MASTER

Introduction to Q

Martin Kellogg
Michael M. Minor
Sally Shlaer*
Nancy Spencer**
Richard F. Thomas, Jr.*
Henri van der Beken†

- *Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720.
**Stanford Linear Accelerator Center, Stanford University, P. O. Box 4349, Stanford, CA 94305.
†Grand Accelerator d'Ions Lourdes, Saclay, France.

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

FOREWORD

This manual is one of a series describing the components of the Q general-purpose data-acquisition system. This system, developed at the Clinton P. Anderson Meson Physics Facility, is designed to be useful in a variety of experimental situations for managing the acquisition and display of experimental data. It runs on the Digital Equipment Corporation (DEC) PDP-11 family of computers running the RSX-11D operating system and equipped with Microprogrammed Branch Drivers (MBDs) and CAMAC branch highway systems.

The Q system is designed to allow high-speed data acquisition and recording. It is driven primarily by the occurrence of events in the experimental apparatus. It is modular in construction and is easy to use, understand, and modify. Its components are an MBD program for data taking, an RSX handler for data recording and distribution, a control and analysis task (the analyzer task), a histogram entry and retrieval system, histogram plotting and dot plotting packages, and an operator control system.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

CONTENTS

1.0	INTRODUCTION	1
1.1	About This Manual.	1
1.2	Purpose of the Q Project	2
1.3	History of the Q Project	3
1.4	Computer System Required to Run Q.	5
2.0	MAJOR SOFTWARE COMPONENTS OF A Q SYSTEM.	6
2.1	The Analyzer	8
2.2	The QAL Compiler	8
2.3	Q: the Real-Time Data-Acquisition Subsystem.	9
2.4	X: the Replay Subsystem.	9
2.5	H: the Histogram Subsystem	9
	2.5.1 The Histogram Management System.	9
	2.5.2 The Histogram Display System	9
2.6	Utilities.	10
2.7	Optional User-Supplied Software.	10
2.8	System Subroutine Library: MDLIB	10
3.0	THE TRIGGER MODULE	10
4.0	HARDWARE DATA PATHS DURING DATA ACQUISITION.	12
5.0	HOW DATA ARE ACQUIRED.	13
6.0	DATA BUFFERING AND EVENT DISTRIBUTION.	13
7.0	DOCUMENTATION GUIDE.	18
7.1	An RSX-11D Primer.	19
7.2	CAMAC Systems: A Pedestrian's Guide.	19
7.3	QAL and the Analyzer Task.	19
7.4	QMODULE: CAMAC Modules Recognized by the QAL Compiler.	19
7.5	Q Handbook	19
7.6	A System for Histogram Entry, Retrieval and Plotting.	19
7.7	MDLIB: System Subroutine Library	20
7.8	MBD Operations with MBDSTD and the RSX-11 CC Handler	20
7.9	An MBD Primer.	20
7.10	Programming the MBD Model II	20
	GLOSSARY.	21
	ACKNOWLEDGMENTS	23
	REFERENCES.	24
	INDEX	26

INTRODUCTION TO Q

by

Martin Kellogg, Michael M. Minor, Sally Shlaer, Nancy Spencer,
Richard F. Thomas, Jr., and Henri van der Beken

ABSTRACT

This manual presents an overview of the Q data-acquisition system. Its purposes are to assist an experimenter in determining whether the system is suited to his needs and to provide an introduction to the more detailed manuals. This system employs a powerful technique for dealing with multiprocessor systems by generating code and communication protocol for two processors from a single source program in a high-level language. Some details of experimental event buffering and distribution are discussed.

1.0 INTRODUCTION

1.1 About This Manual

This manual is intended to provide an overview of the Q general-purpose data-acquisition system. The components of the system and the manner in which they fit together and function are described at a conceptual level. It is our intent to supply sufficient information so that a potential user of the system can begin to assess how well the system meets his needs. To this end a detailed exposition of experimental event buffering and distribution is given. In-depth discussions of other aspects of the system are presented in documents that are referenced here and the user is urged to read those documents for further details. Finally, a glossary defining the terminology used throughout the system is provided.

It is assumed that the reader has a nodding acquaintance with real-time multitasking computer operating systems (if not, see Sec. 7) and a reasonable understanding of the computing problems associated with high-speed data acquisition.

1.2 Purpose of the Q Project

The purpose of the Q Project is to provide a high-performance general-purpose experimental data-acquisition system. Q is designed to provide those facilities routinely required by all experimenters, such as the acquisition and recording on magnetic tape of raw data, histogramming of raw data and calculated quantities, and display of histograms. In the design of Q, great attention was paid to the division of labor required to perform a modern physics experiment effectively. Q attempts to provide sufficient specialized computing support so that the experimenter can be free to concentrate on physics without distraction by complex computing issues involved with buffering schemes, multiprocessor communication protocols, memory allocation methods, queuing systems, and the like. In the majority of Q experiments, the only code a user needs to supply is that specific to the physics of his particular experiment, for example, subroutines that process raw data from his detector system to identify particle types or to calculate missing masses. These aspects of the computing problem are readily dealt with in FORTRAN; hence it is necessary that at least one member of the experimental team have some fluency in that language. Because a general-purpose system is unlikely to suit all needs equally well, a major goal of the project has been to provide a modular system that is relatively easy to understand, modify, and maintain. For those experimenters who require more sophisticated or highly specialized processing, the source code for Q is made available so that many of its components can serve as building blocks for such systems.

