SusiQ

A Data Acquisition system & Event Analyzer for PC based systems

Reference Manual

Soft. v7.2 Feb 96
Doc. v3.0 Nov96

Authors
P.-A. Amaudruz : Data Acquisition
R. Tacik : Graphic package

Table of contents

IndexTable

SusiQ System overview

Unlike most commercial acquisition packages, SusiQ is a ***multi data word*** data acquisition and analyzer package. It uses high level language for acquisition definition and event analysis. This package is based on a PDP /VAX version ***SUSI*** used at a nuclear physics research centre. This package is a stand alone system, which can be used off the shelf or full blown with user specific FORTRAN routines. The off-shelf package contains an executable *susiq.exe* which can take care of most of the simple detector test requirements. Setup, acquisition, event analyzer and histogram file definitions are the only necessary files to make SusiQ operational. SusiQ supports two type of CAMAC hardware interfaces: The KCS 3922 and the DSP 6002 CAMAC crate controllers.

So, **"What can SusiQ do for you ?"...**

- ♥ Reads your CAMAC event record on: LAM request, TIME basis, or RANDOM TIME basis.
- ♥ Processes each event record through an event-by-event analyzer, where conditional tests based on the content of the record or computed quantities will generate a test array result.
- ♥ Keeps track of the statistics of these test results for conditional histogram generation.
- ♥ Saves either raw data or computed data on a storage device (disk file) for later replay.
- ♥ Saves test results and histogram results on run by run basis.

SusiQ can also display **online** :

- Raw data (CAMAC records) on event-by-event or user specified rate.
- Computed quantities, test results on event-by-event or user specified rate.
- Test statistical results based on a user conditions.
- Histograms of a single variable of the raw or computed array gated on user defined conditions.
- Dotplot (scatter dotplot) of two variables on user specified condition.

Introduction

The SusiQ Data acquisition and Analyser package can be used in two different ways.

- Basic SusiQ (built-in executable)
- User Customized SusiQ (User-built package)

The first package, *Basic SusiQ,* refers to the *off-the-shelf* package. It does not of course provide all the features that SusiQ is capable of, but the basic data acquisition procedures are provided. The second package, *User Customized SusiQ,* refers to the full blown package with which users may tailor SusiQ to their own particular application. This package also requires the MicroSoft FORTRAN 5.x compiler to generate the necessary object and image files. The *pro's* and *con's* of the two implementations are presented in the following table.

Features	Basic Susiq	User Customized SusiQ
KCS or DSP crate controller selection	yes	has to be built
FORTRAN compiler required	no	yes
Simple or Complicated Data acquisition definition	yes	yes
Full commands of the test package	yes except all user callable routines	yes
Full options of the histogram package	yes	yes
Full Dotplot capability	yes	yes
"Data taping" to file	yes	yes
Data replay from file	yes	yes
Time based histogram	yes	yes

- The *Basic SusiQ* package resides in the *\basic* directory. When histograms of a few CAMAC words are the only requirements, this package is well suited to address this problem. Basic data gates, Boolean operations, conditional branching and data calibration can be performed without any effort. This package requires that you setup a dedicated directory where you will copy the contents of the *\basic* directory, and then modify the definitions to match your requirements.
- The *User Customized SusiQ* package will be used mainly when data treatments from the test package alone are no longer sufficent; when data manipulation has to be performed on an event-by-event basis, or when extraction of skimmed raw data or pre-analyzed data is necessary. Under any of these conditions, users are required to generate their own FORTRAN code to perform the corresponding actions and by using the provided SusiQ compilation and Linking tools, a User Cutomized SusiQ can be generated.

Distribution listing

SusiQ requires a directory tree of three branches under the *\SQ* directory. The main directory contains all the necessary system files and tools to get SusiQ running. The *\basic* directory contains the *Basic SusiQ* set of files required to run without the need of user specific routines. The *\srce* directory contains all the necessary files to build a users specific SusiQ. (This does not include the FORTRAN compiler). The *\test* directory contains a minimum set of application files for a user-specific SusiQ. The SusiQ image file is not included. The user must generate it through the *Compilation & Linking* procedure (See User Customized SusiQ).

```
|___SQ          ⇦ SusiQ init/compilation tools
    |__BASIC    ⇦ Basic configuration (without FORTRAN)
    |__SRCE     ⇦ SusiQ sources and objects code
    |__TEST     ⇦ Full SusiQ setup
```

SusiQ distribution listing

Main Directory	Commands SUSIQ
\SQ\UPDATE.TXT	List of modifications applied to SusiQ
\SQ\MKLINK.EXE	Linker builder
\SQ\CHKSQPAT.EXE	DOS path checker
\SQ\LOADSQ.BAT	SusiQ environment loader
\SQ\CAMTEST.EXE	CAMAC test
\SQ\FCNAxxx.EXE	CAMAC tool for KCS or DSP
\SQ\SUSCOF.EXE	Constant equivalence builder
\SQ\SUSEQU.EXE	Computed quantities equivalence builder
\SQ\SUSEVT.EXE	Raw event equivalent builder
\SQ\DDRA.EXE	Graphic package (DRA)
\SQ\HM.EXE	Histogram manager package
\SQ\SH.EXE	SusiQ help
\SQ\SUSIQ.HLP	SusiQ help file
\SQ\SQLINK.xxx	SusiQ linker tools
\SQ\UNLOADSQ.BAT	SusiQ unloader (remove SusiQ path)
\SQ\QLINK.BAT	User SusiQ linker
\SQ\QLIBO.BAT	Internal library manager for compilation
\SQ\QFORO.BAT	Internal compiler manager
\SQ\QFOR.BAT	Internal compiler manager
\SQ\QFALL.BAT	User SusiQ compiler tool
\SQ\MODERN.FON	Default font for graphics
\SQ\INSER.BAT	Internal compiler tool
\SQ\FORT.BAT	Internal compiler tool
\SQ\FORD.BAT	Internal compiler tool for debugging
\SQ\PC004DRV.COM	DSP Camac driver
\SQ\CAMACDRV.SYS	KCS Camac driver
\SQ\SQ{B}HARD.BAT	SusiQ Hardware selection tool
\SQ\H6127.EXE	Histogram conversion tools v6.1 to v7.x
Sub Directory	**Basic package**
\SQ\BASIC\SUSIQ.ZOO	Zoom definition
\SQ\BASIC\SUSIQ.DOT	Dotplot definition
\SQ\BASIC\SUSIQ.TST	Test definition
\SQ\BASIC\SUSIQ.HST	Histogram definition
\SQ\BASIC\SUSIQ.EXE	SusiQ Image file
\SQ\BASIC\ADCx01.BOX	Box definition
\SQ\BASIC\SUSIQ.CST	Constant definition
\SQ\BASIC\SUSIQ.DFT	SusiQ default settings
\SQ\BASIC\SUSIQ.EQU	Computed quantity definition
\SQ\BASIC\SUSIQ.ACQ	SusiQ acquisition definition
\SQ\BASIC\SOUND.2	BOR sound definition
\SQ\BASIC\SOUND.1	EOR sound definition
\SQ\SRCE\SUSIQ.LIB	Main SusiQ objects library

Sub Directory	Source SusiQ
\SQ\SRCE\SQHEADER.INC	SusiQ Hardware definition header
\SQ\SRCE\SQCAMAC.FOR	SusiQ CAMAC dependent routines
\SQ\SRCE*.OBJ	All SusiQ object files
\SQ\SRCE\PAASYS.LIB	Private objects library
\SQ\SRCE\SUSITST.INC	Test include file
\SQ\SRCE\SUSIHIS.INC	Histogram include file
\SQ\SRCE\SUSIGRF.INC	Graphics include file
\SQ\SRCE\SUSIEQU.INC	Computer quantity include file
\SQ\SRCE\SUSICTL.INC	Main SusiQ control include file
\SQ\SRCE\SUSIACQ.INC	Acquisition include file
\SQ\SRCE\CAUSER.INC	Kinetics include file
\SQ\SRCE\CAMDSP.LIB	DSP objects library
\SQ\SRCE\CAMKCS.LIB	Kinetics objects library
\SQ\SRCE\CAMACL.LIB	Main SusiQ CAMAC library
\SQ\SRCE\SUSIQ.LIB	Main SusiQ library
Sub Directory	**User Customized SUSIQ**
\SQ\TEST\SUSIQ.ZOO	Zoom definition
\SQ\TEST\SUSIQ.DOT	Dotplot definition
\SQ\TEST\SUSIQ.HST	Histogram definition
\SQ\TEST\SUSIQ.CST	Constants definition
\SQ\TEST\SUSIQ.DFT	SusiQ setup definition
\SQ\TEST\QUSER.LIB	SusiQ User objects library
\SQ\TEST\SUSIQ.TST	Test definition
\SQ\TEST\SUSIEQU.INC	User computed quantities equiv. file
\SQ\TEST\SUSIEVT.INC	User raw data equivalence file
\SQ\TEST\SUSICOF.INC	User constants equivalence file
\SQ\TEST\OUTUSER.FOR	User skim routine
\SQ\TEST\USER5.FOR	User routine
\SQ\TEST\USER4.FOR	User routine
\SQ\TEST\USER3.FOR	User routine
\SQ\TEST\USER2.FOR	User routine
\SQ\TEST\USER1.FOR	User routine
\SQ\TEST\UEOR.FOR	User End_Of_Run routine
\SQ\TEST\UBOR.FOR	User Begin_Of_Run routine
\SQ\TEST\SUSIQ.EQU	Computed quantities definition
\SQ\TEST\SUSIQ.ACQ	Acquisition definition
\SQ\TEST\SUSGEN.FOR	Random event generator routine
\SQ\TEST\SOUND.2	EOR sound definition
\SQ\TEST\SOUND.1	BOR sound definition
\SQ\TEST\PROT.FOR	User routine
\SQ\TEST\PION.FOR	User routine

Some things to be aware of

SusiQ Limitations

The limitations consist essentially of the predefined size of the declared arrays in the system. This version does not use dynamic memory allocation. The different sizes of the arrays are determined via PARAMETER declarations in the corresponding include files. The most common limitation seen by the user is the displayed number of test results on the screen. Even if the arrays are internally large enough to handle the analysis properly, the online display will put a limit on the number of variables seen on the screen. The three main display tables have the following limits:

- EFFICIENCIES 38 test variables
- PREV 38 raw data words
- PEEPER 72 variables (raw+computed var.+tests)

The rest of the system limitations are given in the following table. The declaration of these parameters are made within one of the include files. Essentially, one include file is provided for each section of SusiQ, i.e. acquisition, test and histogram section. Three more include files are available which deal with general inter-routine communication (*susictl.inc)*, graphic support (*susigrf.inc)* and camac access (*causer.inc).*

System limitation table

Parameter	Limit	Include files	Description
bufsize	200	*susiACQ*	CAMAC address
ncammax	99	*susiACQ*	CAMAC function
maxhis	50	*susiHIS*	Max #_of histograms definitions
maxtotbin	12000	*susiHIS*	Overall max #_of bins
Gmaxhstbin	2048	*susiHIS*	Max #_of bins per histo
maxdot	15	*susiHIS*	Max #_of dotplot definitions
maxxydot	100	*susiHIS*	Max #_of bins for projection dotplots
tstsize	512	*susiTST*	Test control (#_of commands)
parsize	512	*susiTST*	Symbol table (raw+computed+test)
arrsize	128	*susiTST*	Efficiency array (test)

SusiQ display

The screen is divided into three sections or windows. The top part of the screen is referred to as the *Status window*, the middle of the screen as the *Display window* and the bottom part as the *Prompt window.*

- *Status window* : This window contains information which will remain throughout the SusiQ session. [File name] is displayed if running in the SUSTAP, CAMTAP or REPLAY mode or in the corresponding TIMGEN and RANGEN modes.

SusiQ version, Run Status, Acquisition mode, Current run number, status Flags, Time & Date Experiment number, Experiment name, Current event number, [File name]

- *Display window* : The centre part of the screen is reserved for the display of the data and for system messages. It allocates most of the screen area. So far no user routine is provided for accessing this section of the screen.

