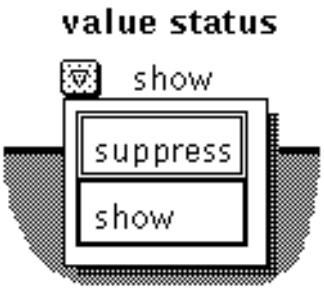
The only type of plotting overlay presently available is a rectangular grid.

When **time** is selected for plotting on the horizontal axis, then the sample values are plotted against their respective time stamps.

The value of the first channel in the list is plotted along the horizontal axis when **value** is selected. The values for the remaining channels are then plotted against the values for the first channel. In order for a point to be plotted for a channel, both the channel and the first channel must have the same time stamp. If plotting is with lines, the lines connect points with adjacent time stamps.

The **bin number** option is oriented primarily to plotting data such as waveforms. The waveform is thought of as having ‘n’ bins of data, numbered from ‘n’ through ‘n−1’. If scalar channels are plotted with this option, then ‘bin number’ means ‘sample number’ within the group of samples.





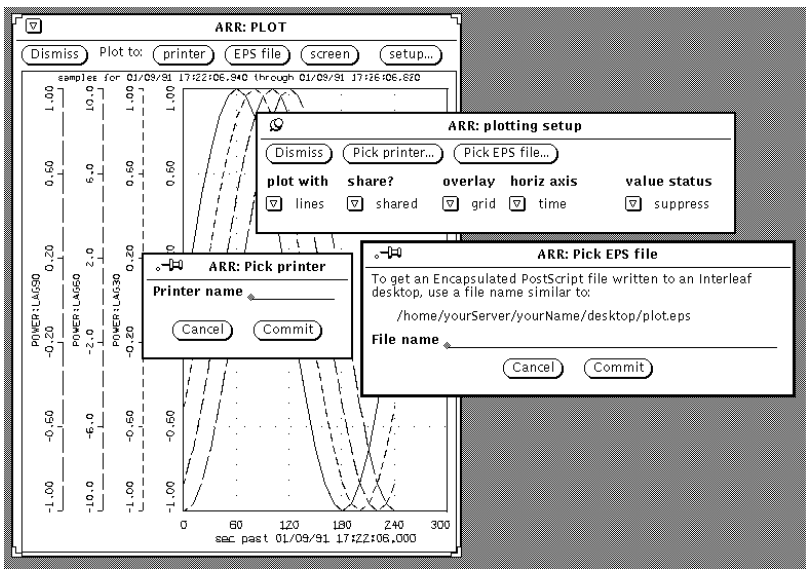
When the **show** option is selected, then an indicator of alarm status is drawn at each point. See section 4.1.3. for the meanings of the codes. (The codes in the example on the right are ‘H’ for a high alarm limit violation and ‘h’ for a high warning limit violation.)

4.5.1. PostScript and Encapsulated PostScript plots

The **printer** button can be used to get a hard copy of a plot. This requires specifying the name of a PostScript printer.

An encapsulated PostScript (EPS) file can be produced with the **EPS file** button. Such files can be included in documents prepared with such word processing programs as Interleaf or TeX. A path name for the EPS file must be specified.

In both cases, the size of the plot which is produced depends on the size of the plot on the screen. For printing, portrait orientation will normally be used; large rectangular plots will be done with landscape orientation.



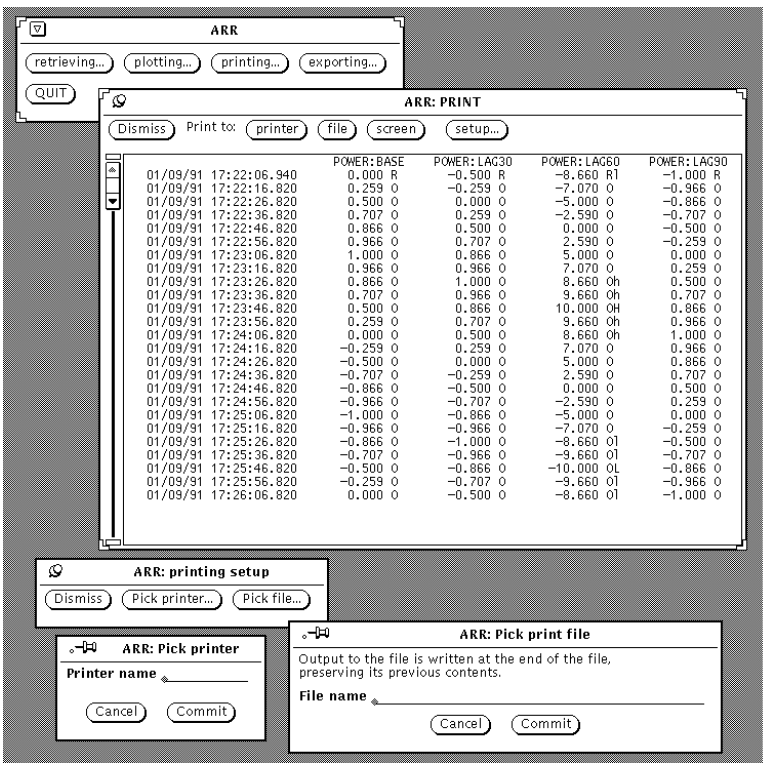
4.6. Printing Retrieved Data

A textual presentation of the current group of samples is available via the **printing...** button on the main “ARR” window. See section 4.1.3. for an explanation of the status codes.

