INS--884

JP9110121

INS-Rep.-884
July 1991

INSTITUTE FOR NUCLEAR STUDY
UNIVERSITY OF TOKYO
Tanashi, Tokyo 188
Japan

A Data Acquisition System based on a Personal Computer

K. Omata, Y. Fujita, N. Yoshikawa, M. Sekiguchi and Y. Shida
Institute for Nuclear Study, University of Tokyo
3-2-1 Midori-cho Tanashi, Tokyo Japan, 188

Presented at the 7th Conference REAL TIME '91 on Computer Applications in Nuclear, Particle and Plasma Physics, Julich, Germany, June 25-28, 1991 and submitted to the IEEE Transactions on Nuclear Science.

INS-Rep.-884
July 1991

A Data Acquisition System based on a Personal Computer

K. Omata, Y. Fujita, N. Yoshikawa, M. Sekiguchi and Y. Shida
Institute for Nuclear Study, University of Tokyo
3-2-1 Midori-cho Tanashi, Tokyo Japan, 188

Abstract

A versatile and flexible data acquisition system KODAQ(Kakuken Online Data AcQuisition system) has been developed. The system runs with CAMAC and a most popular Japanese personal computer, PC9801(NEC), similar to the IBM PC/AT. The system is designed to set up easily a data acquisition system for various kinds of nuclear-physics experiments.

1 Introduction

A data acquisition system named KODAQ (Kakuken Online Data AcQuisition system) has been developed. It was originally designed for the iron-free β-ray spectrometer at our institute for measurements of β-ray spectra with two single-wire position-sensitive proportional chambers in the experiment of e^++heavy-nucleus interactions[1].

The system works on a personal computer, PC9801 (NEC), which is most popular in Japan and is similar to but not compatible with the IBM PC/AT. Two versions of the system have been written so far, one with the CAMAC and the other with a parallel input-output interface board inserted directly to one of the PC's extension slots. Except for the data input, these two versions are the same in data handling. In this paper we report on the one with the CAMAC, as it is more flexible than the other, though more expensive.

In the design of the KODAQ system, the highest priority has been placed upon its flexibility and versatility. We envisage that it should be so easy for anyone in need of a new data aquisition system to set up his own without special knowledge on computer language. For this purpose we have made a new data handling language (KODAQ commands). A user can customize easily the KODAQ system to his own needs and tastes, using these KODAQ commands. The KODAQ interprets and executes the command sequences as typed-in.

2 Overview

The KODAQ system works under the operating system of MS-DOS. Fig. 1 shows the program flow. The configuration of the KODAQ system is normally defined in two files: one is a text file INITKDQ.KDQ consisting of KODAQ commands (see Fig. 2) and the other is PROCAMAC.C written in the C language.

The INITKDQ.KDQ is called and executed at the system start and defines all environmental conditions such as:

- How many and what kind of display windows are to be opened at which part of the CRT display?
- How many areas with what memory size are to be reserved for data storage?
- Which data are to be displayed in these windows?

An example of the INITKDQ.KDQ is shown in Fig. 2, where two windows are defined for histogramming purpose and the corresponding memory areas. After initialization the program enters into an idle state, monitoring the mouse status and keyboard hit. If the system is set in data-acquisition-enabled state, all data displayed on the defined windows are updated periodically and the preset counts/time are checked every second. In any state of the system, a function-key hit or a command string terminated by <cr> code from the keyboard is analyzed and executed. The other file written in the C language contains CAMAC data-read and monitoring routines which must be compiled and linked to the main program as a part of the interrupt handler in advance. For easy composition of the routines, some macros and subroutines are available. Fig. 3 shows an example, in which a data is read from one of the CAMAC modules and is stored in a memory area for histogram. As shown in the Fig. 3, data-read part and monitoring part are separated. Both procedures should be given for each event type. The monitor part is optional.

A flow diagram of the interrupt-handling procedure is shown in Fig. 1a. In data-acquisition-enabled state, the interrupt-handling routine is activated as a foreground task