- *Prompt window* : The bottom part of the screen is reserved for input/output of commands to/from the system. This space (last 2 lines of the screen) can be accessed by the user through a couple of system routines which are described in the *SusiQ Internal Tools* section. (Iprompt, Oprompt)

Installation considerations

The installation of SusiQ requires some attention. SusiQ can be ported to any PC's from a 8086 to 80486, but the performance will be scaled in the same manner. In addition to the software aspect, the hardware required for the acquisition may set the performance limitation on the installation. On the other hand, SusiQ has been installed on 286 and higher models from 8 to 33 Mhz. If a slow PC is a problem, too fast a PC may have also some drawbacks. The hardware here again being the limitation, due to the fact that the I/O board may be designed for 8 bit data transfer thus limiting the data thruput. As *SusiQ* can take user FORTRAN objects, it expects to have MicroSoft FORTRAN 5.x present and installed on the hard disk. The compilation and linking commands which are defined in the different batch files of *SusiQ* use the **FL**, **LINK**, **LIB** commands from the MicroSoft Compiler. The GRAPHICS.LIB library should also be available through the LIB symbol definition. No special handling is provided by SusiQ for screen dumps. It is up to the user to look into the different commercial products for screen grabbers. DOS provides in its standard package, a TSR call *graphics.com*. which supports a large variety of printers. Other fancy TSR like *pizza* can also be installed prior running SusiQ. Regarding CAMAC access, SusiQ supports the KCS and DSP interfaces. DSP requires a special driver *pc004drv.com for proper operations*. Unfortunately, it does not support the ORed LAM masking bit. This means that a LAM check on the expected module has to be performed within *chklam.for*. A special batch command file (*sqbhard.bat*, *sqhard.bat*) allows you to switch from KCS to DSP and via versa for both SusiQ packages (*Basic and full blown*). The actual routines accessing CAMAC are located in the *\sq\srce* directory. ONLY those routines have to be modified if CAMAC access has to be changed. In these routines the variables used for CAMAC access are primarily the following:

CAMAC access routines

opcamsq.for	Initialize CAMAC access open CAMAC channel (KCS requires a handle on the CAMAC driver)
clcamsq.for	Close CAMAC channel
chklam.for	Check LAM routine (LAM handling routine)
camsq.for	CAMAC function routine (single data collection)

CAMAC variables

chan	CAMAC channel
lammask	Station LAM mask (from **LAM_SOURCE** in *file.ACQ*)
ic	Crate number
is	Slot number
ia	Subaddress number
if	CAMAC function
id	Input/Output data
errstat	error parameter

Basic SusiQ installation

Basic configuration refers to the installation of SusiQ without the need of user routines. The files contained in the *\basic* directory are sufficient to start an acquisition system based on the KCS or DSP CAMAC system.

Basic SusiQ user files

SUSIQ.DFT	Basic SusiQ default file (Set for SUSGEN acquisition)
SUSIQ.ACQ	Basic acquisition, just to give you an idea
SUSIQ.CST	Basic constant definition file. With *numout* definition for *outuser* (see *internal tools*)
SUSIQ.EQU	Basic computed quantities definition file (must be present, BUT NOT USED)
SUSIQ.HST	Basic histo definition file with the 3 types of histograms
SUSIQ.TST	Basic test package, with call to user1 for *outuser* example (see *internal tools*)
SUSIQ.DOT	Will be filled automatically when you define a DOTPLOT. Does not have to be present.
SUSIQ.EXE	Image file. In fact ANY SUSIQ.EXE can be used if no user routine is requested in *file.TST*
SOUND.1	End of Run alarm file
SOUND.2	Begining of Run alarm file

The installation of *Basic SusiQ* should follow this guide line:

- Add <*dev*>*:\sq* to the DOS path by editing *autoexec.bat*. Where <dev> can be any valid drive (in this example c:\). For the path, DOS limits the length to 128 characters (beware!).

PATH = ...c:\dos;c:\...;...;c:\sq

- Install the CAMAC driver and reboot the computer.

For KCS the following line has to be entered in config.sys
device = c:\sq\camacdrv.sys or devicehigh = c:\sq\camacdrv.sys

For DSP the following line has to be entered in the autoexec.bat
c:\sq\pc004drv.com

- Run the *loadsq <dev>* where <dev> is the disk on which SusiQ resides (C:, D: ...) *loadsq* sets several SusiQ environment variables. It is not essential for the *basic susiq* but it will make the internal help work propery. It is possible that more space is required to load these parameters. Refers to DOS manual for increasing the environment space.

c:\sq\loadsq c

- Select the CAMAC interface (KCS or DSP) by running SQBHARD <interface_name>. This batch command file will copy the pre-linked hardware-dependent image file to SusiQ.exe. In either CAMAC interface, the necessary driver is available in the sq\ directory under the name *camacdrv.sys* for KCS or *pc004drv.com* for DSP. It has must be installed prior to running SusiQ.

c:\sq\sqbhard dsp
or
c:\sq\sqbhard kcs

- Setup a new experiment directory and copy the necessary files from the \sq\basic\ directory.

mkdir c:\expt45
copy c:\sq\basic*.* c:\expt45

- Modify the configuration file *setup.DFT* and the corresponding files to reflect your setup. It is IMPORTANT to make sure that the file name declarations in the setup file match the actual definition files.

edit susiq.dft
... susiq.acq ⇨ expt45.acq
... susiq.tst ⇨ expt45.tst

- Invoke susiq from the experiment directory. SusiQ should at this point be running.

c:\expt45> susiq

- After closing SusiQ, the environment space can be released and the previous path reinstalled by executing the *\sq\unloadsq.bat* command.

c:\expt45> unloadsq

User Customized SusiQ

The installation of *full blown susiq* requires that MicroSoft FORTRAN 5.x first be installed. The *full blown susiq* package is completely independent of *Basic SusiQ*. This way both packages can be implemented on the same computer but in two different directories. The major difference between these two packages is the declaration of environment variables for the FORTRAN compiler. The setup is done via a batch command file *loadsq.bat*, which must be run before invoking SusiQ. In all other respects, the hardware installation of *user costumized SusiQ* follows the same steps as *basicSusiQ*.

- *basic susiq* installation, check *loadsq.bat* for the FORTRAN environment path.
- Build the actual SusiQ image file. To do so refer to *Compilation & Linking procedures*.

Basic Operation of SusiQ

Much user frustration can be eliminated when running a brand new software package by having some basic understanding of the inside of the software. Is is always nice to know what the program is suppose to do at any given time to anticipate its reaction. SusiQ performs two major activities:

1. Check for event to be read.
2. Check for keyboard activity.

After these two tests, SusiQ will automatically do its internal house managment which results in the update of the screen to reflect any new run status. This check loop ought NEVER to be broken to ensure SusiQ a proper servicing to both parties (acquisition, user).

- SusiQ first reads the setup file *file.DFT*, loads and checks all the definition files, then presents the SusiQ page on the screen. If failure occurs during loading messages, will be displayed on the screen to help you trace the problem.
- At this point SusiQ is waiting on the keyboard for user inputs. The system will be in a *standby* mode looking at the keyboard for user interaction and refreshing the screen depending on the refresh rate parameter ([s] status)

- A normal SusiQ operation will request a begining of run. SusiQ will then verify if all the definition files were loaded properly and only if so will proceed to the next stage of starting the initialisation of the run. (BOR phase, [c] begin)
- When the run is started, SusiQ goes into polling mode for LAM or user intervention. Because SusiQ does not run in a multitasking environment, it will be either serving a LAM request (acquisition of an event) or a user request (user command).
- When a LAM occurs, the CAMAC acquisition sequence will take place, followed by the analysis and histogram incrementation procedure for that event. Only then will SusiQ return to the user/refresh job.
- When the run is ended (automatically or on request [c] end), SusiQ will take the appropriate action for closing the run and resume the user/refresh job.

Compilation & Linking procedures

Compilation has been facilitated by the use of two batch command files *qfall*, *qfor*. These batch files automatically update the necessary libraries involved in the linking phase. *Qfall* is the recommended procedure to be used to ensure complete consistency within the SusiQ image. The linking *Qlink* is performed with the help of dedicated batch and FORTRAN programs to make this operation more environment independent.

☞ All these command files require that you issue them from your working directory and that *loadsq <dev>* has been performed.

Compilation & linking tools table

QFALL	Generates the necessary include files from *file.ACQ*, *file.EQU*, *file.CST*, Compiles all the FORTRAN sources in the current directory and inserts them into the user object library *QUSER.LIB* It is the standard command for generating a brand new version of SusiQ.
QLINK	Link SusiQ using the *QUSER.LIB* and the SusiQ system libraries. Creates a file *sqlink.tmp* which is used as the batch file for the real LINK command.
QFOR module	Compile and insert the given module into *QUSER.LIB* in the current working directory. Can replace *qfall* and save time if the user knows what he is doing. But it is generally preferable to run *qfall* in any case to make sure that the new SusiQ image file will be consistent.
SQHARD <interface_name>	Compile the KCS or DSP routines and insert them in the susiq library. Must be done during the installation of SusiQ for the full SusiQ package. (See *User Customized SusiQ* section)

Examples **Compilation &Linking**

```
c:\expt1> QFALL
SUSIQ user routine compiler facility
%susevt-I- Getting SUSIQ.ACQ
       Integer*4  ADC0
       Integer*4  ADC1
       Integer*4  ADC2
       Integer*4  TDC0
       Integer*4  TDC1
      equivalence (Ibuff(    1) ,ADC0)
      equivalence (Ibuff(    2) ,ADC1)
      equivalence (Ibuff(    3) ,ADC2)
      equivalence (Ibuff(    4) ,TDC0)
      equivalence (Ibuff(    5) ,TDC1)
 SUSEVT - PAA - Done

%suscof-I- Getting SUSIQ.CST
       Real*4 NUMOUT
      equivalence (const(        1) ,NUMOUT)
-SUSCOF -PAA-  Done

%susequ-I- Getting SUSIQ.EQU
...
 SUSEQU - PAA - Done

fl /Ox /G2 /c /If:\sqd\srce OUTUSER.FOR...
Microsoft (R) FORTRAN Optimizing Compiler Version 5.10
USER1
fl /Ox /G2 /c /If:\sqd\srce USER1.FOR...
Microsoft (R) FORTRAN Optimizing Compiler Version 5.10
Copyright (c) Microsoft Corp 1982-1991. All rights reserved.

PION.FOR
fl /Ox /G2 /c /If:\sqd\srce PION.FOR...
Microsoft (R) FORTRAN Optimizing Compiler Version 5.10
Copyright (c) Microsoft Corp 1982-1991. All rights reserved.

Inserting/Replacing USER1.OBJ in Quser...
Inserting/Replacing PION.OBJ in Quser...

Microsoft (R) Library Manager  Version 3.18
Copyright (C) Microsoft Corp 1983-1991.  All rights reserved.
OUTUSER.FOR  PION.FOR  PROT.FOR  SUSGEN.FOR  UBOR.FOR
UEOR.FOR  USER1.FOR  USER2.FOR  USER3.FOR  USER4.FOR  USER5.FOR
11 file(s)     7578 bytes
-----------------------------------------
c:\expt1> QLINK
-----------------------------------------
c:\expt1> QFOR user1
fl /G2 /c /If:\sqd\srce user1.for...
Microsoft (R) FORTRAN Optimizing Compiler Version 5.10
```

Definition Files

SusiQ takes all its information from several definition files. All of them are ordinary ASCII files, which makes them easy to modify. All the definition files have a common organisation. They are each divided into two sections delimited by an "=" (equal sign) in the first column. Any text prior to the equal sign is regarded as a comment (with the exception of the constant file *file.CST* see this section) . After the equal sign, SusiQ expects to see a known command and will attempt to interpret the string as a valid instruction. The "!" (exclamation mark) can be used anywhere in the second section to comment out any string placed on it's right hand side.

Setup file *Susiq.dft*

This file sets up SusiQ for the upcoming acquisition. The file name is unique and **must be named *susiq.dft*** . A good practice is to have a dedicated directory with a dedicated *susiq.DFT* defining the experimental setup. The following table describes all the possible setups definitions which may be included in *susiq.DFT*. The first 7 entries are the minimum requirement. The other definitions will default to the conditions shown if they are not explicitly entered in *susiq.DFT*.