The Q system consists of a large number of components on disk; however, only those components that an experimenter actually uses are loaded into primary memory. Furthermore, a component will normally occupy memory only during the short time necessary to perform its function. In this way a user can select those components of Q that are relevant to his application, ignore those that are not, and is not forced to carry the overhead for a feature he does not wish to use or uses infrequently. This structure differs markedly from that commonly found in single-user environments, where a single program module contains diverse features and the user must modify the program to remove those features that are not of interest. With Q, if a user decides he does not wish to use a certain subsystem, he simply avoids reference to it. He can also delete from his disk the task modules supporting that subsystem.

Q is an ongoing project that attempts to incorporate desirable new features in an orderly fashion so as not to invalidate existing code supplied by Q experimenters. As experience is gained with its operation, segments of the system continue to be redesigned to provide more reliable operation, higher performance, and better user interfaces. Users are invited to make suggestions, comments, or contributions, which will be investigated, as time is available, with a view toward incorporation into the system.

1.3 History of the Q Project

In 1970, a design study for data-acquisition systems at the Los Alamos Meson Physics Facility (LAMPF)¹ recommended that a single family of computers be selected to support data acquisition at LAMPF. The computer selected was the Digital Equipment Corporation (DEC) PDP-11. Additionally, the CAMAC system² was specified as the instrumentation standard to be used. To provide an interface between the PDP-11 computers and a multicrate CAMAC system, the Microprogrammed Branch Driver (MBD)³ was developed and became a LAMPF standard data-acquisition hardware item.

Several software systems were developed based on this hardware configuration and the DEC-supplied Disk Operating System (DOS). However, with the advent of new computers in the PDP-11 family, DEC offered new operating systems, which could take advantage of the new hardware, and withdrew its support from DOS, which could not. Thus, many of the systems that had been developed based on DOS were forced towards obsolescence. In addition, such systems were often developed by single individuals who, because of time constraints, had to limit their attention to the immediate requirements of their own experiments. As experiments began to be mounted faster than data-acquisition systems could be provided, attempts were made to recycle these DOS-based systems for the next experiment. When the computing needs of the new experiment were similar to those of the last, this effort was successful; when they were dissimilar, the reliability and speed of the system left much to be desired. The original implementor of such a system soon found himself indispensable, being the only person who understood the code sufficiently to modify or maintain it. It became apparent that a collection of standard, centrally supported software would be desirable, just as standard, centrally supported hardware had proved to be.

Early in 1974, we initiated feasibility studies, which showed the following:

The logical flow of an experiment through the CAMAC and computer systems could be defined independently of the experiment.

- . Very high data rates could be obtained by preloading the MBD with code well conditioned to its task of transferring raw data from CAMAC to the PDP-11 memory. (The first measurements showed a data rate of 6 microseconds/word during the actual transfer.)
- . The MBD requirements of an experiment could be expressed in a language easily mastered by a FORTRAN programmer. A compiler could be developed to produce the required well-conditioned MBD code from this language.
- . The time required to switch control between tasks in RSX-11D (DEC's real-time computer operating system) could be reduced two orders of magnitude (to the speed of a hardware interrupt) in certain carefully restricted cases.

All the above were thought to be necessary components of a workable general solution. To determine what else would be required, an informal survey of the DOS-based systems was made in mid-1974. The survey revealed that, while each system had certain unique properties or areas in which it excelled, all systems were similar from the point of view of software system design. Thus, while there had been duplication of effort, we took comfort from this similarity, interpreting it as evidence that the problem of data acquisition for LAMPF was indeed amenable to general solution.

Armed with this experience and the luxury of computers idled by the Great Shutdown, we began to design the data-acquisition system in late 1974. We soon decided that it would be based on DEC's RSX-11D, a real-time multiprogramming PDP-11 operating system. We would make extensive use of the RSX-11D multitasking facility to create a highly modular, and therefore maintainable, system. Our goal was to achieve the performance seen during the feasibility study, only this time in the context of a fully integrated system. In production, the performance would be limited either by data-recording or CAMAC speeds. Therefore, several of the components at the heart of the system were designed with particular emphasis on speed and were dependent on the RSX-11D operating system structure.

It had become apparent early in the feasibility study that a standardized experiment-independent hardware interface to the software system would be desirable. Such a device would provide a central control point where the event signals could be sorted at high speed. The LAMPF Trigger Module (see Sec. 3.0) was designed to provide this interface, and it is an integral part of the data-acquisition system.

We released a preliminary version of the data-acquisition system in the summer of 1975 under the name "Q."* Since then we have gained considerable experience with its use. Some experimenters have used all features as provided, some have used only a subset of the features, and some have modified extensively or replaced entirely certain subsystems. Experience to date indicates that an experiment requiring little or no modification to the basic system becomes fully operational within a couple of weeks after the detectors and CAMAC hardware systems have been defined. If extensive modifications to Q are required, several months may elapse before the experiment is running in close to final form. Very few experiments seem to fall into the latter category. In either case, the software provided by the experimenter will probably continue to be developed throughout the data-acquisition phase of the experiment as the user explores his data and develops more powerful methods of extracting information from them.