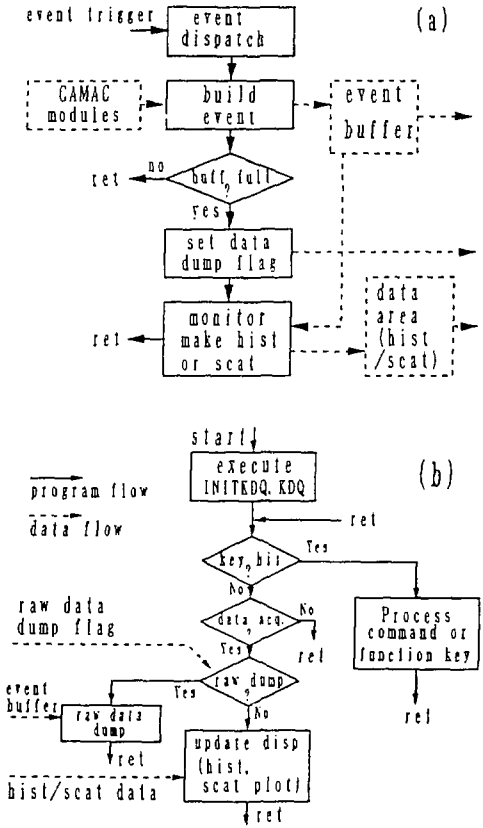

Figure 1: A (a)foreground/(b)background program flow of the KODAQ system.

```
!define display area(top,left,bottom,right)
WORLD    0,15                639,383
!define window(name,xdiv,ydiv,xloc,yloc)
DFNW  wnd1    1     2     1     1
DFNW  wnd2    1     2     1     2
!window attribute
!    (name,hist/scat,x0, xmax, y0, ymax)
ATRW  wnd1   hist     0  2047   0   -1
ATRW  wnd2   hist     0  2047   0   -1
!define memory region(name,size,comment)
DFNM     mem1     2048       E1
DFNM     mem2     2048       E2
!display memory data to window(memory,window)
DSPD     mem1           wnd1
DSPD     mem2           wnd2
```

Figure 2: An example of the system setup file(INITKDQ.KDQ).

```
/* procamac.c */
#include "camac1.c"
  case 1:     /* event 1 */
    camac16(16,15,2,bp++); /* read & reset */
    break;
#include "camac2.c"
  int e1;
  case 1:     /* monitoring event 1 */
    e1 = *bp++;
    hist(e1,memo1);
    break;
#include "camac3.c"
```

Figure 3: An example of the data read and monitor routines(PROCAMAC.C).

each time an event-trigger signal is sent to the CPU.

The event trigger is issued by the interrupt module in the CAMAC crate. Eleven event types can be distinguished by the system. One trigger may contain more than one event at a time. In the interrupt-handling routines, therefore, the event types are first scanned. For each valid event type, the data are read from the corresponding sections of the module as specified in the PROCAMAC.C. The data are packed event by event into one of the event buffers. If the event buffer is full, the interrupt handler sets the flag for the buffer and calls a monitor routine.

In the monitor routine, each event data is read from the event buffer and according to the event type it is added to the histogram data, pair data for scatter plot or others, etc. in the assigned memory area in the manner as specified in PROCAMAC.C. An example of the KODAQ display is shown in Fig. 4, where the INITKDQ.KDQ and the PROCAMAC.C of the examples in Fig. 2 and Fig. 3 are used. The buffer-full flag is checked in background job. If raw-data-dump mode is enabled the data in the full event buffer is dumped to the assigned device as the raw data file and the flag is cleared.

3 Hardware

The hardware setup of the KODAQ is shown in Fig. 5. The interrupt module and output module are indispensable. File devices, fixed disk, MO(magnetic optical) disk, and RAM disk are optional for raw-data dump. CAMAC crates are extensible up to four crates, using the crate controller CC/7000[2] in each crate. All the CPU models of the personal computer PC9801, 8086/V30 to 80386, with clock frequency up to 33 MHz, have been found to run the KODAQ system without any problems. The interrupt

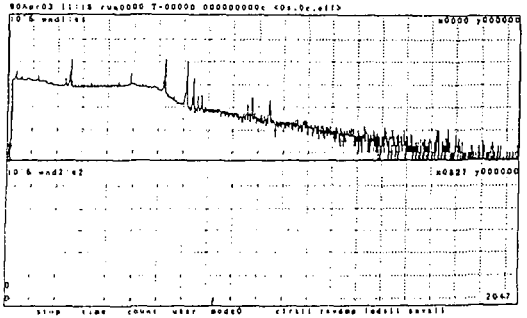

Figure 4: An example of the KODAQ display.