Setup command Table

Command syntax	default	Description
ACQFILE *file.ACQ*	required	Acquisition definition file name
TSTFILE *file.TST*	required	Test definition file name
HSTFILE *file.HST*	required	Histogram definition file name
EQUFILE *file.EQU*	required	Equivalence definition file name
CSTFILE *file.CST*	required	Constant definition file name
DOTFILE *file.DOT*	required	Dotplot definition file name
ZOOFILE *file .ZOO*	required	Zoom definition file name
PEDIT *file.ext*	*q.exe*	User Editor
PEDITOR *file.ext*	*susiq.tst*	Default file for editor
EXPERIMENT *expt_name*	*JOB*	For save_set file concatenation (**4 char max**)
TSTNORM *test_name*	PASSALL	Normalization test for the efficiencies
FONT *path\file.FON*	*modern.fon*	Font file for Histogram presentation
ACQ_MODE *mode_string*	*SUSGEN*	Default acquisition mode (see ACQ_MODE table)
EXPT_# *number*	1	Used for identification only
DEBUG	off	Generates a log file of the setup procedure. The file will be *SETUP.DBG*
BEEP	off	Turn on the Beep flag for BOR and EOR signal
CLRTST	off	Turns on system flag for automatic clearing of the test result before starting a run.
CLRHST	off	Turns on the system flag for automatic clearing of all histogram contents before starting a run
SAVTST	off	Turns on system flag for automatic test result save_set to a file after ending a run. The file will be named TSTrun_#.SQT
SAVHST	off	Turns on system flag for automatic histogram save_set to a file after ending a run. The file will be named HSTrun_#.SQH
DEBUG1	off	Turns on the user flag *DEBUG1*
DEBUG2	off	Turns on the user flag *DEBUG2*
VIDEO_MODE *mode*	-3	Set video mode. -3 : maxresmode, 16 : 640x350 4colors 17 : 640x350 BW Default -3. No special screen rearrangement is taken depending on the video mode.
HARDWARE *type*	0	Specify the CAMAC hardware used for the acquisition. 1 : Kinetics, 2 : DSP (not used yet)

UP_STAT *freq*	50	Default update frequency of the Status window
UP_EFFI *freq*	50	Default update frequency of the efficiency table
UP_HIST *freq*	50	Default update frequency of the histogram display
UP_PREV *freq*	50	Default update frequency of the raw data table
UP_PEEP *freq*	50	Default update frequency of the peeper data table
WAIT_LAM *number*	10	Default number of LAM checks before screen update
FGCOLOR *string of number*	see table	Foreground color definition. (see FGCOLOR table)
BGCOLOR *string of number*	10x0	Background color definition. default all set to 0
EVT_LIM *number*	0	Default event limit (unlimited)
HIS_LIM *number*	1000	Default event limit based on histogram
TST_LIM *number*	1000	Default event limit based on condition
RUN_# *number*	1	Used for results file concatenation

ACQ_MODE defines the mode of acquisition. The default is SUSGEN which generates random data for simulation. This parameter has to be set to CAMxxx to be able to access CAMAC interface.

Acquisition mode table

Syntax	Description
SUSGEN	Acquisition through the SUSGEN routine for a gaussian data spectrum centered at 1000 with a sigma of 500
SUSTAP[1]	Like SUSGEN with storage of the raw data to a file
CAMAC	Acquisition through CAMAC using the file.ACQ definition
CAMTAP[1]	Like CAMAC with storage of the raw data to a file
REPLAY	Analysis of pre-saved data into a file using the SUSTAP or CAMTAP acquisition mode
TIMGEN	Acquisition based on time generated LAM
TIMGEN SUSGEN/TAP[1]	Secondary acq_mode within TIMGEN
TIMGEN CAMAC/TAP time cr1 sl1 sub1 fun1 data1 cr2 sl2 sub2 fun2 data2	Secondary acq_mode within TIMGEN with 2 specific user CAMAC operation to generate LAM
TIMGEN CAMAC/TAP[1] time 0 sl1 sub1 fun1 data1 cr2 sl2 sub2 fun2 data2	Secondary acq_mode within TIMGEN with BYPASS of the LAM generation. (right hand side of the 0 is disregarded)
Glossary	
time	Interval time for user CAMAC operation
crx slx subx funx datax	Camac operand description, crate, slot, subaddress, function, data for the first and second CAMAC command

It is especially important to correctly set the forground colors for a monochrome monitor. The table consists of a string of 10 numbers which correspond sequentially to the different parts of the screen.

FGCOLOR Table

position	Default	Description
1	11 / light cyan	Prompt
2	12 / light red	Help
3	13 / light magenta	Listing
4	14 / yellow	Error
5	15 / bright white	Message
6	14 / yellow	Data
7	13 / light magenta	Border
8	12 / light red	Histogram
9	10 / light green	Status_1
10	9 / light blue	Status_2

COLOR assignment Table

Color _#	Color	Color_#	Color
0	black	8	dark gray
1	blue	9	light blue
2	green	10	light green
3	cyan	11	light cyan
4	red	12	light red
5	magenta	13	light magenta
6	brown	14	yellow
7	white	15	bright white

1 See Data file format for record structure

Example file susiq.DFT

```
---> SUSIQ setup definition <---
Valid commands : R:requested, S:suggested, O:optional
R ACQFILE     R TSTFILE     R CSTFILE     R EQUFILE
R HSTFILE     R ZOOFILE     R DOTFILE
S PEDIT       S EDITOR      S EXPERIMENT  S TSTNORM
S FONT        S ACQ_MODE    S EXPT_#
O DEBUG       O DOTFILE     O CALFILE     O BEEP
O CLRTST      O CLRHST      O SAVTST      O CLRHST
O DEBUG1      O DEBUG2      O video_mode  O HARDWARE
O up_stat     O up_effi     O up_hist     O up_prev
O up_peep     O wait_lam                  O fgcolor
O evt_lim     O his_lim     O tst_lim     O run_#

For video_mode the valid mode are :
-3 == $maxresmode   (default)
16 == $erescolor    ( 640x350 4 or 16 color)
17 == $vres2color   ( 640x480 BW )

For HARDWARE the valid mode are :
 1 == Kinetics CAMAC 3922-Z1A (default)
 2 == DSP      CAMAC PC004
For Color Assignment :
FGCOLOR <prompt> <help> <listing> <error> <message> ...
...    <data> <border> <histo> <status1> <status2>
For the acquisition mode :
ACQ_MODE
<SUSGEN/SUSTAP/CAMAC/CAMTAP/REPLAY/TIMGEN/RANGEN>...
if <TIMGEN/RANGEN> <SUSGEN/SUSTAP/CAMAC/CAMTAP>...
... <time interval[s]> <CNAFD(1)> [<CNAFD(2)>]
if  (C(1) = 0) then LAM is BYPASSED (for CAMAC/CAMTAP)
creation date  02:09:56.94   05-OCT-1991
=====================================================
TSTFILE    SUSIQ.TST
CSTFILE    SUSIQ.CST
ACQFILE    SUSIQ.ACQ
EQUFILE    SUSIQ.EQU
HSTFILE    SUSIQ.HST
DOTFILE    SUSIQ.DOT
ZOOFILE    SUSIQ.ZOO
PEDIT      SUSIQ.DFT
EDITOR     Q
TSTNORM    PASSALL
EXPERIMENT TEST
! FONT     f:\sqd\modern.fon        use default
CLRTST
CLRHST
ACQ_MODE   SUSGEN
! ACQ_MODE   TIMGEN  SUSGEN    1  0 12  1 16  33023  1 12  1 16  32768
VIDEO_MODE       -3
HARDWARE          1
EVT_LIM      100000
HIS_LIM          30
TST_LIM       10000
UP_EFFI          50
UP_HIST          50
UP_PREV          20
UP_PEEP          20
UP_STAT          50
WAIT_LAM         10
BKCOLOR      0   0   0   0   0   0   0   0   0   0
!0 black   ,1 Blue  ,2  Green  , 3 Cyan  ,4 Red   , 5 Mag  , 6 Brow, 7 White
!8 darkgrey,9 L_blue,10 L-green,11 L-Cyan,12 L_Red,13 L-Mag,14 Yel
!15 BrightWhite
! colors  Prompt Help listing error message data border histo status1 status2
FGCOLOR  11    13    2    12    14    15    14   3    9    10
! FGCOLOR     12 14 13 15 10 15 11  9 14 13
! FGCOLOR     15 15 15 15 15 15 15 15 15 15   for Black & White
EXPT_#      1
```

Acquisition *file.ACQ*

This file describes the way the acquisition will be performed on an event-by-event basis. This is the file in which you declare what action has to be performed BEFORE and AFTER each run and AFTER each event acquisition. The LAM source is specified in this file. The command part is subdivided into 5 sections referring to the 5 phases of the data acquisition, i.e :

① ***LAM source definition***

Lam_source defines which slot(s) within a given crate generate the LAM. A mask is given to specifiy the slot number. Each bit of the mask corresponds to a slot number starting with slot number 1.

```
LAM_SOURCE 1 #8           ⇨ Crate1 Slot4
LAM_SOURCE 2 #200         ⇨ Crate2, Slot10
```

② ***Begin of Run definition***

The BOR action defines what CAMAC operations must be performed prior to starting a run. These action[s] are done BEFORE the UBOR user routine (if used). The key word **begin_event** must define the end of the BOR section.

③ ***Event definition***

Event reading defines what to readout and which variable name to relate to each CAMAC word. This section usually expects only the CAMAC READ function. The key word **end_event** defines the closure of the event definition.

④ ***Rearming definition***

Reset procedure defines what has to be done after having acquired an event. This usually involves resetting the CAMAC modules for next event. F9-11 type functions are used. Resetting the LAM has to be guaranteed by the user. The system LAM within SusiQ will be reset automatically on the last command of this section via **rearm_lam.**

⑤ ***End of Run definition***

The EOR action defines what action has to be performed at the end of a run AFTER the UEOR user routine(if used). This will give the user an opportunity to do more CAMAC commands if required before the CAMAC channel is closed. This section usually contains the Enable of the crate Inhibit. The keyword **end_camac** closes this section AND the acquisition file.

Acquisition command table

Syntax	Section Validity	Description
LAM_SOURCE cr pattern	1	Define what slot pattern in which crate provides the LAM
CAM24 cr sl sub fun	2,3,4,5	Single function (ex : F9,F24,..)
CAM24 cr sl sub fun DATA data	2,3,4,5	Single function with data loading
CAM24 cr sl sub fun NEXT # SUB	3	Slot scan for # subaddress starting from sub ! This command should be followed by LIST with # number of var_name in a single line.
CAM24 cr sl sub fun NEXT # SLOT	3	Crate scan for # slot starting from sl. ! the sub is KEPT the same over all the different cr ! This command should be followed by LIST with # number of var_name in a single line.
CAM24 cr sl sub fun NAME var_name	3	Single read function with var_name assignment
LIST var_name ...	3	Variable assignment of the previous CAMAC command

Glossary

cr	crate number [0-7]
sl	Slot number [1-30]
sub	subaddress within a slot [0-31]
fun	CAMAC function [0-31]
data ☞ ***#0 or "0 or |0 is replaced by #ffff*** ☜ ☞ ***dat = 0 use decimal 0 (0) instead of #0 or "0 or |0*** ☜	output data work for Write functionformat can be decimal (eg: 15), octal (eg; "17), hexadecimal (eg: #F)
pattern	hexadecimal bit pattern (ex: #200 for slot 10)
var_name	8 characters string defining the CAMAC word

Example ***file.ACQ***

```
For more info  see the doc first then maybe Pierre.
==========================================
! LAM declaration
lam_source     1 #200
!
! Begin of run procedure
cam24   1 30 0 17 data 4        ! reset Inhibit
cam24   1 30 0 17 data 0
cam24   1 3 0 9
cam24   1 10 0 9
cam24   1 3 0 24                 ! disable LAM slot 3 (ADC)
cam24   1 10 0 26                ! enable  LAM slot 10 (TDC)
!
! Event procedure
begin_event
cam24   1 3 0 0  NEXT 3 SUB ! 3 CAF on a0,a1,a2
list adc0 adc1 adc2    ! assignment to that list
cam24   1 10 0 0  next 2 sub
list tdc0 tdc1
end_event
!
! Rearm procedure
cam24   1 3 0 9                 ! CNAF
cam24   1 10 0 9                ! CNAF
rearm_lam
!
! End of run procedure
cam24   1 10 0 24                ! disable LAM slot 10
cam24   1 30 0 17 DATA 4            ! set Inhibit
end_camac
```

Data file format

SusiQ allows to record data in a binary format while acquiring them (see [F] ...TAP). Later data replay will be possible using the same software package (see [F] replay).
The events are stored with a particular format which is described here below. The actual file structure is established during creation of the data file previous the beginning of run. The file is written in 'BINARY' format from the MicroSoft FORTRAN. The block size is 2048 bytes per block.

```
open (   unit=tape_dev,              ! device unit
         file=tape_file,             ! file name
         mode='WRITE',               ! access mode
         blocksize=block_max,        ! block size set to 2048 bytes
         form='BINARY',              ! record format type
         status='NEW',               ! creation of new file
         err=xx)                     ! error handling
```

data file format

run_#	1st evt# of buf	format type	ver_#	# of evt in buffer	event length (n)	evt1[1]	evt1[2]	...
evt1[n]	evt2[1]	evt2[2]	...	evt_last[1]	evt_last[2]	...	evt_last[n]	padding to 2048

Test file.TST

The test command file contains the description of actions to be applied to the acquired data on an event by event basis. Using the names assigned in *file.ACQ* for each CAMAC word, it is possible to generate test or condition flags for further data treatment. At that point one can perform not only statistical tests, but computations based on the incoming data through user callable routines within the test definition file. Three groups of commands are available in the test file:

- ***Test condition commands***: A condition flag in SusiQ is created by each test condition, with the exception of PASSALL which is a predefined condition always set to .TRUE. by Susiq for each event read in. For each newly created test condition, a scalar is associated with it to keep track of the statistic of the success of that test.
- ***Computation commands***: Some basic data manipulations are provided directly within the test file. These commands provide some common functions and avoid the necessity of writing user-specific routines in FORTRAN for simple operations.
- ***Branching commands***: The structure of the test file is built on blocks. The block delimiter is the '*'. This allows jumps on condition from one block to another. Histogram incrementation will always be performed after the exit of the test section.

