Annales Univ. Sci. Budapest., Sect. Comp. 17 (1998) 193-199

REAL-TIME COMPUTER MEASUREMENT CONTROL UNDER DOS-WINDOWS OPERATING SYSTEM

Z. Illés and K. Havancsák (Budapest, Hungary)

Abstract. The present paper shows what are the main facilities of designing a real time physical measurement control on personal computer.

Data acquisition and measurement control system has been accomplished for ion beaming control on the cyclotron U-400, at Joint Institute for Nuclear Research, in Dubna. A 16 channel ADC/DAC was constructed in KFKI, Budapest. This measuring unit is divided into two parts. The first one is connected to the cyclotron sensors, giving signals to the other part, connected to our PC at the control room. In principle this problem can be solved under DOS or under higher level operating environment e.g. Windows. The program under DOS will be a simple clock-based application, using the real time clock of DOS.

The measurement control program was written under Windows in object oriented language VISUAL C++. This gave us the latest methodology and environment in the software development. The main problem in this case is how to provide the nessesary real time character for ion beam control under a typical message based multi-tasking operating system.

1. Introduction

Recently in physical laboratories the usual analog measuring units have often been changed for a personal computer (PC) driven control system. An up-to-date PC has enough capacity to control not only a simply process, but a complex physical measuring system, too. On PC-s under DOS, the operating system gives a clear way to build a real-time control system. Recently not only in offices, but in the laboratories, too, the users would like to use applications under Windows operating system, because of its general 'easy to

use' capabilities. In Windows, as usually in multi-tasking operating systems, the tasks communicate with one another by normal system messages. Therefore the real-time control under Windows has some specialities. The main speciality is that we have to divide our real-time task into two parts. The first one is a virtualized system driver, which gives the necessary real time service, and communicate with the other part through the normal message queue.

1.1. Elements of a computer control system

A generalized overview of computer control system is shown in Figure 1.

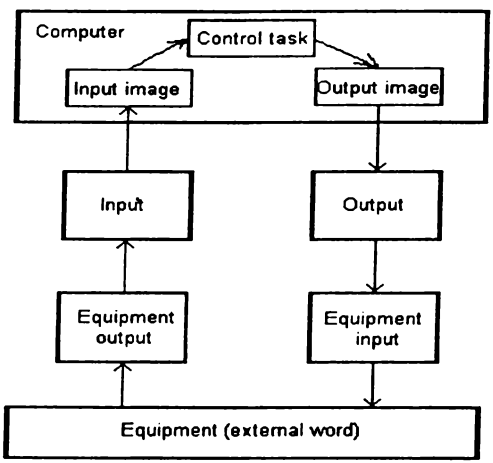

Figure 1.

The interface part between the computer and controlled equipment provides the connections between the process (in the external world) and the controller task. The input devices plus the input software provide the information to create an input image of external process, that is gives the parameters to the controller task to create the proper answer command set (output image) to the equipment.

1.2. Real-time system - a definition

The term 'real-time' will be used to refer to systems in which [1]:

a) the order of computation is determined by the passage of time or by external events,

b) the computation must be completed within a specified maximum time interval at every occasion of event or passage of time interval.

Two examples for the first category real time system are an automatic bank teller or an electrical annealing furnace which can hold a given temperature for a time period. In the first case an example is real time, it is driven by external events, placing a card into the machine indicates the transaction. An electronic annealing furnace is also a real time one, because every given temperature must hold for a given time period, and this passage of time indicates the temperature changing.

A second example for category 2, Fast Fourier transformation (FFT) system. At designing the control algorithm we have to choose a sampling interval. The computer as an FFT unit shows the measured signals. We might say, the processing rate must be minimum 18 Hz (the interval between datas 1000/18 ms,˜55ms), because our eye can distinguishes 18 pictures per second. This then requires that every 55 ms the input value should be read and the controller must send the answer to the output, and this process must not be longer than 55ms.

In order to ensure the real-time character of the system, in the case of an external event it activates the service task immediately, and only after finishing it the system can continue the original task.