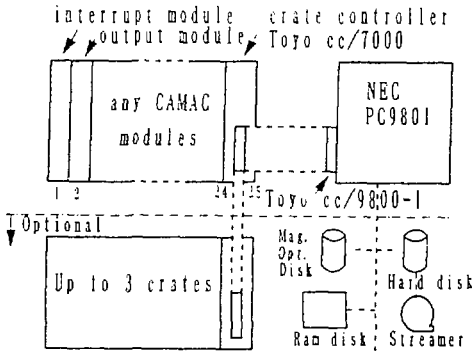

Figure 5: Minimum hardware requirement of the KODAQ.

module at station #1 is used to realize the multi-event triggering. Only the input signals to the interrupt module generate the LAM(Look At Me) signal and make interrupt to the system. The LAM signals from all other modules are inhibited. The output module at station #2 is used to send out a computer busy signal. These two modules are the minimum required for the standard KODAQ system.

3.1 Event type

Table 1: Event types of the KODAQ system.

event type	comment
1 to 11	hardware triggered event(0 $\cdots$ 10th bit)
12	start event(created at START)
13	stop event(created at STOP)
14	start date, time, comment etc.
15	stop date, time, comment etc.

The system can handle up to 15 event types. But some of them are reserved for special use by the system as shown in Table 1. If a 16-bit interrupt module is used, the user can use the lower 11 bits for event triggering. Event type 12 and 13 are assigned for software interrupt and these events are generated when the START/STOP command is issued. If any additional operations on CAMAC modules are necessary at START/STOP, the corresponding CAMAC access macros are to be included with a C statement CASE 12/ 13. The procedures are executed at START/STOP. Data of event type 14 and 15 can't be generated by the user. These events are created automatically by the system and contain the usual information at START/STOP as listed in the table..

3.2 Data read and monitor

As data-read and monitor operation are directly coupled to the interrupt handler, the procedures for these operations should be written in the C language to get fast response. Macros and subroutines are prepared to help the users to write these routines in the C language. The data-read routines are

crate(cn) : defines crate device # 0 to 3 for use in the following routines,

camacc(n,a,f) : issues a CAMAC command to the assigned module,

camac16(n,a,f,bp) : reads a 16-bit data and packs it to an event buffer,

camac24(n,a,f,bp) : reads a 24-bit data and packs it to an event buffer,

and the monitoring routines are

hist(data,memory#) : makes histogram data to an assigned memory area,

scat(x,y,memory#) : packs an x-y pair of data to an assigned memory buffer for scatter plot.

Monitor routines not supported by these macros and subroutines should be written by the users themselves. It is often necessary to select those events that satisfy some conditions. In the KODAQ system, one can set markers in the display window of a histogram or a scatter plot by using a mouse or cursor keys, and a pair or a group of markers may be used to designate a region of interest. Some macros are available to allow the users' routines to handle the markers of KODAQ system in histograms or scatter planes.

3.3 Response time

Dependence of the system response time on CPU model and clock frequency has been measured as shown in Table 2. The interrupt response time doesn't change smoothly with clock frequency. Clearly, there is a limit set by some other hardware specifications than those of the CPU. A similar situation showed up in the CAMAC

Table 2: Response time of the KODAQ system.

PC9801 CPU model	—80386—			—8086—	
Clock freq.(MHz)	33	20	8	10	8
Interrupt overhead(μs)	50	50	85	170	210
Single module access(μs)	16	16	24	60	76

Table 3: Response time of the VME-CAMAC system.

function	response(μs)
Interrupt overhead	250
Single module access	27

– test conditions –

VME: 68020(20MHz)/OS-9

VME-CAMAC interface: CES8216 + Kinetic 3922

Interrupt processing: wait by fSsleep and send wakeup flag by fSsend

module access time. These results suggest the interface determines the limit at the higher clock speed. The interface was developed six years ago when personal computer had no today's art. So, we need a new interface with better specification matched to the updated CPU technology.

The response time of the VME-CAMAC system as measured with KEK's device handler[3] is listed in Table 3. These data show that the PC-CAMAC system is not inferior to the VME-CAMAC. In practice, the speed of the system is usually limited by the output devices when raw-data dump is used.

4 System Operation

The system is controlled with the KODAQ commands which can be executed from the keyboard at any time. Some commands used most frequently are assigned to the function keys of the PC. The mouse is optional, and with the display cursor the user can select one of the displayed windows and change the parameters on the active window, i.e. the display mode, the scale of the x-y axis, the marker positions, etc. It may also be used to click one of the function key templates, thereby executing the associated command.

4.1 KODAQ commands

All the functions of the system are prepared as KODAQ commands. All the commands are interpreted and executed any time. For example, number or size of display windows can be changed dynamically. But it is recommended to do so in the state of data aquisition disabled. Sometimes it competes with the interrupt handling and causes serious damage to the system. Typical commands are listed in Table 4.