Even if the “PRINT” window is visible when a new retrieve is done, the new group of samples isn’t automatically displayed. Instead, the **screen** button must be used.

Use the **printer** button to get a hard copy of the samples—if a printer name hasn’t previously been selected, it will be necessary to do so. (The same printer name is used both for plots and text; if the printer used for text isn’t a PostScript printer, then it will have to be changed before getting hard copies of plots.)

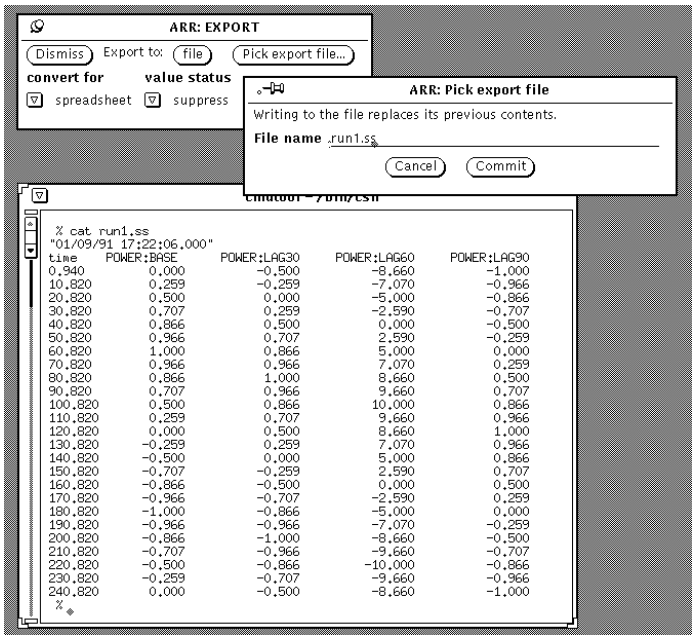
It is also possible to write the samples to a file in a form suitable for printing or editing. This is done with the **file** button. A file path name must be specified prior to the first write.



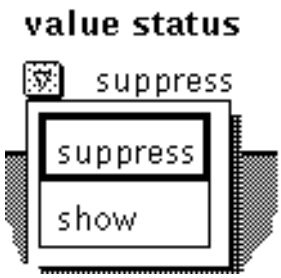
4.7. Exporting Retrieved Data

The “EXPORT” window supports writing groups of samples in forms which can be used by other programs. The exported output is written to a file. Any previous contents of the named file is lost—if several sets of data are to be exported, then the file name will have to be changed prior to each export operation.

Use the **file** button to actually perform the export.



The only type of export conversion presently supported is **spreadsheet**. This generates a file which can be used with Lotus 1-2-3 or Excel.



The status of individual values can be omitted from the exported file (when **suppress** is chosen) or included (using **show**).

4.7.1. format of spreadsheet files

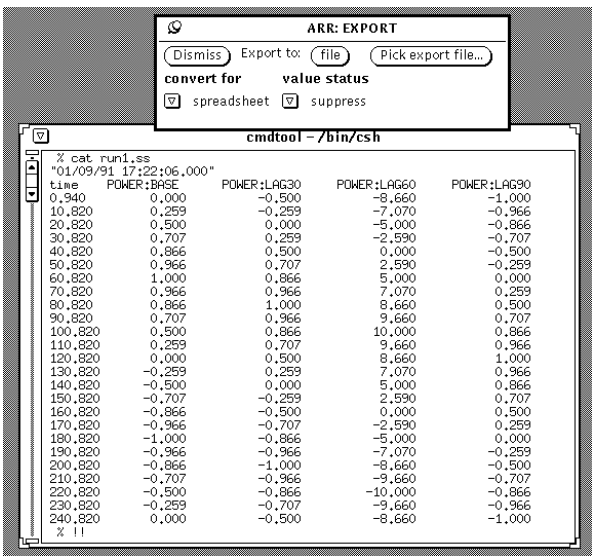
The format of spreadsheet files is organized into lines and, within lines, cells. The first line of the file is the text date and time used as a reference for the samples. The second line is a set of column headings corresponding to the actual data in the cells on the third and following lines.

The simplest spreadsheet output results when **suppress** is chosen. The resulting file doesn't contain status codes for the values.