Recently, with the Real Time Systems Group at Lawrence Berkeley Laboratory, we have tried to make Q available under the RSX-11M operating system. DEC has "stabilized" the RSX-11D operating system and will provide no major enhancements to this system. However, DEC is developing RSX-11M further and the ability to run Q on this operating system as well would be advantageous. Because DEC claimed that RSX-11M was a subset of RSX-11D and because Q was written to operate mostly "within the rules" of RSX-11D, we expected few problems in the conversion effort. However, a number of difficulties have been discovered. Consequently, the conversion effort has been suspended for the time being and Q is available only under the RSX-11D operating system.

1.4 Computer System Required to Run Q

The following configuration is required to run Q:

1. A PDP-11 computer capable of running the RSX-11D

*Historical Note: The name "Q" was chosen because it is short and easy to spell. It has no particular meaning and is definitely not an acronym. Its choice was strongly influenced by the following naming issue: in accordance with our goal of modularity, we would be creating a large number of keyboard command tasks, all of which had to be named. We wanted these commands to fall into a coherent perceptual group for the benefit of our users; in addition, it was necessary to avoid names that duplicated those in the set supplied with the operating system. Because DEC had not implemented any command beginning with the letter Q, we chose to distinguish our commands by the presence of this hopefully unpopular initial letter. It was far too late to retreat when DEC introduced the most obvious command of all: QUE.

operating system. A computer having at least the features of a PDP-11/34 fulfills this requirement. An FP-11 or equivalent unit is strongly recommended for any computer for which a hardware floating point option exists.

2. The RSX-11D operating system, Version 6.2. The F4P FORTRAN compiler is required. For computer systems that do not include appropriate floating point hardware, software routines are available to emulate the hardware.
3. We recommend at least 96K words of memory. Experience has shown, however, that using the full 128K words of memory can help avoid a number of problems, especially if a large amount of core is to be used for histograms. A minimal system can operate in 80K if the user needs only a few histograms and requires no complex computation.*
4. A CAMAC system.
5. An MED (CAMAC branch driver).
6. The LAMPF Trigger Module.
7. A magnetic tape unit (if recording of data is desired).
8. Five megabytes of disk storage for the operating system, Q routines, and user programs.
9. A Tektronix 4000 series graphic terminal (if displays are required).

2.0 MAJOR SOFTWARE COMPONENTS OF A Q SYSTEM

The major components, described briefly below and pictured in Fig. 1, are relatively independent of one another. These components are:

1. The analyzer task
2. QAL: the Q-system compiler
3. Q: the real-time system
4. X: the replay system

*The July 1976 release of Q (Q776) was run (successfully but with difficulty) on a 56K system for a single experiment. We have had no experience with more recent releases of Q on small systems.

5. H: the histogram management and display system
6. A collection of utility programs
7. An optional set of user-supplied tasks to provide experiment-dependent support.
8. MDLIB: The subroutine library; this is not a component in the same sense as the others, but contains subroutines used by the other components.

Most Q experiments use all the component subsystems; this is not necessary, however, and the subsystems may be regarded as a set of building blocks. For example, one can use the real-time acquisition subsystem (Q) with or without the histogramming (H) subsystem or the replay (X) subsystem. An experimenter may substitute his own real-time, histogramming, or replay subsystem if he prefers. The histogram management and display system can be used in other tasks such as Monte Carlo simulations. Any ties that exist between subsystems are loose; for example, the real-time acquisition and replay subsystems are tied only through the format of data on the magnetic tape. Tapes written by the real-time acquisition system can be analyzed by an unrelated data analysis facility. Alternatively, tapes written in Q format by other data-acquisition systems can be played back with the replay subsystem. In this case, the user must supply an analyzer to specify the format of the data on the tape and the analysis to be performed.

Fig. 1.
Major components of the Q system.

Modularity of design also prevails within each subsystem, where one finds components that interface intimately with RSX-11 on the one hand and with the analyzer on the other. Each subsystem (Q, X, and H) contains, among other things, tasks that allow it to be controlled from the keyboard by the experimenter. The list of keyboard commands can be extended easily to provide extra facilities for a specific experiment, and existing commands can be modified so long as their interfaces to the remainder of the system are preserved.