Table 4: Typical KODAQ commands.

Command	Function
WORLD	defines the size of display area on the screen.
DFNW	defines a window with a logical name and specifies its size and location in the WORLD.
ATRW	specifies the attribute of a window, hist/scat, and its vertical and horizontal scales.
DFNM	allocates a memory area for data storage and gives a logical name to it.
DSPD	associates an allocated memory area to a pre-defined window.
START	enables interrupt from CAMAC and starts data acquisition.
STOP	disables interrupt and stops data acquisition.
GCOPY	gets a hardcopy of the graphic display.
PRINT	gets numerical printout of the data.
SAVE	saves data in the memory to file(s).
LOAD	loads data from file(s).
RAWDUMP	enables raw-data-dump.

4.2 Command input

Any string terminated by the <cr> code is considered as a command. If it is one of the KODAQ command, the corresponding operation is executed. If not, the system searches for a file with an extension '.KDQ'. A file with this extension is a command file and contains KODAQ commands only. In case the first character of the entered string is '$', the string except the first character is executed as an MS-DOS command. If the first character is '!', it is a null command, doing nothing. The system accepts also a multi-command, a sequence of commands separated by ';'(semicolon) and terminated by <cr> code.

4.3 Command alias

One can define a new command using DEFINE command as follows:

DEFINE newcommand:=command,

where command is a KODAQ command. Using this function, a complex command sequence can be given a simple name. By this means, the user can customize the KODAQ system to his needs.

4.4 Special file name

The system uses the following files for special purposes. The user has to take care to avoid the same file name or command name.

INITKDQ.KDQ is used as the default file name called at the system start, as described previously. MARKERS.KDQ is the default file name created by one of the KODAQ commands, SAVEMARKER. This file contains

the marker definitions and their status saved when the SAVEMARKER is executed. If the command MARKERS is defined in the INITKDQ.KDQ, the system calls back automatically all the markers at the system initialization.

MACROS.KDQ is a text file containing the KODAQ commands extended by the DEFINE command. The user can keep his own convenient commands in this file. For example, he can use DOS commands with their original names if a statement like

DEFINE DIR:=$DIR

is written in the file, redefining the KODAQ command $DIR as DIR.

4.5 Raw-data play back

The system has raw-data play back as a standard function for off-line analyses. The command ANAFILE takes the name of the raw-data file as the argument. In this case, the START command starts off-line analysis of the data in the raw-data file. In off-line analysis, monitoring will be done by using the procedures as described in the PROCAMAC.C. Although interrupt is disabled, monitoring proceeds when an input buffer is full with the raw-data read from the file. Upon detecting the EOF of the data file, STOP command is executed, terminating off-line analysis.

4.6 Pseudo-interrupt command

The system has two pseudo-interrupt commands for the program flow control on the background job. One is a clock-time interrupt command ONTIME. It is used to perform some operations expressed by a KODAQ command a certain seconds after the data acquisition started. For example, when the command

ONTIME 500 [STOP;GCOPY;CLEARALL;START

is issued, the system stops data taking, gets a hardcopy of the display, clears all the memory area and restarts data taking, after 500 sec from the start. Because of the START command at the end, this series of operations will be repeated until data taking is stopped manually or an ONTIME command with different parameters is given.

Another pseudo-interrupt command is ONSTOP command. It defines what to do when data acquisition is stopped for any reason, by ONTIME command or manually with the function key.

Conclusion

Because the system has a feature of versatility and flexibility, it has been used at various sites and in several different kinds of nuclear experiments; measurements of position distribution of neutralized particles ejected from the electron-cooling device at TARN II[4], measurements of light-ion spectra from the proton-induced nuclear reaction with a set of counter telescope consisting of four transmission-type surface-barrier solid-state detectors, as examples. It has proved to be very useful for these kinds of experiments.

For a small scale experiment, the system should be adequate in processing capacity and has a good cost performance. We think the system may operate on IBM PC with little or minor changes, because it is very similar to the personal computer PC9801.

Acknowledgements

The KODAQ system has been continuously improved to incorporate various functions suggested by many users, to whom we would like to express our hearty thanks.

References

[1] M.Sakai et al.,Submitted to Phys. Rev. C.

[2] Toyo Technica. Co., User's manual of CC/7000. Tokyo, Japan

[3] Y.Yasu et al., KEK internal 90-2(1990)

[4] T.Tanabe et al.,Proc. of 2nd European Particle Accelerator Conference, Nice, 1990, pp. 54-56.