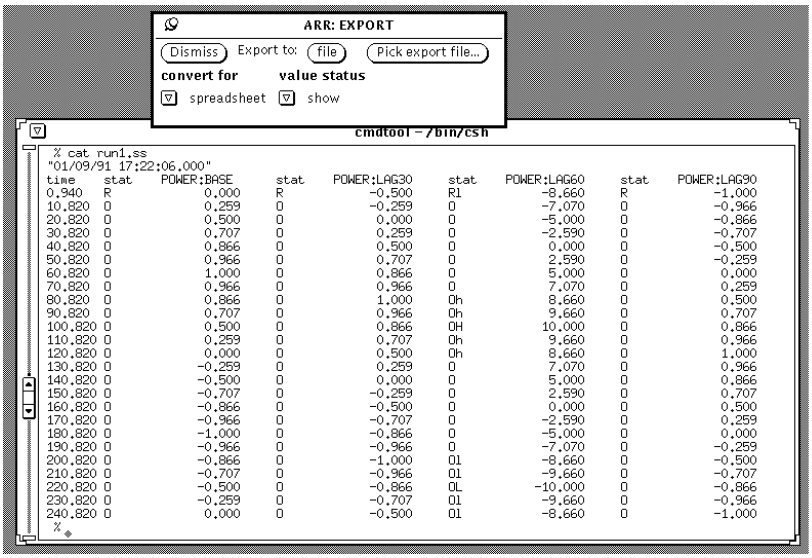
The first cell of each line is the elapsed time, in seconds, for the sample from the reference time on line 1 of the file.

If there is no data for a channel in a particular sample, then the channel's cell will contain the text 'no_data'. Most spreadsheet programs will complain mildly and use a value of 0 for the cell.

740.820	−0.500	0.000	5.000	0.866
750.820	−0.707	−0.259	2.590	0.707
760.820	no_data	−0.500	0.000	0.500
770.820	no_data	−0.707	−2.590	0.259
780.820	−1.000	−0.866	no_data	0.000
810.820	−0.707	−0.966	no_data	−0.707
R20.820	−0.500	−0.866	−10.000	−0.866



When **show** is used, each value is *preceded* by a status code, as explained in section 4.1.3. Thus, there are 2 cells for each value—one for the code and another for the actual value.



5. ARR_kbd—ARCHIVE RETRIEVAL UTILITY, keyboard version

ARR_kbd, the archive retrieval utility, supports retrieving data from AR files and displaying that data in text or plotted form. ARR_kbd provides functionality similar to that of ARR, but has a keyboard interface rather than ARR's graphical user interface. ARR_kbd operates under either SunView or the X Window System. The presentation for ARR_kbd is broken up into several sections. Section 5.2. describes the philosophy of operation of ARR_kbd, to provide the basis for a clear understanding of the various ARR_kbd commands. Section 5.3. on page 24 lists the various ARR_kbd commands and gives detailed descriptions of each.

Actual instructions on using ARR_kbd are covered in section 5.4. on page 28. The differences between accessing AR 'by channel' data files (section 5.4.1. page 28) and AR 'sample set' data files (section 5.4.2. page 28) are discussed. Several examples of using ARR_kbd, with the resultant output, are given in section 5.4.4. (page 30).

5.1. Quick Reference

```
% cd directory<nl>                  retrieve from a 'by channel' data file
% ARR_kbd<nl>
  ARR: channel chanA chanB<nl>
  ARR: plot 10<nl>
  ARR: quit<nl>
%

% cd directory<nl>                  retrieve from a 'sample set' data file
% ls arSet/*.txt                    get a list of 'sample set' files
% ARR_kbd<nl>
  ARR: sampleSet arSet/test.txt<nl>
  ARR: fileInfo                     get a list of channels in the file
  ARR: channel POWER:BASE, POWER:LAG30<nl>
  ARR: print 10<nl>
  ARR: plot<nl>
  ARR: ^D
%
```

```
generic commands:
  dataOut         [filePath]      (default is to normal output)
  bg
  help            [topic]
  quit (or ^D)
  source          filePath

ARR-specific commands:
  byChannel
  channel[,fill] chanName [chanName ...]
  delete          all  or  chanName [chanName ...]
 *export,type     [nSamples]          type: ss or ssStat
  fileInfo
  plot[,type]     [nSamples]          type: [p][t,x]{y or yy}[s] e.g., y, ty, xys, or pxyy
  plotEPS[,type]  [filePath]
  plotHC[,type]   [printerName]
  position        timeStamp  or  +hh:mm:ss.msc  or  -hh:mm:ss.msc
 *print           [nSamples]
  printHC         [printerName]
  rewind
  sampleSet       filePath
  skip            nSamples
 *tsTest          [nSamples][,interval[,slop]]
                  default for nSamples is 1000000000
                  interval in floating seconds
                  if interval specified, default for slop is +/− .0015 sec
```

Output from commands flagged with * can be routed to a file by using the "dataOut filePath" command. The present contents of the file are preserved, with new output being written at the end.

5.2. General Overview

ARR_kbd retrieves data from files created by AR. Retrieved data is in the form of synchronous samples. Once a group of synchronous samples has been retrieved, it can be repeatedly processed in various ways. When a new group of synchronous samples is retrieved, the prior group is no longer available for processing. Some ARR_kbd commands both retrieve a group of synchronous samples and also process it; other commands are restricted to processing a previously retrieved group.

The capabilities for ARR_kbd are similar to those of ARR. When questions arise about ARR_kbd, the documentation for ARR should be consulted.