Test commands table

Syntax	Description
Test Conditions	
GATE flag ACQvar ll hl mask	flag = .TRUE. if ll<=ACQvar.AND.mask<=hl
GATE flag EQUvar ll hl	flag = TRUE. if ll<=EQUvar<=hl
ANYB flag ACQvar mask	flag = TRUE. if (ACQvar .AND. mask) # 0
ALLB flag ACQvar mask	flag = TRUE. if (ACQvar .AND. mask) = mask
AND flag flag1 flag2 ...	flag = .TRUE. if (flag1 .AND. flag2 .AND. ...)
OR flag flag1 flag2 ...	flag = .TRUE. if (flag1 .OR. flag2 .OR. ...)
XOR flag flag1 flag2 ...	flag = .TRUE. if (flag1 .XOR. flag2 .XOR. ...)
BOX flag ACQvar1/EQUvar1 (xvar) ACQvar2/EQUvar2 (yvar) Box_file	flag = .TRUE. if (xvar,yvar) circonscribed by defined shape stored in Box_file
Computation commands	

USER1	Call user routine
USER2	Call user routine
USER3	Call user routine
PION	Call user routine
PROT	Call user routine
USER4 EQUvar ACQvar Iα	Call user routine, passing EQUvar, ACQvar, Iα as parameters
USER5 EQUvar ACQvar α	Call user routine, passing EQUvar, ACQvar, α as parameters
SCAL EQUvar ACQ/EQUvar α β	EQUvar = α*AQC/EQUvar + β
DELY Σ Δ ACQvar1 ACQvar2 α1 α2 β	Σ = ACQvar1 + ACQvar2, Δ = (α1*ACQvar1 - α2*ACQvar2) + β
Branching commands	
*	End block descriptor. This delimits the block number.The block number 1 is delimited by the "=" sign in the first column and the first "*" or the "**". It always has to be put after a JUMP or REJE command
JUMP flag block#	Jump to block# if flag=.TRUE.
REJE flag	reject (disregard) further test commands if flag=.TRUE. ⇨ go to end_of_test descriptor.
**	End_of_test descriptor. This close the test file and must always be present as the last line of the file.

Glossary

ACQvar...	Predefined variable name from *file.ACQ*
EQUvar...	(Pre)defined variable name from *file.EQU*
flag	new condition name
flagx	previously define flag. No more than 18 flags per line
Iα	Integer*4 parameter
α... β...	Real*4 parameters
mask	Integer*4 parameter. The mask can be specified either in decimal (ex: 511), in octal (ex: "777) or in hexadecimal (ex: #FFF or |FFF). The value 0 can be entered as full mask value (see example)
LL	Low limit parameter
HL	High limit parameter
Σ	Sum
Δ	Difference
Boxfile	*file.BOX* box definition (see BOX)
–	minus sign placed in front of input flagx for negation

Example ***file.TST***

```
SusiQ test definition
itstime is created by invoking a dummy logical operation.
and itstime - passall will always assign FALSE to itstime,
but will be used in the user1 routine and passed to the histogram increment routine.
The equivalent function of that  line is : itstime = (.NOT.passall) .AND. .TRUE.
===================================================================
gate gadc0 adc0 0 1000  "0
gate gadc1 adc1 0 1000  "0
and  gadc  gadc0 gadc1
dely asum adiff adc0 adc1 1. 1. 500.
and  itstime -passall
user1
**
```

Histogram *file.HST*

The histogram file contains a list of variables and definitions booking for graphical data representation. Any variable name defined in *file.ACQ* or *file.EQU* can be used in the histogram file with an incremental condition flag attached to it (from *file.TST*.) The definition of histogram is done in a single line.The total number of histograms should not exceed *maxhis* nor should the total number of bins allocated for them exceed *maxtotbin*. Each individual histogram can have a maximum of *maxhstbin* bins.Two types of histogram are available in SusiQ.

- ***standard histogram type*** The definition is composed of its name, number of bins, low and high limit, conditional test name to be applied to it and the actual data variable to be histogrammed.

Standard histogram parameters

Histogram name	The name should be unique among the histogram names

Number of bins	Allocation of the number of bins for that histogram. 'maxhstbin' defines the system limit.
Low limit	Defines the lower data limit for the histogram
Up limit	Defines the upper data limit for the histogram
condition name	pre-declared flag name on which the histogram will be incremented
Data variable	pre-declared ACQvar or EQUvar name which contains the data to be histogrammed

- ***chart recorder type*** This second histogram type allows you to monitor a variable as a function of time or of a test condition. It is sometimes interesting to record the variation of a data word as a function of time. If for example one CAMAC module reads a temperature, pressure or voltage, SusiQ allows you to display, check and take action based on the value of that parameter. If the CAMAC module has its own reading cycle with a LAM generation, SusiQ can be applied in the ***Standard acquisition mode***. If the module does not generate a LAM, SusiQ can do it using the ***Time based Self LAM generator mode***. This mode is required if further action or readout has to be performed before receiving the LAM. Finally if no LAM is required then ***Time based acquisition mode*** is sufficient. This chart recording format contains the chart name, number of bins, low and high limits, the ***time*** keyword followed by the time interval in seconds or the ***test*** keyword followed by the test name and finally the actual data variable name to monitor.

Chart recorder parameters

Chart name	The name should be unique among the histogram and chart names
Number of bins	Allocation of the number of bins for that chart recorder. Each bin will be a time interval.
Low limit	As the time interval is already define in *file.ACQ*, this value is disregarded. Put 0.
Up limit	As the time interval is already define in *file.ACQ*, this value is disregarded. Put 0.
then	
TIME	Keywords for time chart recorder
time_interval	time interval for the chart recorder. Time_interval is given in seconds.[2]
Data variable	pre-declared ACQvar or EQUvar name which contains the data to be histogrammed
or	
TEST	Keywords for chart recorder.
condition name	pre-declared flag name on which the histogram will be incremented
Data variable	pre-declared ACQvar or EQUvar name which contains the data to be histogrammed

*Example **file.HST***

```
SusiQ test definition
=====================================
 adc1   2048 0 2000 passall adc0       ! standard histogram declaration
 adc0   2000 0 2000 passall adc0
 adc0   2000 0 4000 passall adc0
 adc10   2001 -2000 2000 passall adc0
 adc01   500 0 2000 inbox   adc0
 adc02   500 0 2000 inbox1   adc0
 adc03   500 0 2000 inbox2   adc0
 time0  500 0 2000 TIME 1  adc0       ! time based chart recording declaration
 time1  500 0 2000 TEST gdiff adc0    ! test based chart recording declaration
```

Constant file.CST

☞ This file is required ONLY if the user intends to write one of the user routines.

---

[2]This time defines the minimum time interval between two updates. Depending on several speed factors (CPU,CAMAC access, Test package length, User interrupt etc...], this time interval can vary drastically. Note that if any of [D],[S],[H],[M],[N]... and all intermediate pages of Susiq commands are invoked, the acquisition will be frozen which will delay the update of the chart until the next histogram update.

The intention of the constant file is to group together all parameters used in any user routine, such as particle masses, geometric dimensions, calibration parameters, etc. This allows the user to quickly modify any of these parameters while a run is in progress ([Q] cst). If the user wants to add a new parameter name to the constant list, he must recompile (QFALL) the user routines to install the change, and then relink SusiQ to generate a new image file. This procedure is described in *Compilation & Linking procedures*. This file has also been used for lookup table declarations for chamber channel unscrambling. These tables can be accessed by any user routine if the constant include file is present. The names of the 4 available tables are : *LOOKUP(x,128)*, where *x* goes from 1 to 4. Each table can have a maximum of 128 elements. Each table should be separated from the next one by a star '*' or by an equal sign '=' for the last declared table. The constant listing will then follow after the equal sign '=' with one constant declaration per line as : <parameter_value> <parameter_name>.This array can also be used by the user as common memory space within SusiQ between the different user routines. Be aware, however, that every time the command [Q] CST is performed, *file.CST* will be reread and will overwrite the contents of the array.

Example file.CST

```
Constant file for SUSIQ test package
Lookup table for Fast chamber
*******************************************************************
  4  8 12 16 20 24 28 32 36 40 44 48 52 56 60 64    !lookup( 1 to 16)
  3  7 11 15 19 23 27 31 35 39 43 47 51 55 59 63    !lookup(17 to 32)
 61 57 53 49 45 41 37 33 29 25 21 17 13  9  5  1    !lookup(33 to 48)
 62 58 54 50 46 42 38 34 30 26 22 18 14 10  6  2    !lookup(49 to 64)
===================================================================
1000.   numout            ! Default #_of_event for OUTUSER call
.0324   decal             ! Delta E1 calibration factor
```

Equivalence *file.EQU*

☞ This file is required ONLY if the user intends to write one of the user routines.

The variables declared in *file.EQU*. allow a bridge between SusiQ and the user routines, such that the user can pass variables from his own routines to SusiQ for further tests or histogramming, or to get access to variables from SusiQ in his own routines. All the names in this file will be **automatically** declared as **REAL*4**. No declaration is required by the user routine as long as the user compiles it with QFALL (see *Compilation & Linking Procedures*). All the names must be unique. Up to 18 names can be put in a single line. Each name will appear in the ***peeper*** page. If a variable is not used in any user routine, a default value of -1000000 is assigned to it. This value will show as '*******' in the ***peeper*** page.

Example file.EQU

```
SusiQ equivalence names
==========================================
udf  ovf  mean  sig
ratio
```

Dotplot file.DOT

The dotplot file can be written before starting a SusiQ run if desired. The intention however, was that online dotplot definitions would determine the save_set at the end of the run. As you acquire data, it is possible to define new dotplots. If at the end of a SusiQ session, you exit properly (i.e. without crashing, ctl-alt-del or equivalent), SusiQ will save all the previous dotplot definitions in a file labeled *file.DOT* provided that you have defined the name of that file. i.e. **DOTFILE** *file.DOT* in *susiq.dft.* The format of the dotplot in the *file.DOT* is:

Xvar_name X_low_limit X_up_limit Yvar_name Y_low_limit Y_up_limit flag_name

Example ***file.DOT***

```
 ---> SUSIQ Dot definition <---
 creation/update date  15:39:20.09   13-AUG-1992
==================================================
 ADC0        .000 2000.000 ADC1        .000 2000.000 PASSALL
 ADC0        .000 2000.000 ADC1        .000 2000.000 INBOX
 ADC0        .000 2000.000 ADC1        .000 2000.000 INBOX1
```

Box file.BOX

The box file is generated by SusiQ. After a successful save this file will reflect the latest definition of the box contour entered during a [B] command. *file.box* is an ASCII file but it should not be modified. The first line contains the file name and the X and Y variable names. The second line contains the version number and the number of verticies defining that box. The remaining lines contain the X,Y pairs of the vertices.

SusiQ Internal Tools

Internal Tools refers to all routines the user may access from within SusiQ user routines. The correct use may require some specific declarations.