2.1 The Analyzer

The analyzer is the most important part of the Q system and the one that each user must supply to specify his experiment. It indicates what data should be acquired from CAMAC in response to a particular trigger and what, if any, analysis is required on these data. Histogram entries are normally made by the analyzer; a tape of filtered data can be recorded by it. The analyzer, because it is experiment-specific, will vary greatly in complexity from one experiment to another. A special language, QAL (Q Analyzer Language), has been provided to facilitate writing an analyzer and to impose a standard structure for the analyzer interface to other system components. The experimenter who wants to write an analyzer needs to have a good conceptual understanding of CAMAC, to be fairly fluent in FORTRAN, and to be able to use the RSX-11 utilities such as the text editor, FORTRAN compiler, and task builder. A manual entitled "An RSX-11D Primer"⁵ familiarizes the user with the RSX-11D operating system. "CAMAC Systems: A Pedestrian's Guide"⁶ provides a good introduction to CAMAC. For a detailed exposition on how to write an analyzer and its role in the Q system, see the manual, "QAL and the Analyzer Task."⁷

2.2 The QAL Compiler

QAL is a compiler for the Q Analyzer Language, which is designed to facilitate the preparation of an analyzer. It functions in a manner similar to the FORTRAN compiler to produce object code from special analyzer statements. QAL employs a powerful technique for dealing with multiprocessor systems by generating code and communication protocol for both the PDP-11 and the MBD from a single source program in a high-level language. For a detailed description of the QAL compiler, see "QAL and the Analyzer Task."⁷

2.3 Q: The Real-Time Data-Acquisition Subsystem

This subsystem acquires the data for the experiment, records the raw data on magnetic tape if the user so chooses, supplies the data for each experimental event to the analyzer for processing, and allows keyboard control of the experiment. Further details on this subsystem are provided in Secs. 4.0, 5.0, and 6.0 and in the manual, "Q Handbook."⁸

2.4 X: The Replay Subsystem

This subsystem reads a Q-format tape, supplies the data for each experimental event to the analyzer for processing, and allows keyboard control of the "replayed" experiment. The same analyzer can be used for replay as was used to acquire the data, or more extensive processing may be added to the analyzer based on the needs of the experiment. The user interface to the replay subsystem is similar to that for the real-time acquisition subsystem. For further details see "Q Handbook."⁸

2.5 H: The Histogram Subsystem

2.5.1 The Histogram Management System

This system allows the user to create, increment, save, retrieve, clear, and delete histograms. Any task needing histograms, not only an analyzer, can use its facilities. Keyboard commands are available for user control of this system. For details see "A System for Histogram Entry, Retrieval, and Plotting."⁹

2.5.2 The Histogram Display System

This system allows the display on a Tektronix 4000 series graphics terminal of stored histograms created with the histogram management system. Real-time scatter plots are also supported. This display system can plot histograms built with the histogram management system only; it is, however, an independent subsystem and the user can augment or replace it while keeping the histogram management facility. The plotting facilities can be controlled from the keyboard. For further details on this subsystem, see "A System for Histogram Entry, Retrieval, and Plotting."⁹

2.6 Utilities

Several utility programs that are useful in conjunction with Q are available. They provide such facilities as listing a Q-format tape, copying and listing any magnetic tape, listing the contents of the MBD, executing single CAMAC commands in any module on the CAMAC branch, and displaying the memory allocation of an RSX-11 partition.⁸ For details on the use of these utilities see "Q Handbook."

2.7 Optional User-Supplied Software

The user may provide tasks in addition to the analyzer to support his experiment. Provision is made for optional user-supplied tasks to be called when data taking is initiated, suspended, resumed, or completed, or when an end-of-tape sticker is sensed on a data tape. See "QAL and the Analyzer Task"⁷ and "Q Handbook"⁸ for more details on these particular tasks. There are no limits on the number or complexity of tasks supplied by the user, except those constraints implied by memory size and timing considerations.

2.8 System Subroutine Library: MDLIB

This library contains the subroutines necessary for the analyzer (or other tasks) to interface to the various subsystems of Q. It also provides routines to support graphics on the Tektronix 4000 series graphics terminals, FORTRAN interfaces to sophisticated RSX-11 capabilities, and other useful routines, most of which are available for use by the experimenters. See "MDLIB: System Subroutine Library"¹⁰ for descriptions of the available routines.

3.0 THE TRIGGER MODULE

The LAMPF CAMAC trigger module is a required component in every Q system. It is used to provide a standardized experiment-independent hardware logical interface to the remainder of the real-time data-acquisition system.

The trigger module allows for 32 possible events, each with a separate "trigger request" bit. Events are priority-ordered with event 31 having the highest priority and event 0 the lowest. The trigger request bit for any event (0-31) can be set or cleared by

a CAMAC dataway command. Trigger request bits for events 0 through 23 can be set by Look-at-Me signals (LAM's) from modules within the same crate as the trigger module. The bits for events 0 through 15 can be set by pushbuttons and those for events 4 through 11 can be set by external NIM signals. For each trigger request bit there is a corresponding enable bit in the trigger module. These enable bits can be set and cleared by CAMAC dataway commands. A trigger request bit can be set only when the corresponding enable bit is set. In addition, enable/disable switches are provided on the front panel of the trigger module for events 4-11. See Fig. 2.

After considering the different triggering capabilities of each event type, the user chooses those events that best suit his experimental needs. He must select at least one event type; he can use up to 24 different event types if he wishes. Events 24 through 31 are reserved for synchronization and are not available to the user. Whenever a trigger request bit and the corresponding enable bit are set and the LAM on the trigger module is enabled, a CAMAC LAM is generated by the trigger module. To interface the trigger module to Q, the LAM from the trigger module must be wired to graded LAM bit number 23, which drives MBD Channel 6. See "Q Handbook"⁸ for further details.