5.3. ARR_kbd Commands

Many ARR_kbd commands have "`[nSamples]`" as a parameter. This indicates that specifying a sample count is optional. If no sample count is specified, the command will process the current group of synchronous samples. If a sample count is specified, then a new group of synchronous samples is retrieved prior to processing. When a new group samples is retrieved, retrieval begins at the current position in the file.

For commands which are followed by a comma-separated parameter, such as "plot[,type]", it is important that no space appear between the command and the comma whenever the parameter is used.

5.3.1. generic commands

dataOut [filePath]
Sets the destination for the "export", "print", and "tsTest" commands. By default, the output from these commands is written to the standard output. If a file path is specified, then the output from the commands is written to the end of the file, preserving the previous contents. Using the "dataOut" command without specifying a file path redirects output to the standard output.

bg
Under VxWorks, moves ARR_kbd into the "background", so that ARR_kbd will continue running but allow the shell to be used for other purposes. To resume interactions with ARR_kbd, use the ARR_kbd command at the shell prompt. Using this command under UNIX produces an error message. (See section 5.4.3. on page 29 for more details.)

help [topic]
Allows getting on-line information. If no topic is specified, a list of legal topics is printed.

source filePath
Causes ARR_kbd to read the indicated file as the source for commands. As commands are read from the file, they are printed on the screen. When end of file is encountered, ARR_kbd resumes reading commands from the keyboard. There are some limitations on the commands which can be used in a source file. The "source" command is specifically allowed. The "bg" and "quit" commands are not allowed.

Comments lines are allowed in the file—precede them with the # character in column 1.

5.3.2. byChannel

Selects arChan/arChanData.bin (the normal 'by channel' data file from AR) as the source for retrieval of data. If any channels were selected prior to using this command, they are deleted from the synchronous sample specification.

5.3.3. channel[,fill] chanName [chanName ...]

The named channels are added to the synchronous sample specification. Several channel commands can be used to create the list of desired channels. It is allowed to mix the channels in the list, having some specified without the ",fill" option and some with it.

The order in which channels are added determines the order they will appear in output. Generally, the first channel appears on the left for text forms of output and on the bottom for plotted forms. For x vs. y plots, the first channel is used as the 'x channel' and corresponds to the horizontal axis.

If the ",fill" option isn't used, each channel must have a value for a particular time stamp in order to produce a complete synchronous sample. Using the ",fill' option for a channel allows ARR_kbd to 'fill in' a value from a prior time stamp in order to avoid a partial synchronous sample. (See section 4.1.2., page 10, for more details.) The ",fill" option is ignored if retrieval is from a 'sample set' data file.

5.3.4. delete chanName [chanName ...]

One or more channels are deleted from the synchronous sample specification.

5.3.5. delete all

All channels are deleted from the synchronous sample specification.

5.3.6. export,type [nSamples]

The synchronous samples are exported into a particular form, as specified by the type. If nSamples is specified, a new group of synchronous samples is obtained before exporting. The output from this command is normally written to the standard output; to direct it to a file, use the "dataOut" command. Unless noted otherwise, an export type is intended for use only with scalar channels. For a presentation of the file formats, see section 4.7.1. The valid types are:

`ss`	produces output suitable for importing into a spreadsheet program.
`ssStat`	same as "`ss`", except each value is preceded by a one or two character status flag.

5.3.7. fileInfo

Prints some information about the file from which retrievals are being done. This includes a list of the channels which the file contains, and the range of time stamps available in the file. The channel type and element count are shown for each channel.

5.3.8. plot[,type] [nSamples]

Plots the synchronous data. If nSamples is specified, a new group of synchronous samples is obtained before plotting. If the ",type" option is omitted, the previous plot type is used. The initial default is ",ty". The valid plot types are discussed below.

When a channel has a value of 'no_data' and when a value has a 'restart' status, there will be a gap in the plot. For x vs. y types of plots, if the 'x channel' has such a gap, then there will be a gap in all the plots.

The first time the plot command is used, the plot window has a default size and position on the screen. This plot window can be freely resized and repositioned; subsequent uses of the plot command will use the most recent size and position. ARR_kbd doesn't accept commands while the plot window is on the screen; to dismiss the plot window, click the right mouse button with the pointer in the plot window.

Only the first value is plotted for waveform channels when they are plotted with non-waveform oriented plot types.

plot types are selected with codes which appear, at first glance, to be somewhat cryptic. There are several fundamental plot codes, each of which can have a prefix and/or suffix code. The valid prefixes and suffixes are:

m as a prefix results in a mark plot, rather than the usual line plot
`p` as a prefix results in a point plot, rather than the usual line plot
`s` as a suffix results in showing the display of alarm and status information

`ty` Each channel has its own individual grid, displaying time vs. value. If ‘s’ is used as a suffix, values with an error or alarm status have a status code plotted. The individual plots are sized vertically so that they share the window in which plotting occurs.

`tyy` All channels share the same grid, displaying time vs. value. In other respects, same as “ty”.