Internal Tools table

Susgen	Random event generator
Outuser	User specified data recorder
Clean_f	Check for proper file access
Get_hist	Histogram characteristic retriever
EventIX	Raw data index retriever
CalibIX	Computed quantity index retriever
TestIX	Test condition index retriever
Clhist	Clear histogram content
Iprompt	User input parameter facility
Oprompt	User output parameter facility
UBOR	User Begin_of_Run routine
UEOR	User End_of_Run routine
User flags	Compilation state status

susgen routine

This routine generates a gaussian function centered at 1000 with a sigma of 100. Each word which would otherwise be read from CAMAC will instead be generated by *susgen* independently of the event length, . This routine resides in the user directory and can be modified to match user specifications. It is primarly used to debug the test package.

call susgen (h)

➚ **h** : return data variable

outuser routine

This routine allows the user to save a set of raw data or computed data into a file with a user specified format. The call to this routine should be done within one of the user routines. The format with which the data are saved is entirely at the discretion of the user. The standard application of this routine is to generate a skim file in ASCII format for later processing stage with other programs (i.e. PLOTDATA, DataBase, SpreadSheet, etc...). To operate the *Outuser* routine, the system flags DEBUG1 and DEBUG2 are used. While a run is in progress, toggeling the DEBUG1 flag ([F] DEBUG1) will interrupt the analyzer on its next pass into the *Outuser* routine. Then *Outuser* will request a file name for storing the data and resume the acquisition. From that point on every time the analyzer processes an event, the specified data will be recorded in the file. The number of events to record is declared through a constant *numout*. This parameter should be defined in *file.CST*. The call to *Outuser* should be done through one of the user routines and the user routines should be called from within *file.TST*.

call outuser

✧ no argument

clean_f routine

This routine is used to prepare a program to open a file. It first checks if the character string referring to the file name is too long. Next it composes the full file name (body file + extension) using the std_extention string if the input string does not contain the extension (defined by a dot extention delimiter). Finally it checks whether or not given file already exists. If it does, it will rename the file to a backup file name with an extension provided by the call. If the file does not exists, the routine will return a full file name.

call clean_f (file_name, file_length,std_extention,backup_extention,*err)

- ➘ **file_name** : file name to be checked
- ➘ **file_length** : number of characters composing the file_name
- ➘ **std_extention** : default extention if not provided in the file name
- ➘ **backup_extention** : default extention if the file already exists under the same name
- ➚ **file_name** : valid file name for further processing
- ➚ **file_length** : file name length
- ➚ ***err** : error label in case of file name too long.

get_hist routine

This routine retrieves the histogram characteristics for a given histogram number. The histogram number should refer to a valid number otherwise *get_hist* returns 0 for all the output variables. The standard use of this routine is to extract the mean of a variable over time. It has to be used in conjunction with the chart recorder histogram type. In the UEOR routine, it can be used extensively for histrogam post-processing and generating a result file ready for a spreadsheet style program. In the example at the end of this section, the call to get_hist is made from user1 for mean variation display over a given number of events (kadc0, itstime). As long they have been defined in *file.EQU*, any variables used in the routine will be passed to the SusiQ test and histogram package where further processes can be applied to them.

call get_hist (Histo_number, Xlow, Xhigh, Ysum, Undf, Ovfl, Mean, Sigma)

- ➘ **Histo_number** : Histogram number
- ➘ **Xlow** : Low data limit of the histo (0 for full)
- ➘ **Xhigh** : High data limit of the histo (0 for full)
- ➚ **Ysum** : Total count content within Xlow & Xhigh
- ➚ **Undf** : Total underflow count below Xlow
- ➚ **Ovfl** : Total overflow count above Xhigh
- ➚ **Mean** : weighted mean of the histo within the Xlow & Xhigh limits
- ➚ **Sigma** : Standard deviation of the histo within Xlow & Xhigh limits

testIX function

This function allows you to retrieve the index of a test condition (flag) when the name of the test is given as input. The array to which the index can be applied is **FLAG().** Used in user routines it gives you the ability to be more code independent. The index of a test condition is established during the compilation of the test file. From that point on it will not change until the next test compilation. This means the index can be retrieved only once i.e. after each test compilation. To do so, it is a good practice to search for it once during the first pass through the user routine. Some system flags are provided to allow you detect a new test compilation. *newtst1, newtst2* are set to .TRUE. after each test compilation. These are available to you for this purpose.

☞ The function name must be declared as LOGICAL*1 parameter.

index = TestIX (test_condition_name)

➘ **test_condition_name** : predeclared test name from the *file.TST*
➚ **index** : returned index from the **FLAG()** array of the corresponding name.

calibIX function

This function is similar to the TestIX function, but instead of the flag array, calibIX retrieves the index of the computed quantity. The array to which the index can be applied is **CALIB()**. Its main purpose is to allow looping in user routines when repetitive operations have to be done on consecutive computed quantities.

☞ The function name has to be declared as INTEGER*4 parameter.

index = calibIX (computed_quantity_name)

➘ **computed_quantity_name** : predeclared name from the file.EQU
➚ **index** : returned index from the **CALIB()** array of the corresponding name.

eventIX function

This function is similar to the CalibIX function, but instead of the computed array, eventIX retrieves the index of the raw data array. The array to which the index can be applied is **IDATA()**. Its purpose is to allow looping in user routines when repetitive operations have to be done on consecutive raw data words.

☞ The function name has to be declared as INTEGER*4 parameter.

index = eventIX (raw_data_name)

➘ **raw_data_name** : predeclared name from *file.ACQ*
➚ **index** : returned index from the **IDATA()** array of the corresponding name.

clhist routine

This routine is available for clearing the contents of a given histogram. The histogram must be referred to by its number.

call clhist ('ONE', histo_number)

➘ **histo_number** : histogram number
☞ Extra caution has to be taken regarding the histogram number. As the histogram file can be edited at any time, the histogram number may point to a different histogram if another histogram has been added prior the requested one into the *file.HST*.

UBOR routine

The **User_Begin_Of_Run** routine provides a last chance to do preliminary work before the run starts. *UBOR.for* already contains all the necessary include files for accessing any variable in the system. This routine is activated after the opening of the CAMAC channel to allow access to the CAMAC crate. The standard use is for initialization of some variables.

UEOR routine

The **User_End_Of_Run** routine is executed just before the CAMAC channel is closed to allow user access to CAMAC if necessary. All the files opened by the user must be closed at that point. *UEOR.for* already contains all the necessary include files for accessing any variable in the system.

User Flags

Within the user routines, depending on the application, it may be necessary to know if a compilation of either the test file, histogram file or constant file has been done. This detection can be used to execute specific code extracting the corresponding index of the variable names or particular computation, e.g: eventIX, calibIX, testIX, get_hist. For that purpose, SusiQ provides a series of system flag that the user may toggle or acknowledge at any time. These flags are defined in the *susictl.inc* include file which must be present in the corresponding user routine.

User Flags Table

newtst1, newtst2	Set to TRUE after test compilation ([q] acq , [q] tst)
newbor1 newbor2	set to TRUE during Begin Of Run procedure ([c] begin)
newcst1 newcst2	set to TRUE after constant compilation ([q] cst)
debug1, debug2	set to FALSE during SusiQ initialization. Used for instance in the *outuser* routine for data skimming.
private2, private3, private4	set to FALSE during SusiQ initialization. Used for system debugging, but available to the user as well.

Examples Internal Tools I (get_hist, newtst,testIX,clhist)

```
c****SUSIGPS Data Analyser*****************
c "USER1.for"
c**************************************************
c Xenon test flasher
subroutine user1

      parameter (pi=3.1415926)
       include 'susictl.inc'
       include 'susiacq.inc'
       include 'susitst.inc'
       include 'susicof.inc'
       include 'susiequ.inc'
       include 'susievt.inc'
       integer*4       testix,ix
       if (newtst1) then
          ix = testix ('itstime')
          write (6,*) 'Itstime : ',ix
          newtst1 = .FALSE.
       end if
       if (adc1.le.0) return
       ratio = kdif*adc0/adc1
       if (now.gt.kadc0) then
          call get_hist(1,0.,1000.,ssum,udf,ovf,mean,sig)
          mean = mean*1000.
          call clhist  ('ONE',1)
          flag(ix) = .TRUE.
          now=0
       endif
       now=now+1
       ratio  = -1.*ratio
       return
       end
```

Examples Internal Tools II (Outuser routine, Iprompt, Oprompt, clean_f debug1,debug2)

```
SusiQ test definition
============================================
gate gadc0 adc0 0 1000 "0
gate gadc1 adc1 0 1000 "0
and  gadc  gadc0 gadc1
dely asum adiff adc0 adc1 1. 1. 500.
and  itstime -passall
user1
**
-------------------------------------------------------------------------------------------
c***********************SUSIQ Data Analyser********
c"USER1.for"
c***********************************************************
       subroutine user1
       parameter (pi=3.1415926)
       include 'susictl.inc'
       include 'susiacq.inc'
       include 'susitst.inc'
       include 'susicof.inc'
       include 'susiequ.inc'
       include 'susievt.inc'
       call outuser
       return
       end
-------------------------------------------------------------------------------------------
c***********************SUSIQ Data analyzer**********
```

```
c      "outUSER.for"
c*************************************************************
       include 'fgraph.fi'
       subroutine outuser
       include 'fgraph.fd'
       include 'susictl.inc'
       include 'susiacq.inc'
       include 'susihis.inc'
       include 'susitst.inc'
       include 'susicof.inc'
       include 'susiequ.inc'
       include 'susievt.inc'
       character        bel/7/
       character*12     string
       integer*4        nchar,counter
       record /rccoord/ curpos
      if (debug1) then
         call iprompt    ('File name  : ',nchar,string)
         if (nchar.eq.0) return
         call clean_f (string,nchar,'DDD','DAT',*99)
         open (unit=8,file=string(1:nchar),status='NEW',err=99)
         debug1 = .FALSE.
         write (*,'(1x,a\)') bel
         counter = 0
         debug2 = .TRUE.
         call oprompt('File name '//string(1:nchar)//' open!')
       else if (debug2) then
         if (counter .lt. INT(numout)) then
cccccc------------------------------------------------------------------
c        USER! place your variables (RAW/COMPUTED/...) in a WRITE
c        format. DON'T forget the ERR= statement!!
cccccc------------------------------------------------------------------
c
           write (8,10,err=9999) adc0,tdc0
10           format (1x,i6,2x,i6)
c
cccccc------------------------------------------------------------------
c        USER! To change the number of event to be taken in that file
c        change the value of the variable NUMOUT in the *.CST definition file.
c        To get this routine started, Turn ON at any time during a
c        running SusiQ the DEBUG1 flag.
c        You will be prompted for a filename to be entered.
c        After NUMOUT event, the file will be closed.
c        But you HAVE TO CALL this routine through the USERX routine
c        Like it is done in the USER1.FOR routine in that directory.
c        Good LUCK, Pierre
cccccc------------------------------------------------------------------
           counter = counter + 1
         else
           call oprompt ('File name '//string(1:nchar)//' closed '
     *//'                                    ')
           debug2 = .FALSE.
           close (unit=8)
           write (*,'(1x,a\)') bel
         end if
       end if
       return
99     call Oprompt (' %Outuser-F- Error while file opening')
9999   return
       end
```

Examples Internal Tools III (susgen)

```
c
c===   This routine generates a gaussian centered at 1000 with a sigma of 100
c===   This routine is called ONLY if you turn SUSGEN on
c===   User you can modify this routine to simulate your data if you wish!
       real*4    h,v,t
10       call random(h)
       h = 2000. * h
       v=exp(-0.5*((h-1000.)/100.)**2)
       call random(t)
       if (t .gt. v) goto 10
         h = h
       Return
       end
```

External Tools

External Tools refers to stand alone programs that the user may run outside of SusiQ. These programs are in one way or another related to SusiQ. If the command *loadsq <dev>:* has been executed, all external tools programs are on the DOS path and hence available directly by entering their names.

☞ Restrictions may be applied to some of them due to the hardware installed.

External Tools table

SH	Susiq Help (equivalent of [M])
DDRA	Disk DRA (equivalent of [D])
HM	Histogram manager
FCNA	CAMAC access tool
CAMTEST	CAMAC crate scanner

SH.exe [SusiQ Help]

This program allows the user to invoke the SusiQ help feature from outside SusiQ. The contents of this help are identical to those invoked with the [H] command within SusiQ.