In response to the LAM from the trigger module, the MBD obtains from the trigger module the encoded number of the highest priority trigger set. This becomes the event number used throughout the Q system. The MBD performs a sequence of CAMAC operations that the experimenter has associated with this particular event trigger and then clears the trigger request bit for that event, allowing other events to be processed. Events are not interruptible; if the MBD is in the process of acquiring data for a lower priority event when a higher priority event is triggered, the higher priority event will wait until the acquisition for the lower priority event is complete. For example, suppose that while event 10 is being serviced, events 7 and 15 are both triggered. When servicing for event 10 is complete, event 15, the higher priority event now requested, will be processed.

4.0 HARDWARE DATA PATHS DURING DATA ACQUISITION

Fig. 3 displays the path of experimental data during acquisition. To start the acquisition of data, the experiment generates a signal to the trigger module to trigger an event.* The trigger module then generates a LAM to the MBD. The MBD determines which event was triggered and proceeds to acquire the CAMAC data for that particular event and to transfer them to the PDP-11.

In response to an event trigger, the MBD can access any CAMAC module in the system, including the trigger module itself. Therefore, one event trigger can be used to trigger another event synchronously.

*Note that all events can be triggered via a CAMAC dataway command and that events 0 through 15 can be triggered by pushbuttons. Therefore, for checkout, the user need not have available the actual signal that will be used during the experiment to start the data-acquisition sequence.

Fig. 2.

Event inputs of the trigger module.

Fig. 3.

Path of data during acquisition.

The PDP-11 can process the data in various ways (see Secs. 5.0 and 6.0) and, if the user requests it, can write all the raw data acquired to magnetic tape. The PDP-11 can also communicate with the MBD (and therefore the CAMAC system) asynchronously with external event triggers. All events can be triggered by a CAMAC dataway command; therefore, events can be triggered from the PDP-11 under the user's control.

5.0 HOW DATA ARE ACQUIRED

Fig. 4 shows how data are acquired by the real-time data-acquisition subsystem. Associated with each event trigger is an event specification or recipe, which resides in the MBD and which tells what operations in the MBD/CAMAC system should be performed in response to this particular trigger. The event data are produced as a result of executing this event-specific procedure whenever a trigger associated with the event is received by the MBD. The event specifications are supplied by the user in his analyzer.

Data acquired in response to an event trigger are transferred by the MBD into large buffers in the data-acquisition handler (QA) in the PDP-11. Whenever the PDP-11 is alerted by the MBD that a buffer is full, a header containing such information as the date, time, and run number is appended to this buffer and the augmented buffer is written on magnetic tape (if the user requested taping). The data then are picked out of the buffer event by event in accordance with a sampling algorithm controlled (to some extent) by the user and are sent to his analyzer for processing. (See Sec. 6.0 and "Q Handbook"⁸ for further details on data buffering and event distribution.)

6.0 DATA BUFFERING AND EVENT DISTRIBUTION

This section consists of a detailed description of the buffering and event distribution capabilities provided by Q. From experience we have learned that this information is required if a potential user is to assess the applicability of the Q system to his needs. Additional information on these topics is presented in "Q Handbook."⁸

The first layer of data buffering is optional and, if present, takes place in the MBD. When MBD buffering is not in effect, each data word is transferred to the PDP-11 as soon as it is acquired. If MBD buffering is in use, the data words are stored temporarily in the MBD for transfer to the PDP-11 later,

either on request or when the MBD buffer fills. Thus while buffered MBD operation actually takes more time in the MBD, the load is distributed in time so that a net performance increase will result under certain experimental conditions. The relevant factors include both burst and average data and event rates, beam time structure, and absolute time to acquire raw event data. As a guideline, one can expect that the highest performance will result if one chooses unbuffered MBD code for a beam with no time structure and buffered code for a beam with the LAMPF time structure (500 microsecond pulses at 120/second). If the beam has still a different time structure, it may be best to experiment a bit; this is quite easy since the experimenter is allowed to choose the style of MBD operation at run time. The choice of buffered or unbuffered MBD operation has no effect on any aspect of the system other than performance.

Fig. 4.
Acquisition and distribution of event data.

The second layer of buffering takes place in the large data buffers in PDP-11 memory. The MED can fill these buffers in either of two modes: the single-event-per-buffer mode or the multiple-event-per-buffer mode. These modes describe only the buffering of data in the PDP-11 and are independent of the buffering that may be taking place in the MBD. In the single-event-per-buffer mode, the MBD selects a new PDP-11 buffer at the end of every event and alerts the PDP-11 via an interrupt that this operation has been accomplished. In the multiple-event-per-buffer mode, the PDP-11 is not interrupted until an entire large buffer in the PDP-11 (which contains many events) is filled. The multiple-event mode is normally used during the active part of a data-acquisition run, because it takes maximum advantage of the multiprocessor aspects of the system and therefore gives the highest performance. The single-event mode is used between runs or when a run has been suspended to allow prompt response to the control "events" that occur during these intervals. In any case, the PDP-11 can perform other processing, such as writing previously filled buffers to magnetic tape or analyzing previous events, while the MBD is filling the current PDP-11 buffer. During operation of Q, the switching back and forth between single-event and multiple-event-per-buffer modes is done automatically.