`xy` The first channel is used as the ‘x channel’, with the additional channels being used as ‘y channels’. The value pairs are plotted in time stamp order. Each ‘y channel’ has its own individual grid, displaying ‘x channel’ value vs. ‘y channel’ value. If ‘s’ is used as a suffix, values for the ‘y channels’ with an error or alarm status have a status code plotted. (The status of the ‘x channel’ isn’t displayed.) The individual plots are sized vertically so that they share the window in which plotting occurs.

`xyy` All channels share the same grid, displaying ‘x channel’ value vs. ‘y channel’ value. In other respects, same as “xy”.

`y` Each channel has its own grid, displaying ‘bin number’ vs. value. If ‘s’ is used as a suffix, values with an error or alarm status have a status code plotted. The individual plots are sized vertically so that they share the window in which plotting occurs. This type is aimed at plotting waveforms. If used for scalar channels, then ‘bin number’ means ‘sample number’ within the group of samples.

`yy` All channels share the same grid, displaying ‘bin number’ vs. value. In other respects, same as “y”.

5.3.9. plotEPS[,type] [filePath]

Produces an Encapsulated PostScript (EPS) file containing the plot of the synchronous data. The plot type is treated the same as for the “plot” command. If filePath is omitted, then the filePath specified on a previous “plotEPS” command is used. The previous contents of the file are erased.

See section 4.5.1. on page 20 for additional details.

5.3.10. plotHC[,type] [printerName]

Produces a hard copy of the plot of the synchronous data. The plot type is treated the same as for the “plot” command. The printer must be a PostScript printer. If printerName is omitted, then the printerName specified on a previous “plotHC” (or “printHC”) command is used.

See section 4.5.1. on page 20 for additional details.

5.3.11. position timeStamp

Changes the retrieval position in the file, so that the next synchronous sample will begin at the specified time and date. If there is no sample exactly matching the specified time stamp, then the position will be at the first sample encountered following that time stamp. The following examples show some of the allowed forms for the time stamp; both a date and time are always required. See section 6. for more details on time stamps.

```
1/2/91 8:13:1
01/02/91 08:13:01
1/2/91 8:13:1.000          (fractional sec must be 3 digits)
10/27/91 8:13:1
```

5.3.12. position +/−hh:mm:ss.msc

Changes the retrieval position by moving forward (+hh:mm:ss.msc) to a later time or backward (−hh:mm:ss.msc) to an earlier time. The ‘.msc’ specifies fractions of a second, and is optional. The specified time is added or subtracted from the time stamp of the last sample in the current group of synchronous samples. For example,

```
position +2:0:0            moves to a time 2 hours later than current position
```

`position -0:15:0`	moves to a time 15 minutes earlier than current position
`position +0:0:0.500`	moves to a time 500 msec later than current position

5.3.13. print [nSamples]

Prints the synchronous data. If nSamples is specified, a new group of synchronous samples is obtained before printing. The data appear in columnar form, with column headings inserted every 60 lines. If the output has been directed to a file with the “dataOut” command, then page breaks are also inserted and the lines are indented 4 spaces. Each value is followed by a one or two character status code. Only the first value is printed for waveform channels.

5.3.14. printHC [printerName]

Produces a hard copy printout of the synchronous data, formatted as described for the “print” command. The printer may be either a PostScript printer or an ordinary printer. If printerName is omitted, then the printerName specified on a previous “printHC” (or “plotHC”) command is used.

5.3.15. rewind

Changes the retrieval position to the beginning of the file.

5.3.16. sampleSet filePath

Selects the named file as the source for retrieval of data. The file must be an AR ‘sample set’ data file. When retrieving from ‘sample set’ files, the “,fill” option on the “channel” command is ignored.

5.3.17. skip nSamples

Skips over the indicated number of synchronous samples.

(Skipping is accomplished by reading synchronous samples, one by one. For large values of “nSamples”, this can be a time-consuming process. In some cases, it may be more efficient to use the “position” command.)

5.3.18. tsTest [nSamples][,interval[,slop]]

Performs tests on the time stamps of synchronous samples. “tsTest” doesn’t actually assemble a group of synchronous samples; instead, it processes synchronous samples one by one beginning at the current position.

The output from the test is handled similar to the output from the “print” command; it can be redirected to a file using the “dataOut” command.

See section 5.6. for a complete discussion of the tests performed and the messages produced.

5.4. Using ARR_kbd

```
% cd directory<nl>                    retrieve from a 'by channel' data file
% ARR_kbd<nl>
  ARR: channel chanA chanB<nl>
  ARR: plot 10<nl>
  ARR: quit<nl>
%

% cd directory<nl>                    retrieve from a 'sample set' data file
% ls arSet/*.txt                      get a list of 'sample set' files
% ARR_kbd<nl>
  ARR: sampleSet arSet/test.txt<nl>
  ARR: fileInfo                       get a list of channels in the file
  ARR: channel POWER:BASE, POWER:LAG30<nl>
  ARR: print 10<nl>
  ARR: plot<nl>
  ARR: ^D
%
```

ARR_kbd is available from the command line with the ARR_kbd command. It accepts no command line options. When first started, ARR_kbd waits for input from the keyboard; the string which prompts for input is " `ARR:` ". It is possible, however, to use a text editor to store commands for ARR_kbd in a text file; the ARR_kbd "source" command causes ARR_kbd to read the file, processing the commands in it in sequence.