DDRA.exe [histogram display package]

This program allows the user to display a previously saved SusiQ histogram file. The layout of the histogram display is identical to the one from the [D] command within SusiQ. This program will become obsolete in future SusiQ versions. The replacement program will be HM (Chaos Histogram Manager)

HM.exe [CHAOS Histogram Manager]

This program is a modified version of the original HM program (Histogram Manager) written by Stefan Ritt for the Susi area data acquisition at PSI. The modifications include the option to retrieve histogram files saved under several formats such as SusiQ, Online-Susi_Ybos through network (CHAOS experiment), Susi_Regina. HM automatically recognizes the extension **.SQH.** A list of defined histograms is then displayed. The advantage of HM over DDRA is the high capability of display, fitting and information retrieval. HM is a mouse oriented program which makes its use much easier for newcomers.

Some hints:

- HM has a built-in data base for automatic fitting. This feature has been heavily used at PSI but not at TRIUMF. This means that not much experience with it is available at TRIUMF.
- This program is intended to be used for peak fitting and arithmetic operations on different histograms. This implies that HM does not retrieve a full set of histograms in one shot, but rather requests an entry for each histogram window from the same or a different save_set file. The loading of histograms can be done in several ways:

  1) activate the desired window
  2) load file *.SQH , selection of the histogram
  3) activate next desired window
  4) repeat from step 2)

  or

  1) activate desired window
  2) load file *.SQH , selection of the histogram
  3) drag histo to new desired window , hit copy
  4) hit F4 for next histogram
  5) repeat from step 3)

  or

  If a 3 button mouse is installed, the third button provides a pop-up menu for
  a histogram listing of the open *file.SQH*

FCNA.exe [Interactive CAMAC access]

This program allows the user to issue single/multiple CAMAC commands interactively. FCNA works only with the KCS hardware configuration. It's use is simple and a comprehensive help is provided within the program.

CAMTEST.exe [CAMAC crate scanner]

This program, written by a Russian group, provides complete information regarding the functionality of any module inserted into a specific CAMAC crate. It returns a list of all the valid functions on all the subaddresses for all the modules of a crate under a given criteria which can be either : X response, Q response, X and Q response. CAMTEST works only with the KCS hardware configuration.

SusiQ Commands

The first level of command within SusiQ is invoked through a single keystroke command. The key assigned to each command is usually the first character of the command. All interactions with SusiQ must pass through one of the command keys. When a key is activated, the display screen presents the sub-level command listing. From that point on, the acquisition is frozen. No further data will be taken and analyzed until an action has been taken in response to the command in the prompt area.

SusiQ Command table

Group	1st Level	2nd Level etc...	parameters...	...
Box	[B]	file | L	box# | verticies	
Cmd	[C]	BEGIN	run_number	{file}
	[C]	CLRTST		
	[C]	CLRHST		
	[C]	SAVTST	file	
	[C]	SAVHST	file	
	[C]	END		
Dra	[D]	... see DRA for internal command listing		
Erase	[E]	histo_name | histo_number | *		
Flag	[F]	SAVTST | SAVHST |		
		CLRTST | CLRHST |		
		BEEP | DEBUG1 | DEBUG2 |		
		SUSGEN | CAMAC | SUSTAP | CAMTAP | REPLAY | TIMGEN | RANGEN|		
Help	[H]			
Limit	[L]	Event	number of event	
		Test	test name	number of test
		Histo	histogram name | histogram number	bin content
Manual	[M]			
News	[N]			
SusiQ	[Q]	ACQ | TST | HST | CST		
		DOS	dos command	
		Edit	file name	
		SAVE	file name	
		RESTORE	file name	
Set/Show	[S]	LOck		
		STatus		
		WAITLAM	number	
		EXPT#	number	
		EXPERIMENT	name	
Update	[U]	EFficiencies	normalization test name	update_frq
		HIstogram	histogram name | histogram number	update frq
		DOtplot...	dotplot number	
		PEeper	update frequency	
Exit	[X]	{ if run => End run ? Y | N }	Y | N	
Zoom	[Z]	# | -# | S | B | R | O | C		

[B]... for Box

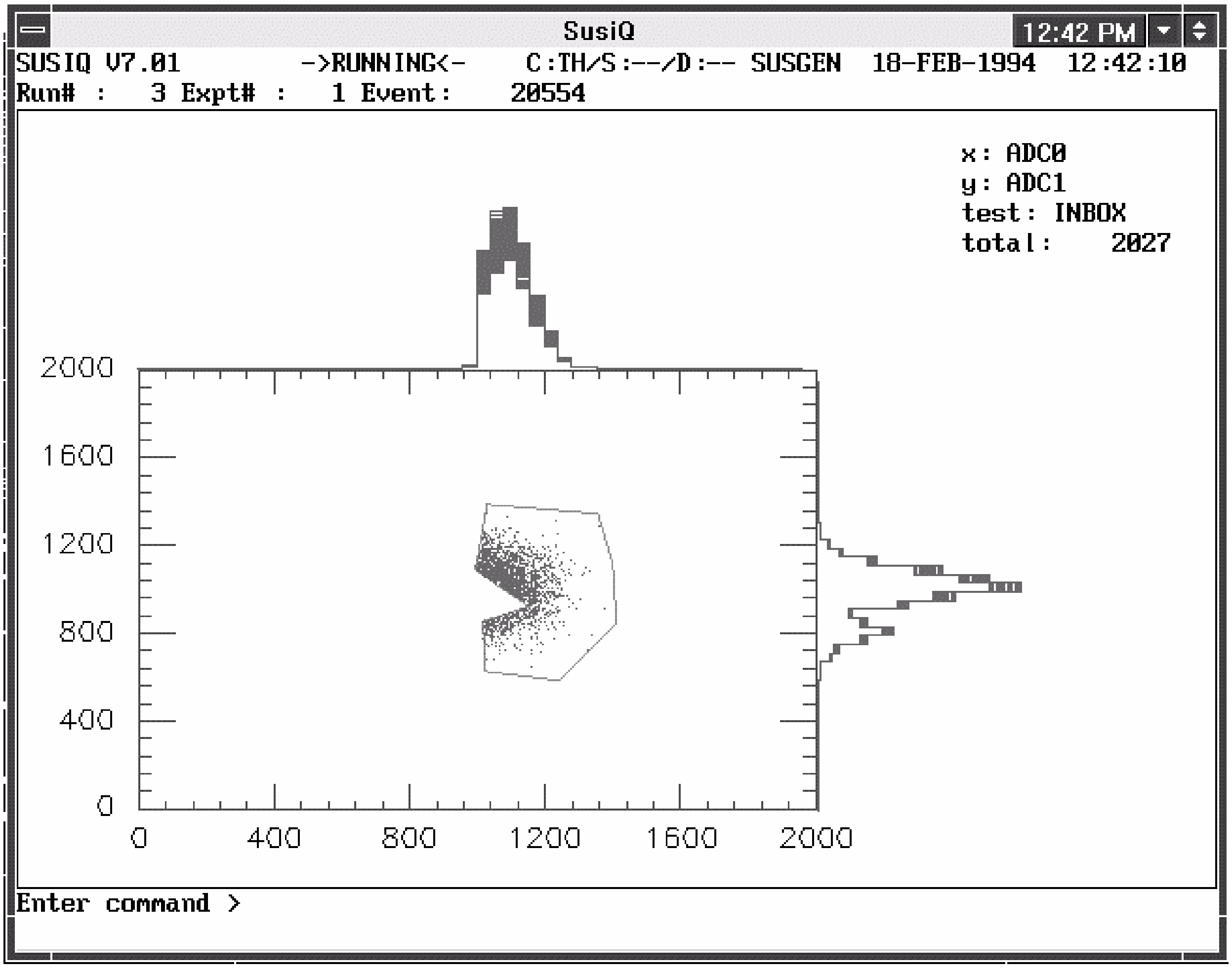
Box allows you to define graphically a geometrical region as a conditional parameter. This statement implies that you know the shape of the correlation of the 2 variables before the test can be applied based on that condition. This will necessitate a double path through the procedure for establishing an operative test box. During this process, the command will display a box setting status page. This listing can also be obtained without starting a box booking by the command [B] L<cr>. This page is composed of :

Box options table

Box_#	Slot position
Status	Access status of that slot
— Accessible	box file and X/Y match requirement
— Overwrite	X/Y match only for temporary test box
— Protected	X/Y do not match test file
Box file	Box file name seen in test file or recently created
X_var	X variable involved in the current slot
Y_var	Y variable involved in the current slot
Nvert/type	Current number of definition for that slot
Status	Status of the slot content
— empty	slot empty
— active	loaded in memory

The necessary steps to activate a boxfile are the following:

1) Setup the test file to reflect the expected BOX condition that you would like to see.
2) Compile the test file and start a run.
3) Setup the dotplot of X_var vs Y_var gated on Test_var of interest.
4) Display the newly created dotplot.
5) Invoke the BOX command, and specify the file name that you gave in the test file.
6) Draw the contour of the test box with the mouse by pressing the left button for ending the current vertex and the right button for closing the contour.
7) Select the slot number reflecting your requirement.
8) Hit return to return to the dotplot. From that point the test box is active. Its results can be checked from the efficiency table.



<[UP] do> + <[B] cr> Dotplot with projection and Box overlay

Examples Box

2SusiQ test definition
===
gate adc0 adc0 10 1000 #3FF
gate gadc2 adc2 10 1000 #3FF
gate gtdc0 tdc0 5 2000 "0
gate gtdc1 tdc1 5 4000 "0
box inbox adc0 adc1 adc01.box
and nbox passall -inbox
box inbox1 adc0 adc1 adc101.box
box inbox2 adc0 adc1 adc201.box
and gdiff gadc0 gadc2 gtdc0
reje -gdiff
*
scal cadc0 adc0 1. 0.
scal cadc2 adc2 1.06 50.
user1
**
Susiq sequence for defining a new box while SusiQ is running:

→[Q] e<cr> susiq.tst
Invoke editor from within SusiQ. Edit the file as example above and exit with save
→[Q] acq<cr>
Recompile test,histo on the fly, the run will not be ended, but the test and histo content will be cleared.
→[U] ef<cr><cr>
Display the efficiency table, check that the test inbox counter shows 0 counts.
→[U] do<cr> adc0<cr> 0<cr> 2000<cr> adc1<cr> 0<cr> 2000<cr> passall<cr> <cr><cr>
Display the dotplot of the 2 variables of interest
→[B] adc01.box<cr> Y<cr> ... define region... close box... 1<cr><cr>
Invoke box definition it will prompt you for a file name. Defaults to first box file of the test, then it will ask for creation or update. After completion of the recording, load it in the appropriate slot. return to the dotplot screen.
→[B] <cr>
Overlay the box file contour on the dotplot.
→[U] ef<cr><cr>
Display the efficiency table, check the test inbox for count incrementation.

[C]... for Command

This command group includes important commands for run control. The secondary command must be typed in full.

[C] BEGIN ...

This command starts a new run corresponding to the Status display. Input parameters will be requested depending on the acquisition mode.

- SUSGEN,CAMAC ...experiment_# ...run number
- SUSTAP,CAMTAP,REPLAY ...experiment_# ...run number ...file name

The actual operation sequence after acceptance of the input parameters is:

1) do CLRTST if requested
2) do CLRHST if requested
3) Open CAMAC channel
4) Set system flag BOR
5) perform *ubor* routine
6) Sound beep if requested
7) update status window
8) Wait for event...

[C] END

This command closes an active run in the following sequence :

1) do SAVHST with default file name *HSTrun_#.SQH* if requested
2) do SAVTST with default file name *TSTrun_#.SQT* if requested
3) Set system flag EOR
4) perform *ueor* routine
5) Sound beep if requested
6) Close CAMAC channel
7) update status window
8) wait for user input...

[C] CLRHST

This command clears the contents of all the defined histograms. It can be given at any time.

[C] CLRTST

This command clears the content of all the test results. It can be given at any time.

[C] SAVHST...

This command saves the definitions and contents of all the histograms into *file.SQH*. A file name is requested at the command prompt. The format of the file will be readable from DDRA and HM. If no file extension is given *.SQH* will be used. If a file already exists under the same name, it will rename the original file to *file.BQH* and save the requested file under *file.SQH*. The full file name under DOS should not be longer than 12 characters. Make sure that the body file name does not exceed 8 characters.

☞ HM recognizes the SusiQ format ONLY if the file has the *.SQH* extension.

[C] SAVTST...