The large buffers in the PDP-11 contain only events. Events are variable in length and the length can vary even for the same event number; there is, however, a limit of 255 words per event.* A buffer in the PDP-11 appears as follows.

*If the 255-word limit presents a problem, the user can arrange to trigger an auxiliary event at the end of any event specification requiring further data.

The analyzer receives only a single event at a time by a process known as "event distribution." The format of a single event is:

TAG WORD

DATA

EVENT No.	WORD COUNT
EVENT DATA IN THE SAME ORDER AND FORMAT RECEIVED FROM CAMAC	

The tag word or event header word consists of two bytes. The high byte contains the event number as determined from the trigger causing the event; the low byte contains the number of words of event data to follow (not including the tag).

There are three possible processing options available for an event class in the PDP-11: "must process," "may process," and "no process." If an event has been specified by the experimenter as "must process," all events of that class are distributed to his analyzer, even if data acquisition must be slowed down to do so.* "Must process" is usually specified if the user wants to perform his own taping of events or for those events that are used for experiment control. If "no process" is specified for an event, no events of that type are distributed to the analyzer. If "may process" is specified for an event, the user's analyzer receives a fraction of that type of event. The fraction depends on the rate at which events are being acquired and the processing speed of the analyzer. In "may process," every effort is made to allow data acquisition to proceed at the highest possible rate and the limiting factor should be recording speed or CAMAC acquisition speed, not event-processing speed. The analyzer initially specifies for each event whether its processing option is to be "must process" or "may process." A keyboard command can subsequently set the option for any event to "must process," "may process," or "no process." The processing option for an event class has no effect on the recording of data: If taping is enabled, all events are taped; if taping is disabled, no events are taped.

*Actually, data acquisition is stopped when there is no buffer space available in the PDP-11 to allow new events to be acquired; at any other time it runs at full speed.

Whenever a data buffer in the PDP-11 has been filled by the MBD, it is ready for taping and event distribution. The buffer is processed by the data-acquisition handler (QA) as follows:

1. If taping is enabled, the buffer is queued to be written to magnetic tape.
2. The buffer is queued for "normal" event distribution. When a buffer is set for "normal" event distribution, all "may process" and "must process" events in the buffer are distributed to the analyzer.
3. If event distribution has not yet been completed in any of the previous buffers, they are set for "hurry-up" distribution. Whenever a buffer is set for "hurry-up" event distribution, only "must process" events in the buffer are distributed.

Events are distributed in the following manner:

The first event in the buffer is examined. If the event is "no-process," or if the event is "may process" and the buffer is set for "hurry-up" distribution, the event is skipped; it is not sent to the analyzer and the next event in the buffer is examined immediately. If the event is "must process," or if the event is "may process" and the buffer is set for "normal" distribution, the event data are moved into the analyzer's single-event common block and the associated event processor in the analyzer is queued for execution. Control then passes to the analyzer so that the queued event can be processed and the single-event common block freed. When an event processor finishes its job, control is returned momentarily to the event distributor so that another event of the same or a different type can be transferred into the appropriate single-event common block.

While the PDP-11 is distributing and processing events, the MBD is filling available buffers with new event data; it stops acquisition only if no buffer space is available. A buffer in the PDP-11 becomes available for reuse by the MBD when it has been written to tape and all events contained in the buffer have been distributed appropriately. If event processing is slower than taping, the three buffers might look as follows:

Buffer 1 has already been written on tape and events are being distributed from it in "hurry-up" mode. The data-acquisition handler set buffer 1 for "hurry-up" distribution when the MBD informed it that buffer 2 was full. This distribution mode frees buffer 1 as quickly as possible for reuse by the MBD. Buffer 2 is being written to tape and is awaiting "normal" event distribution. As soon as the MBD fills buffer 3, buffer 2 will also be set for "hurry-up" distribution. Because the three buffers are circularly linked, the MBD will attempt to reuse buffer 1 when it finishes filling buffer 3. If event distribution is still active in buffer 1, the MBD must stop acquiring data until the PDP-11 alerts it that buffer 1 is free. There are several other possible buffer configurations, depending on the relative data-acquisition rates, tape recording speeds, and event processing rates.

Note that by the time an analyzer receives an event for processing it may be a very old event indeed. Therefore, there can be synchronized operations in response to an event only by the MBD, not by the PDP-11.

7.0 DOCUMENTATION GUIDE

The following paragraphs provide summaries of the information contained in each document associated with the Q data-acquisition system.* This manual provides an overview of the Q system and the user should be familiar with it before he attempts to use any of the others. The first four documents are listed in the reading order recommended for new users. The first two documents are aimed at new users of the RSX-11D operating system and the CAMAC system; they can be skipped by those who are already familiar

*Because Q is an ongoing project, some documents are revised periodically. Until the revised manuals are published, release notes are issued as internal reports.

with such systems. The remaining documents can be read in any order suitable to the user's interests.