Typically, ARR_kbd will be run from the same directory in which AR_kbdis run. The primary exception will be for processing 'sample set' data files.

To exit from ARR_kbd, use the "quit" command or type ^D.

By default, the output from ARR_kbd is directed to the screen. With the "dataOut" command, though, the output from some ARR_kbd commands can be directed to a file.

The "help" command can be used interactively to obtain information about using ARR_kbd and about its commands.

5.4.1. Retrieving Data From Channel Data Files

A 'by channel' data file recieves data from archive requests which specify "OUTPUT: chan:bin". Using such a file requires running ARR_kbd in the same directory in which AR_kbdis run to produce the file. In order to retrieve data from "by channel~ data files, the ARR_kbd "byChannel" command must be used. Any channel selections made prior to using the "byChannel" command are deleted.

ARR_kbd assumes that the path to the file is:
arChan/arChanData.bin

5.4.2. Retrieving Data From Sample Set Files

A 'sample set' data file is one which receives data from archive requests which specify "OUTPUT: set:txt". In order to retrieve data from 'sample set' data files, the ARR_kbd "sampleSet" command must be used. Any channel selections made prior to using the "sampleSet" command are deleted.

'Sample set' data files produced by AR_kbdreside in the "arSet" subdirectory. The name of the file is the name of the archive request followed by ".txt" as an extension. Thus, for a request named "test", the path name for the corresponding 'sample set' data file is "arSet/test.txt".

For "user owned" 'sample set' data files, ARR_kbd will usually be run in the directory containing the file and the path name will be simply the name of the file.

When a channel is added to a synchronous sample specification, ARR_kbd checks in the first sample in the 'sample set' data file; if the channel isn't found, then it isn't added to the list. This has strong impli-

cations for adding a channel to a “set:txt” archive request—the ‘sample set’ data file should be started fresh in such cases. When retrieving from ‘sample set’ data files, ARR_kbd ignores the “,fill” option on the “channel” command.

5.4.3. running ARR_kbd in the background

ARR_kbd can be “moved to the background” to allow using the shell for other purposes, without stopping the execution of ARR_kbd. The method for doing this differs for UNIX and VxWorks.

Under UNIX, moving ARR_kbd to the background can be done only when running under the csh shell. (This allows the shell to be used for other purposes, but exiting the shell aborts ARR_kbd.) First, type a ^Z to interrupt execution. Then use the csh command “`bg %ARR_kbd`” to move ARR_kbd into the background. To move ARR_kbd into the foreground, so that it will take input from the keyboard once more, use the csh command “`fg %ARR_kbd`”. After the fg command is complete (as indicated by csh typing a line with ARR_kbd), ARR_kbd is waiting for input, but it hasn’t reissued a prompt string; if desired, a Return can be typed in order to make the prompt appear. At the completion of this process, the screen will look something like the following:

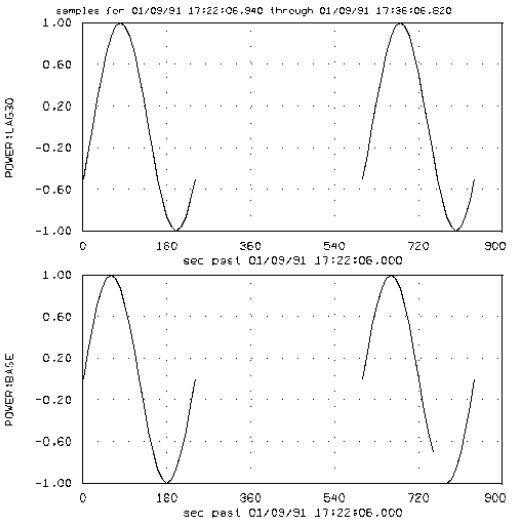
```
   ARR:  ^Z
Stopped
% bg %ARR_kbd<nl>
[1]    ARR_kbd &
% do some other work
  ...
% fg %ARR_kbd<nl>
ARR_kbd
<nl>
  ARR:
```

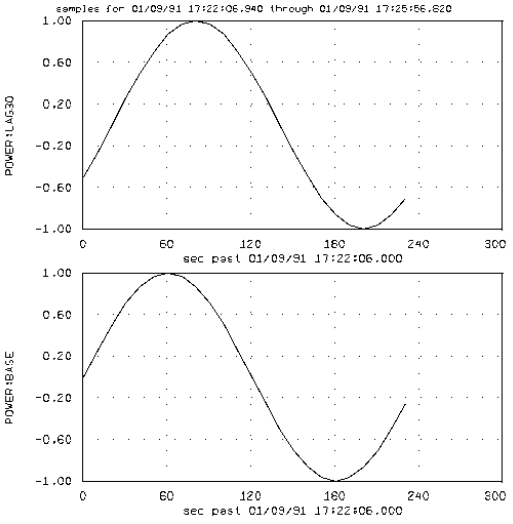
Under VxWorks, the shell doesn’t support the concept of “moving into the background”. Instead, the ARR_kbd “bg” command must be used. After executing the bg command, ARR_kbd is running detached from the shell. To resume interactions with ARR_kbd, type the ARR_kbd command at the shell prompt.