This command saves the definition and test results of all the tests into *file .SQT*. A file name is requested at the command prompt. The format of the file is essentially the same as the *file.TST* file definition with test result and ratio normalized to a given test variable. If no file extension is given *.SQT* will be used. HM recognizes the SusiQ files only if the extention is *SQH*. If a file already exists under the same name, it will rename the original file to *file.BQT* and save the requested file to *file.SQT*. The full file name under DOS should not be longer than 12 characters. Make sure that the body file name does not exceed 8 characters.

[C]... Example

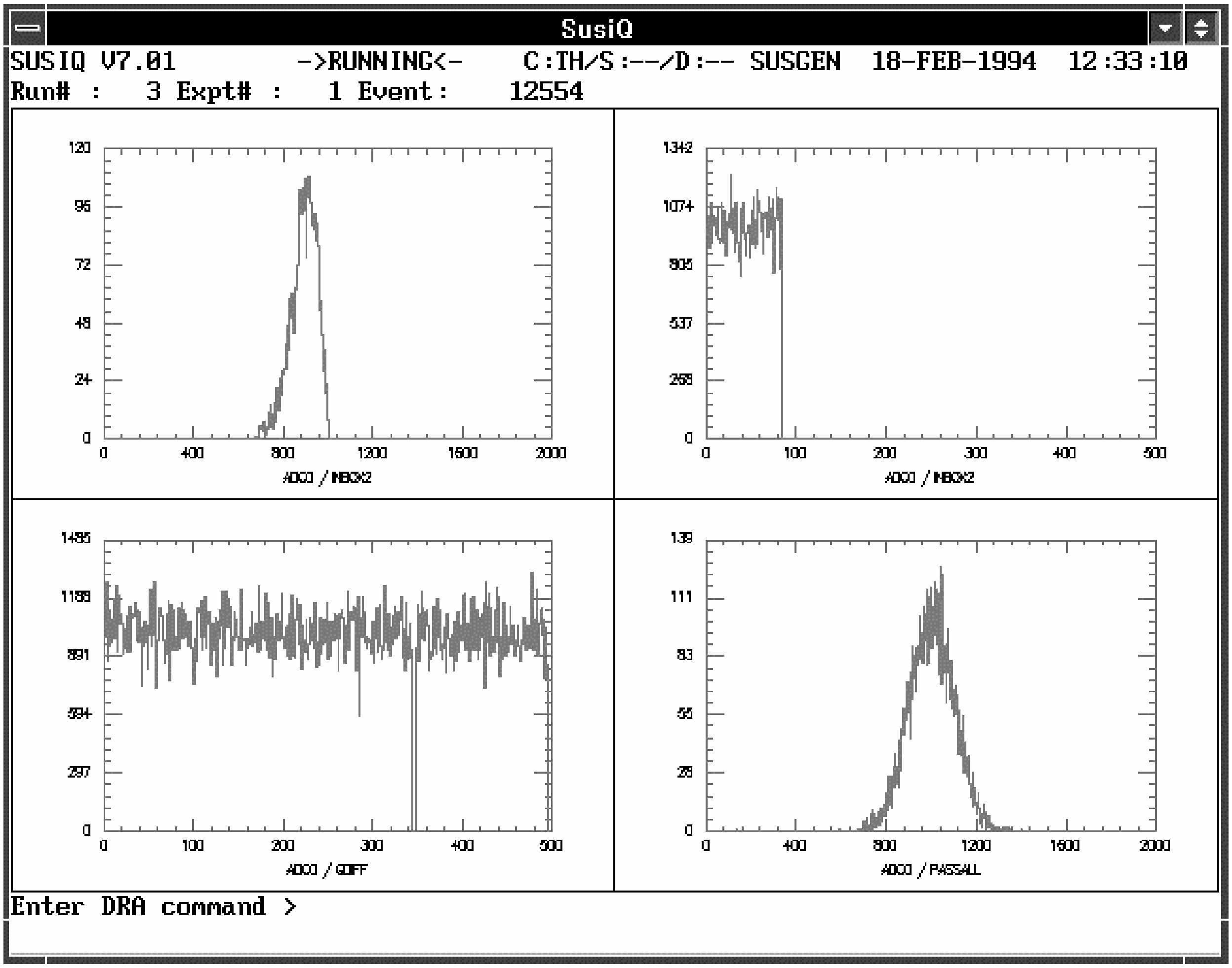
→[C] BEGIN <cr> 3 cr>
Start and force run number 3
→[C] SAVTST <cr> xe1203
Xe1203.SQT created
→[C] SAVHST <cr> xe1203
Xe1203.SQH created

[D]... for Draw

This command invokes the online graphics package. All subcommands are given with 2 characters.

DRA command table

BO	Box display. File name requested.
CH	Clear a histogram contents.
CL	Clear the screen.
CU	Interactive bin value and contents with mouse displacement. The bin,data,bin content are interactively updated on the top of the display. Hit RB to return to main DRA command prompt.
EN	End DRA package (same as EX)
ER	Toggle screen erasing (for overlay). Used for overlaying several histograms. Each of them will be displayed in a different colour.
EX	Exit DRA package (same as EN)
FI	Integrate between limits defined by mouse. Underflow,overflow,bin,data,bin content,mean,sigma are returned at the bottom of the display after having selected 2 points with the mouse.
FX	Freeze X-scale for Update mode, (toggle). Retain the X scale when exiting Dra and apply it to the next [U] HI command
HE	Help.
HI	Display histogram. Histogram number or name requested.
IN	Restore histogram from save_set *file.SQH.*
LH	List booked histograms.
LI	List bin content of a histogram.
NH	Display Next histogram.
PH	Display Previous histogram.
QU	Select single, 2 or 4 histogram page display.
RH	Refresh display.
SA	Save current definition and histogram contents.
XS	Manual X scale setting. Xlow and Xhigh requested manually.
YS	Manual Y scale setting. Ylow and Yhigh requested manually.
ZO	Display predefined zoom setting. zoom number requested. No zoom table is given.



<[D] qu> option with standard histo, Time, Test chart

[E]... for Erase

The command erases the contents of the given histogram. The histogram name or number will be requested at the command prompt. No default is allowed. The option "*" can be used for erasing all histograms contents (equivalent to **[C] CLRHST**).

[F]... for Flag

This command allows you to set or toggle some of the system/user flags. All these flags can be set from the *susiq.dft* setup file for initial settings.

Flag table

CLRTST	Clears test before BOR
CLRHST	Clears the histos contents before BOR
SAVTST	Save all test results into a file at EOR under the name TSTxxx.SQT where xxx is the run number
SAVHST	Save all the histogram definition and contents into a file at EOR under the name HSTxxx.SQH, where xxx is the run number
DEBUG1	Toggle the user flag DEBUG1. (can be used internally in any user routines)
DEBUG2	Toggle the user flag DEBUG2. (can be used internally in any user routines)
SUSGEN	Set acquisition mode using the *susgen routine*
SUSTAP	Set acquisition mode using the *susgen routine* with recording to of the raw data to a file
CAMAC	Set acquisition mode for CAMAC
CAMTAP	Set acquisition mode for CAMAC with recording of the raw data to a file

REPLAY	Set acquisition mode for accessing data from a data file
BEEP	Toggle the BOR and EOR beep sound The BOR is define by the files *SOUND.2* The EOR is define by the file *SOUND.1*

Examples Flags

→[F] savhst<cr>
Toggles EOR Flag for histogram save_set
→[F] beep<cr>
Toggle the BOR/EOR bell
→[F] CAMTAP<cr>
Switch from previous setting to CAMAC with recording disk file (taping)

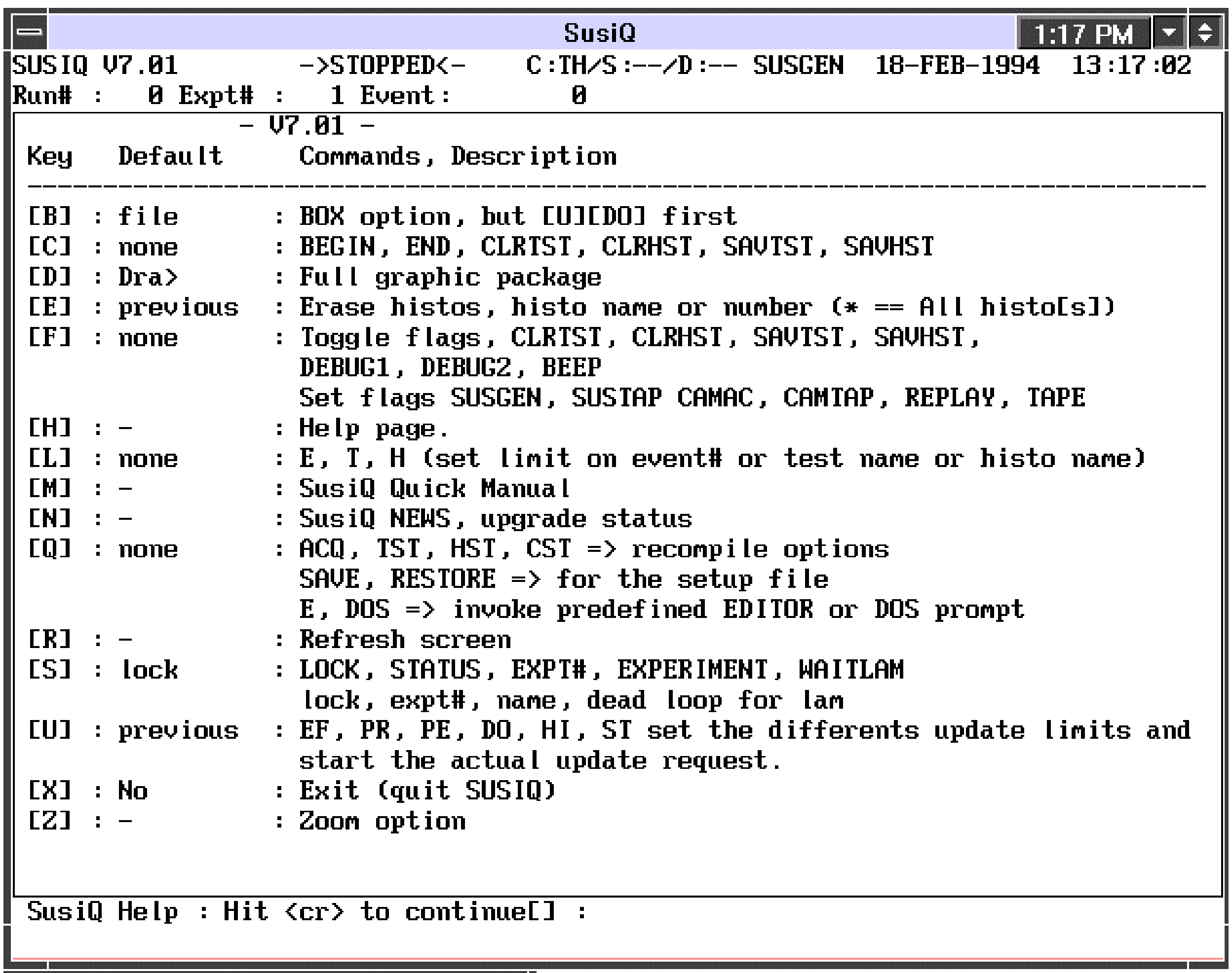
[G] for Dump BMP file

This command allow you to generate a Bit Map File (BMP) graphical file. This command is active ONLY if you are running the [U] HI or [U] DO. At the prompt line, you will be asked for a file name. In case the file name already exists, the previous file will be saved under the same name with the extension '.BMK' If a name is entered without extension, '.BMP' will be automatically added.

[H] for Help

Invokes a single help page summarizing single keystroke first level commands.