7.1 An RSX-11D Primer

This document⁵ provides a simplified introduction to the RSX-11D operating system. It is designed to show the new user how to get his first FORTRAN program running under RSX-11D. The document assumes that the user has previous programming experience in FORTRAN. Useful DEC manuals are:

"RSX-11D User's Guide" (DEC-OXDUA-B-D),
"RSX-11 Utilities Procedures Manual" (DEC-11-OXMDA-A-D),
"RSX-11D Task Builder Reference Manual" (DEC-11-OXDLA-D-D),
"PDP-11 FORTRAN Language Reference Manual" (DEC-11-LFLRA-C-D),
"RSX-11D Special Subroutines Reference Manual" (DEC-11-LFMSA-B-D),
"FORTRAN IV-PLUS User's Guide" (DEC-11-LFPUA-B-D).

7.2 CAMAC Systems: A Pedestrian's Guide

This document⁶ presents a simplified introduction to the CAMAC interface standard and its use. Also see Ref. 2.

7.3 QAL and the Analyzer Task

This manual⁷ instructs the user in the writing of a Q analyzer and its translation with the QAL compiler. We suggest that the user be familiar with this manual, because an understanding of the information presented is a prerequisite to the effective use of the system.

7.4 QMODULE: CAMAC Modules Recognized by the QAL Compiler

This manual¹¹ catalogs the CAMAC modules that are recognized by the QAL compiler and what operations can be performed with them. This manual is a companion to the manual "QAL and the Analyzer Task."

7.5 Q Handbook

This manual⁸ provides operational details for the use of the Real Time Acquisition Subsystem, the Replay Subsystem, and the Utility Programs associated with Q. It also discusses the use of the trigger module.

7.6 A System for Histogram Entry, Retrieval, and Plotting

This manual⁹ describes the histogramming system that has been

developed for RSX-11. It can be used with Q or by any other task whenever the user needs to create histograms.

7.7 MDLIB: System Subroutine Library

This document¹⁰ briefly describes each subroutine module in the library MDLIB. Subroutines for histogramming, triggering an event from an analyzer, and plotting on the Tektronix 4000 series graphics terminals, as well as other generally useful routines, are included.

7.8 MBD Operations with MBDSTD and the RSX-11 CC Handler

This document¹² describes the general CAMAC branch operations available through RSX-11 and the MBD. FORTRAN-callable CAMAC subroutines are provided in the system subroutine library MDLIB.

7.9 An MBD Primer

This report¹³ is an introduction to the Microprogrammed Branch Driver (MBD) and its use with the CAMAC branch highway system. Most Q users do not need to program the MBD. However, if a user needs to write special experiment-dependent MBD code, this report provides valuable guidance.

7.10 Programming the MBD Model II

This report¹⁴ is the reference for MBD programming. It provides a complete description of MBD operations, and may be used alone or with "An MBD Primer"¹³ by a user who needs to program the MBD.

GLOSSARY

ANALYZER: An experiment-dependent module that specifies the data to be acquired from the CAMAC system (event specifications) and the processing to be performed on it (event processors).

CONTROL EVENTS: Events 24-31 are reserved for internal use by the Q system and are referred to as "control events."

DATA NAME or DEVICE NAME: A user-defined symbol that identifies event data words and the CAMAC registers from which they are read.

DATA WORDS: Words that contain data for a given event. Each event trigger used will have a named common (DWD0, DWD1, ..., DWD31) associated with it that contains the data words for the event. The individual words are named by the data names.

EVENT DISTRIBUTION: The process by which the data words comprising a single instance of an event are passed to the analyzer program.

EVENT NUMBER: Up to 32 different event triggers are available. They are numbered from 0 to 31 and are ordered by priority in the trigger module. The highest priority event is event 31. Events 24-31 are reserved for system functions and are not available to the user.

EVENT PROCESSOR: A procedure to be executed by the PDP-11 in response to a given event trigger. An event processor can take the form of a FORTRAN subroutine that accesses the event data in a named COMMON block. It functions much like an interrupt servicer, and entry and exit are provided in a special predefined manner.

EVENT SPECIFICATION: A procedure to be executed by the MED in response to an event trigger. Normally, the result of executing the procedure is the reading of event data from registers within CAMAC modules connected to an experiment.

EVENT TRIGGER: A signal to the trigger module that causes the data-acquisition system to take some action. Event triggers may be associated with physical events within the experiment or may be produced by pressing buttons on the trigger module front panel, by commands from the computer to the trigger module, or by the assertion of Look-at-Me (LAM) signals within the CAMAC system.

HURRY-UP EVENT DISTRIBUTION: A mode of event distribution in which only "must process" events in a buffer are distributed to the analyzer.

MAY PROCESS EVENT: An event type for which the experimenter wishes to process events only if data-acquisition speed is not

impaired.

MODULE NAME: Every CAMAC module in the data-acquisition system must be a member of a named class of modules. The module class normally corresponds to a single make and model. Associated with the module name is a table defining the legal functions that can be addressed to the given type of module. Normally the module name will be a two-letter manufacturer's code followed by a four-digit model number, for example, LR2248 (Lecroy octal ADC) or KS3610 (Kinetic Systems hex scaler). See "QMODULE: CAMAC Modules Recognized by the QAL Compiler"¹¹ for currently defined modules.