5.4.4. examples

This example shows a full session using ARR_kbd, covering getting to the directory, finding out what files are available and what data are available in a particular file. All of the data are listed and plotted, and then the first 24 samples are chosen for further processing. Without knowing how many samples there were in the file, “print 100” was used as an upper bound of interest. The first plot came from the first “plot” command. A “rewind” command was needed to start fresh in getting and plotting 24 samples with “plot 24”.

```
% cd ~dvl/ar
% ls *.txt
arSet/sine.txt     arSet/test1.txt    arSet/test2.txt    arSet/testSet.txt
% ARR_kbd
  ARR:  sampleSet arSet/sine.txt
  ARR:  fileInfo
data from 01/09/91 17:22:06.940 through 01/09/91 17:36:06.820
POWER:BASE 1 DBF_FLOAT
POWER:LAG30 1 DBF_FLOAT
POWER:LAG60 1 DBF_FLOAT
POWER:LAG90 1 DBF_FLOAT
  ARR:  channel POWER:BASE POWER:LAG30
  ARR:  print 100
                          POWER:BASE    POWER:LAG30
  01/09/91 17:22:06.940      0.0000 R      -0.5000 R
  01/09/91 17:22:16.820      0.2590 O      -0.2590 O
  01/09/91 17:22:26.820      0.5000 O       0.0000 O
  01/09/91 17:22:36.820      0.7070 O       0.2590 O
  01/09/91 17:22:46.820      0.8660 O       0.5000 O
  01/09/91 17:22:56.820      0.9660 O       0.7070 O
  01/09/91 17:23:06.820      1.0000 O       0.8660 O
  01/09/91 17:23:16.820      0.9660 O       0.9660 O
    ...
  01/09/91 17:24:26.820     -0.5000 O       0.0000 O
  01/09/91 17:24:36.820     -0.7070 O      -0.2590 O
  01/09/91 17:24:46.820     -0.8660 O      -0.5000 O
  01/09/91 17:24:56.820     -0.9660 O      -0.7070 O
  01/09/91 17:25:06.820     -1.0000 O      -0.8660 O
  01/09/91 17:25:16.820     -0.9660 O      -0.9660 O
  01/09/91 17:25:26.820     -0.8660 O      -1.0000 O
  01/09/91 17:25:36.820     -0.7070 O      -0.9660 O
  01/09/91 17:25:46.820     -0.5000 O      -0.8660 O
  01/09/91 17:25:56.820     -0.2590 O      -0.7070 O
  01/09/91 17:26:06.820      0.0000 O      -0.5000 O
  01/09/91 17:32:06.940      0.0000 R      -0.5000 R
  01/09/91 17:32:16.820      0.2590 O      -0.2590 O
    ...
  01/09/91 17:33:06.820      1.0000 O       0.8660 O
  01/09/91 17:33:16.820      0.9660 O       0.9660 O
  01/09/91 17:33:26.820      0.8660 O       1.0000 O
  01/09/91 17:33:36.820      0.7070 O       0.9660 O
  01/09/91 17:33:46.820      0.5000 O       0.8660 O
  01/09/91 17:33:56.820      0.2590 O       0.7070 O
  01/09/91 17:34:06.820      0.0000 O       0.5000 O
  01/09/91 17:34:16.820     -0.2590 O       0.2590 O
  01/09/91 17:34:26.820     -0.5000 O       0.0000 O
  01/09/91 17:34:36.820     -0.7070 O      -0.2590 O
* 01/09/91 17:34:46.820     no_data MN     -0.5000 O
  01/09/91 17:34:56.820     -0.9660 RR     -0.7070 O
  01/09/91 17:35:06.820     -1.0000 O      -0.8660 O
  01/09/91 17:35:16.820     -0.9660 O      -0.9660 O
  01/09/91 17:35:26.820     -0.8660 O      -1.0000 O
  01/09/91 17:35:36.820     -0.7070 O      -0.9660 O
  01/09/91 17:35:46.820     -0.5000 O      -0.8660 O
  01/09/91 17:35:56.820     -0.2590 O      -0.7070 O
  01/09/91 17:36:06.820      0.0000 O      -0.5000 O
  ARR:  plot
  ARR:  rewind
  ARR:  plot 24
  ARR:  quit
%
```