☞ If a run is in progress, the run will be paused until any key is pressed again.



```
SusiQ                                                    1:17 PM
SUSIQ V7.01        ->STOPPED<-    C:TH/S:--/D:-- SUSGEN  18-FEB-1994  13:17:02
Run# :   0 Expt# :   1 Event:        0
                 - V7.01 -
 Key   Default      Commands, Description
 ---------------------------------------------------------------------------
 [B] : file       : BOX option, but [U][DO] first
 [C] : none       : BEGIN, END, CLRTST, CLRHST, SAVTST, SAVHST
 [D] : Dra>       : Full graphic package
 [E] : previous   : Erase histos, histo name or number (* == All histo[s])
 [F] : none       : Toggle flags, CLRTST, CLRHST, SAVTST, SAVHST,
                    DEBUG1, DEBUG2, BEEP
                    Set flags SUSGEN, SUSTAP CAMAC, CAMTAP, REPLAY, TAPE
 [H] : -          : Help page.
 [L] : none       : E, T, H (set limit on event# or test name or histo name)
 [M] : -          : SusiQ Quick Manual
 [N] : -          : SusiQ NEWS, upgrade status
 [Q] : none       : ACQ, TST, HST, CST => recompile options
                    SAVE, RESTORE => for the setup file
                    E, DOS => invoke predefined EDITOR or DOS prompt
 [R] : -          : Refresh screen
 [S] : lock       : LOCK, STATUS, EXPT#, EXPERIMENT, WAITLAM
                    lock, expt#, name, dead loop for lam
 [U] : previous   : EF, PR, PE, DO, HI, ST set the differents update limits and
                    start the actual update request.
 [X] : No         : Exit (quit SUSIQ)
 [Z] : -          : Zoom option

 SusiQ Help : Hit <cr> to continue[] :
```

<[H]> Help page screen

[L]... for Limit

This command sets one of the three different run limits. By default a run is unlimited.

- Event *limit#* — Set run limit based on the number of events taken
- Histo *Hname/Hnum limit#* — Set run limit based on the first channel of the given histogram reaching the limit
- Test *Tname limit#* — Set run limit based on the number of events passing the given test

Examples Limits

→[L] E<cr> 5000<cr>
Run will be ended after event_number 5000
→[L] H<cr> adc0<cr> 200<cr>
Run will be ended when any first bin of histo ADC0 reaches 200
→[L] T<cr>inbox<cr> 2000<cr>
Run will be ended after inbox contains 2000 counts.

[M] for Manual

Invokes the *Quick Online SusiQ Manual*. A series of ASCII pages is available to you by selecting a topic with a single keystroke.

[N] for News

Invokes a single page advertising the latest changes related to SusiQ.

[P] for Dump ASCII file

This command allow you to generate an ASCII file containing the bin content of thecurrent displayed histogram. This command is active ONLY if you are running the [U] HI histogram update page. At the prompt line, you will be asked for a file name. In case the file name already exists, the previous file will be savedunder the same name with the extension '.BCS' If a name is entered without extension, '.ASC' will be automatically added.

[Q]... for SusiQ

This command invokes the SusiQ online compilation option and other specialized functions.

[Q] command table

ACQ	Compiles & Loads the Acquisition, Test & Histo package in that order
TST	Compiles & Loads the Test & Histo package in that order
HST	Compiles & Loads the Histo package only
CST	Compiles & Loads the Constant package
SAVE *file.dft*	Saves the setup file into the given file name
RESTORE *file.dft*	Restores the setup file from the given file name
DOS *command*	DOS shell. Prompt you for single/multiple DOS command.
E *file.ext*	Invokes an Editor for file modification.

Examples SusiQ

→ e<cr>*file.tst*<cr>
edit file (run may still be active)
file editing... save/exit...
test file not yet replaced in SusiQ. Run still in progress with old version
→ [Q] acq<cr>
compile test... clear test and histo carry on...
Same sequence for histogram can be applied. with [Q] hst<cr>

[S]... for SHOW/SET

Displays the actual system configuration and allows you to change restricted system parameters.

☞ For instance, if the label on the histogram display does not appear, check the status line 'graphic font' for the actual path of the font file.

[S] command table

LOck	Display a dummy warning page to inform that SusiQ is in progress. Return to SusiQ command by pressing any key (try to deceive meddlers).
EXP# *expt#*	Change the experiment number
EXPERIMENT *exp_name*	Change the experiment name used for Save_set file
STatus	Display most of the system/user parameters
WAITLAM	Change the *loop count* for the acquisition

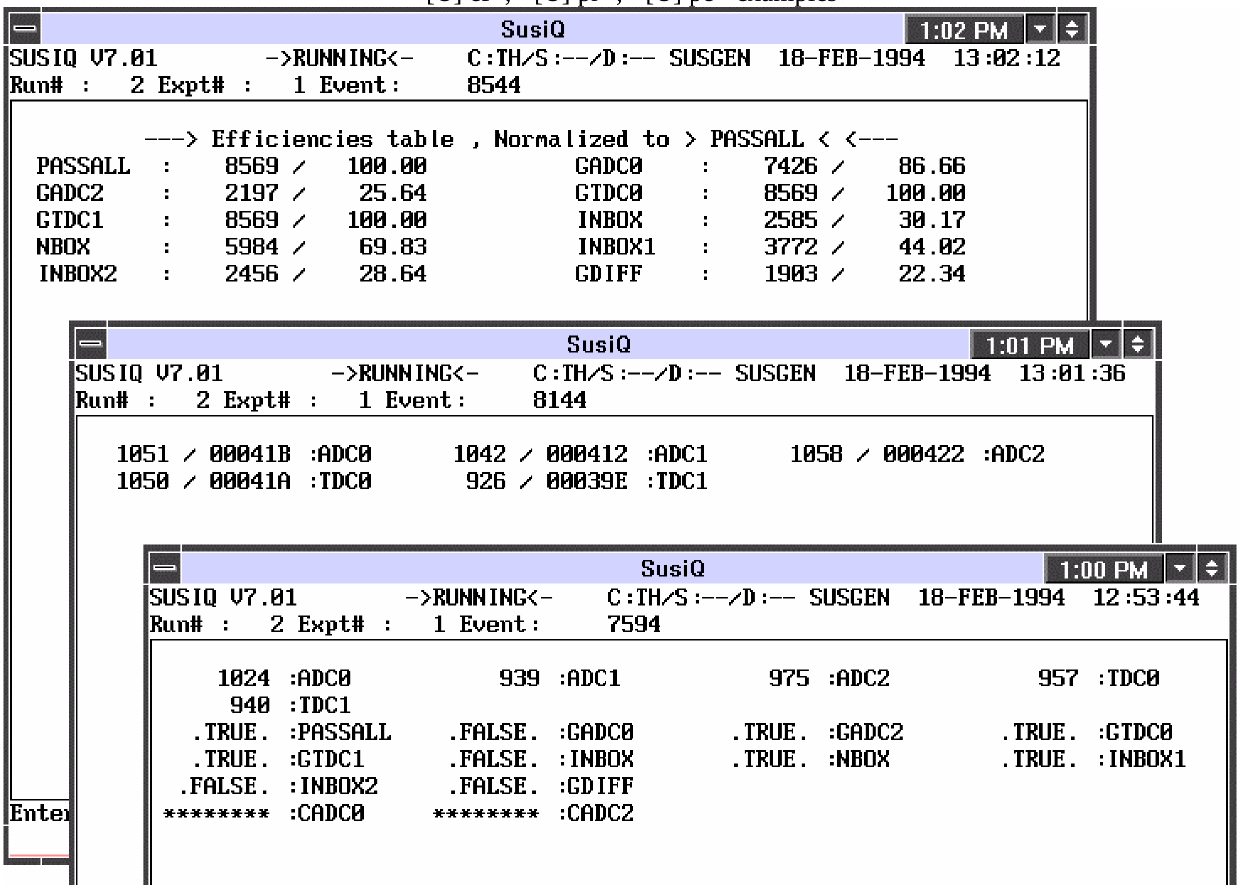
[U]... for Update

Main command for displaying live results. This key is the most used during acquisition. It allows you to display histograms, efficiencies, raw data, computed data, and dotplots online. This means that you still acquire data while you browse through the different display options. The EFficiency table gives you statistics on test conditions with a ratio relative to a predefined normaliszation condition test. The PRevent table contains all the raw data values displayed in decimal and hexadecimal format. The PEeper table gives you the raw data similar to PRevent but in addition you get the computed quantities and the conditional test results on a event by event basis.

Update table

EFficencies	update_frq	normalization test	
HIstogram	histo name/#	update_frq	
PRevent	update_frq		
PEeper	update_frq		
DOtplot	* : dotplot listing -# : remove dotplot # # : display dotplot # X_var : new dotplot followed by: X_low,X_high,Y_var,Ylow,Y_high,test name	update_frq	projection mode
STatus	update_frq		

<[U] ef>, <[U] pr>, <[U] pe> examples



Examples Update

→[U] ef<cr> inbox<cr> <cr>
efficiencies table normalized to inbox with default update frequency
→[U] hi<cr> 2<cr><cr>
display histogram 2 with default update frequency
→[U] hi<cr> time1<cr><cr>
display histogram time1 with default update frequency
→[U] do<cr>adc0<cr>0<cr>2000<cr>adc1<cr>0<cr>2000<cr>inbox<cr><cr><cr>
dotplot of adc0(0,2000) vs adc1(0,2000) gated by inbox with default update frequency and default projection mode

[X]... for eXit

This command will terminate SusiQ in a proper way and return to DOS. If a run is in progress, you will be prompted to close the run before quitting SusiQ. This procedure will insure that the EOR is done properly and the requested save_Set closed. This will also guarantee that the new dotplot and zoom settings defined in this session are saved to disk for subsequent restoration. This last statement is true only if the *susiq.dft* setup file contains a valid DOTPLOT and ZOOM definition.

[Z]... for Zoom

This option allows you to save display limit settings on any of the pre-defined histograms. The zoom settings are not histogram dependent although the listing informs you which histogram the zoom has been recorded from. A maximum of 10 zoom settings is available. The main menu after invoking the zoom command is composed of :

[Z] table

#	#	Select zoom setting # Display the current histogram using the X limit setting from that zoom setting
-#	-#	Remove zoom setting # Renumber all remaining zoom setting
S	Save	Save present setting (used in conjunction with DRA) If in Dra you did a FX then it will be saved as new zoom setting
B	Book	Book manually a setting Request Xlow,Xhigh limits as zoom setting
R	Record	Record a setting with the mouse Display the full current histogram and request 2 limits setting with the mouse as new setting
O	Overall	Select default histogram limits Deactivate any zoom setting
C	Clear	Erase all Zoom settings

Example Zoom

→[U] HI<cr> 1<cr><cr>
Set histogram number 1 display with default update frequency
→[Z] 3<cr><cr>
Select zoom setting number 3, display
→[Z] -2<cr>5<cr><cr>
Remove setting 2, renumber settings, select setting 5, display
→[Z] b<cr>30.4<cr>567.23<cr>6<cr>
Book manually a zoom setting , selects it , display
→[Z] r<cr>7<cr>
Record new settings with mouse by selecting either low or high limits on the histogram display by clicking the left button and a high or low limit and clicking again on the left button, selects it, display.

Frequently Asked Questions

🗎 I went to \sq\basic then tried SusiQ but I don't get the [M] [N] ...
✍ Before running SusiQ you should make sure that the environment variables are set correctly. You forgot to run **LOADSQ device**: first.
🗎 I try to run **loadsq c:** but it says that my path is too long.
✍ DOS limits the path definition to 128 characters ... sorry... you should have a path named **sqpath** which is the extension that **loadsq** trys to add.
🗎 I don't get the scale label on the histogram and dotplot.
✍ Check the path to the *modern.fon* file by doing **[S] st<cr>** the line graphic font should inform you on the problem. To correct the problem, you have to edit the *susiq.dft* file to give a valid path to the **FONT** parameter or comment it out. The default is **dev:\sq**.
🗎 While I'm watching a dotplot, the dotplot title changes colour?
✍ If I tell you that is on purpose, you wouldn't believe me?...
🗎 I started SusiQ and the first page being the help page does not fit on the screen?
✍ Sounds like you not even running with an EGA display... Usually the graphics part will be fine anyway. The file *susiq.dft* has a couple of resolution modes to play with. Try them... (**VIDEO_MODE**)
🗎 After a successful SusiQ installation, the efficiency table does not contain all my test declarations.
✍ Even SusiQ has some limitations. The number of tests is around 100, but the screen in a single page can handle only around 40 results. I left the 100 in case this becomes a real problem, but don't forget that SusiQ is for "simple tests."

Here is a list of some of the potential problems that you may encounter if you want to get in trouble.

✞ Do not play with flag settings if you specified **TIMGEN** or **TIMRND** in *susiq.dft*.
➢ Switching of the acquisition mode flags while you have setup a **TIMMODE** in *susiq.dft* may cause SusiQ to restart in the wrong acquisition mode. Change the setting in *susiQ.dft* and restart SusiQ.

- For monochrome monitors, it is important to setup the right colour list in *susiq.dft* which in that case should be limited to 2. An example for black/white setting is provided for the **FGCOLOR, BGCOLOR** commands in *susiq.DFT*.