MULTIPLE EVENT PER BUFFER: A mode in which the MBD fills an entire buffer in the PDP-11 (which will contain several events) before interrupting the PDP-11.

MUST PROCESS EVENT: An event type for which the experimenter wishes to process every event.

NO PROCESS EVENT: An event type for which the experimenter wishes to perform no processing.

NORMAL EVENT DISTRIBUTION: A mode of event distribution in which both "may process" and "must process" events in a buffer are distributed to the analyzer.

SINGLE EVENT COMMON (BLOCK): A FORTRAN named common, whose name is based on the event type (DWD0, DWD1, ..., DWD31), large enough to hold all the data words that may be produced for a single event by the associated event specification. An event processor obtains event data words from a single event common block.

SINGLE EVENT PER BUFFER: A mode in which the MBD interrupts the PDP-11 every time an event is acquired and transferred to the PDP-11.

TAG WORD: An event header word consisting of two bytes. The high byte contains the event number and the low byte contains the number of data words to follow for the given event.

ACKNOWLEDGMENTS

We are indebted to Arvid Lundy for the design and implementation of the prototype trigger module and to Don Nickell for the production model. Contributions to the design of Q made by Jim Amann and Paul Elkins are gratefully acknowledged. A special note of appreciation is due to Richard Kittell and Randy Jeppeson, who designed and implemented the first versions of the histogram plot and dot plot systems. We owe a great deal to Janis Builta, Elvira Martinez, and Lorrie Voorhees for their dedicated effort in the preparation of this manual. We also appreciate the perceptive responses that a number of our colleagues and users made on an earlier draft of this manual. Finally, we thank the many experimenters who suggested improvements and extensions to Q and who helped us find and fix problems.

REFERENCES

1. H. L. Anderson, P. R. Bevington, J. A. Buchanan, H. S. Butler, R. E. Chrien, S. A. Kirsten, J. N. Snyder, R. F. Thomas, Jr., "LAMPF Data Acquisition Systems," Los Alamos Scientific Laboratory report LA-4504-MS (July 27, August 7, 1970).
2. "CAMAC: A Modular Instrumentation System for Data Handling," EURATOM 4100e (July 1972).
3. Lavon R. Biswell and Robert E. Rajala, "Design and Operation of a Microprogrammed Branch Driver for a PDP-11 Computer," Los Alamos Scientific Laboratory report LA-5144 (May 1973).
4. Jeffrey M. Gallup, Edward G. Lieberman, and Sally Shlaer, "RSX-11M versus RSX-11D: Compatibility and Conflict," Los Alamos Scientific Laboratory report LA-6789-MS (May 1977).
5. "An RSX-11D Primer," Los Alamos Scientific Laboratory Group MP-1 internal report (1977).
6. B. Zacharov, "CAMAC Systems: A Pedestrian's Guide," Daresbury Nuclear Physics Laboratory, Warrington WA4-4AD, England, (1972).
7. Martin Kellogg, Michael M. Minor, Sally Shlaer, Nancy Spencer, Richard F. Thomas, Jr., and Henri van der Beken, "QAL and the Analyzer Task," Los Alamos Scientific Laboratory report LA-6884-M (October 1977).
8. Martin Kellogg, Michael M. Minor, Sally Shlaer, Nancy Spencer, Richard F. Thomas, Jr., and Henri van der Beken, "Q Handbook," Los Alamos Scientific Laboratory report LA-7002-M to be published.
9. Martin Kellogg, Jeffery M. Gallup, Sally Shlaer, and Nancy Spencer, "A System for Histogram Entry, Retrieval, and Plotting," Los Alamos Scientific Laboratory report LA-6920-M (October 1977).
10. "MDLIB: System Subroutine Library," Los Alamos Scientific Laboratory Group MP-1 internal report (1977).
11. Martin Kellogg, Michael M. Minor, Sally Shlaer, Nancy Spencer, Richard F. Thomas, Jr., and Henri van der Beken, "QMODULE: CAMAC Modules Recognized by the QAL Compiler," Los Alamos Scientific Laboratory report LA-6951-M (October 1977).
12. "MBD Operations with MBDSTD and the RSX-11 CC Handler," Los

Alamos Scientific Laboratory Group MP-1 internal report (1977).

13. Sally Shlaer, "An MBD Primer," Los Alamos Scientific Laboratory report LA-5511-MS (August 1974).
14. Richard F. Thomas, Jr., "Programming the MBD Model II," Los Alamos Scientific Laboratory report LA-5404-MS (September 1973).

INDEX

Analyzer	8
CAMAC	3
Control events	21
Data buffering	13
Data name	21
Data words	21
Device name	21
Event distribution	13, 21
Event header word	16, 22
Event number	11, 21
Event processor	21
Event specification	21
Event trigger	21
Graded LAM	11
Histogram display system	9
Histogram management system	9
Hurry-up event distribution	17, 18, 21
May process	16, 21
MBD	3
MDLIB	10
Module name	22
Multiple event per buffer mode	15, 22
Must process	16, 22
No process	16, 22
Normal event distribution	17, 22
QAL	8
RSX-11D	4
RSX-11M	5
Single event common block	17, 22
Single event per buffer mode	15, 22
Tag word	16, 22
Trigger module	10