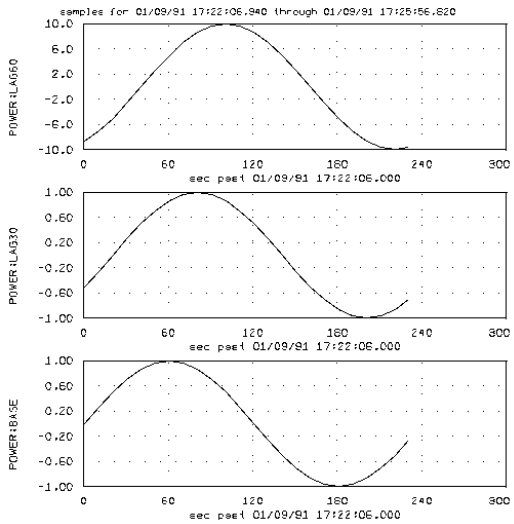


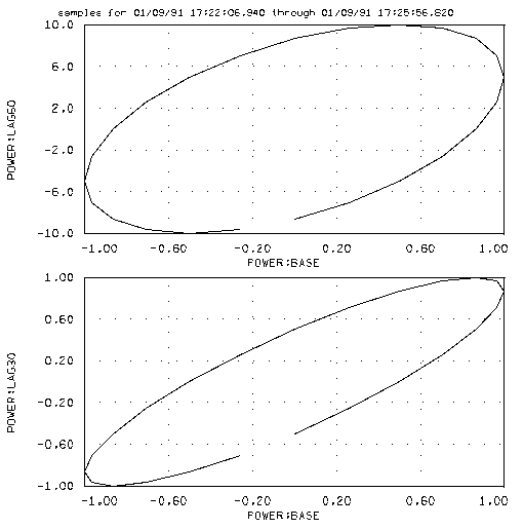


This example shows the retrieval of data for three sinusoidal channels and demonstrates some of the flexibility of repeatedly processing the same group of synchronous samples.

```
% ARR_kbd
  ARR: sampleSet arSet/sine.txt
  ARR: channel POWER:BASE POWER:LAG30 POWER:LAG60
  ARR: plot 24
  ARR: plot,xy
  ARR: plotHC cx1
  ARR: ^D
%
```

The first plot on the right was produced by the “plot 24” command. The second plot was produced by the “plot,xy” command. The “plotHC cx1” command produced a hard copy of the x vs. y plot.





5.5. export—writing synchronous samples in “foreign” formats

The “export” command allows producing synchronous sample data in forms which are compatible with other programs.

The “dataOut” command can be used to direct output from the “export” command into a file. The format of exported output depends on the export option which is used. Since data written when “dataOut” has been used are appended at the end of existing files, it may be necessary to delete such files before using the combination of “dataOut” and “export” commands.

5.5.1. example

This example shows the retrieval of data for four sinusoidal channels and the writing of the resulting synchronous samples to a file which will be used as input to a spreadsheet program.

```
% rm arSet/sine.txt                                  remove pre-existing file, if any
% ARR_kbd
  ARR:  sampleSet arSet/sine.txt
  ARR:  channel POWER:BASE POWER:LAG30 POWER:LAG60 POWER:LAG90
  ARR:  dataOut arSet/sine.ss                        spreadsheet output will go to file
  ARR:  export,ss 24
  ARR:  ^D
%
```

5.5.2. ss option

The ss option produces output for use by spreadsheet programs. It is intended for use with scalar channels; if it is used with array channels, only the first value of the array is output. The output is organized into rows and columns. Each row is ended with a ‘newline’ character; within a row, the columns are separated by a ‘tab’ character. See section 4.7.1. for an example of spreadsheet output.

5.5.3. ssStat option

The ssStat option is similar to the ss option, with the additional feature of adding a status column before each value column. The status is a one or two character status code, as discussed in section 4.1.3. on page 12. See section 4.7.1. for an example of spreadsheet output with status codes.

5.6. tsTest—time stamp test

6. TIME STAMPS

In the control system, the time stamp changes at each system trigger. This time stamp is associated with 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; this feature allows ARR to retrieve samples which are synchronous. 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*.

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
`mm` is the minute within the hour
`ss` is the second within the minute
`msc` is the milli−second within the second

The only nonintuitive display of time occurs during the switch from daylight savings time back to standard time. In the USA, this switch occurs in 1991 on October 27 at 2 a.m., local daylight time—at that time, the time becomes 1 a.m., local standard time. The result is a 2 hour interval during which numeric clock times repeat. 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	`01/02/91 00:02:03.400`	
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	`10/27/91 d01:2:3.759`	‘d’ and ‘s’ placement not strict
legal	`Oct 27, 1991 01s2:3.759`	

7. arAccessLib—ARCHIVE DATA FILE ACCESS LIBRARY

8. INSTALLATION OF SOFTWARE

8.1. Installing the Software

The directory in which the software will be run should contain the link created by the following command. (Many application directories will already have this link.)

```
% ln -s ~gta/release/getrel
```

The first time the software is installed in the directory, the following commands should be executed. The release number of EPICS will be something of the form 2.0.0 to select the desired release.

```
% getrel release_number
% mkdir arReq
% mkdir arSet
% mkdir arChan
```

After the first time, only the '`getrel`' command is needed.

8.2. Effects Of Changing Versions

The release of AR must match the current EPICS release, so that Channel Access will be compatible; AR will not be able to acquire data from IOCs with incompatible versions of Channel Access.

8.3. Platforms Supported

AR runs under SunOS and VxWorks.

ARR runs under SunOS with X11.

ARR_kbd runs under SunOS with either X11 or SunView.