We can define three types of the real-time computer - equipment relation.

i. Clock based system

The relation is defined by the passage of time, or an actual point of time.

ii. Event or sensor based system

The relations is defined in terms of the external events.

iii. Interactive systems

In the third case the relation is defined most loosely. The main requirement is that the operations in the computer should be completed within a predetermined time. The majority of communication tasks fall into this category.

Of course, these categories are often overlapped. As an example a bank teller is an interactive system, because the system must answer within a certain time period, and so after each command. But placing a card into the machine indicates an external event, starting the interactive communication. A clear event driven system does not communicate as an interactive system, this system by determining a received event formalizes an automatic answer. (Turning on the switch "a", as an event, generates the central heating system to turn on, as an answer.)

1.3. Data acquisition and control

A data acquisition and control system has been constructed to serve ion beam diagnostics at the U-400 cyclotron of Joint Institute for Nuclear Research, Dubna, Russia [3]. This system is controlled by a 486 AT personal computer. The distance between the computer and the measuring equipment is approximately 50 metres. This is the reason that between the computer and cyclotron interface there is a special master-slave serial connection form used. This overview is shown in Figure 2.

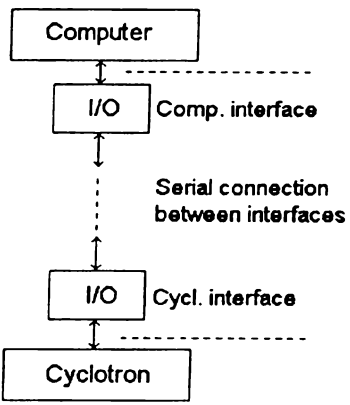

Figure 2.

The system includes 16 analog/digital (A/D) channels, and 16 digital communication channels. It measures the current and distribution of ion beam. The ion beam distribution is measured by a multi-foil 9 channel detector. We have to provide the regular irradiation of the sample with a pregiven current of the ion beam while the sample achieves the needed dose. The dose is given by the following relationship:

$$D = \frac{1}{T} \int_0^T I(t)dt,$$

where D is the actual dose, $I(t)$ is the measured current, t is the time, T is the measured time interval.

The A/D conversion is ready in every 20 ms, so we must serve it periodically in passage of time 20 ms (clock based system). The data signal can be considered as an external event also (event based system). So usually in a real problem the all of the previous categories (i, ii, iii) are mixed.

2. Elements of software engineering on PC

On PC the first problem is under which operating system the controller program should be constructed. We can distinguish three types of programming [2]:

- sequential,
- multi-tasking,
- real-time.

The speciality of a real time program is that the sequence of its actions is not determined by the designer but by the environment, and the environment task (one usually uses real-time clock task) cannot be delayed.

Under DOS operating system we can easily create a real time control program. We have to design the event driven task(s), and remap the needed event handler. In this case, when occuring an external event (usually an interrupt request) the sequential process control is stopped, and is activated the interrupt service task [4].

In multi-tasking operating environment the real time controls do not communicate by the usual synchronization way (message queue, shared variable,..), because the environment task cannot be delayed. Therefore in a multi-tasking operating system the real time control has two parts, an environment task (event driven) and a 'main' task, and they communicate through the usual way. If a multi tasking operating system has built-in facilities to serve external event handling, then this system is often called 'real-time multi tasking operating system' (Sorix op. system, real-time unix, Siemens AG). The costs of these systems are much more higher than a normal most often used Windows configuration.

At the ion beam control program we need exactly measure the progress of time, and the ion beam current. Having measured data the environment task (real time service) controls whether the actual current is in a specified interval. If the current has a 'suitable' value, then the task should pass the measured data to the main process, and stops his own work. In the opposite case the task must break immediately the ion stream, and inform the main process about the problem.

If we solve this problem under DOS, then the program uses an operating system timer interrupt facility. In this case the 1CH interrupt handler is remapped to our interrupt servise task. This task measures the current and the ion beam distribution data, and after testing the current whether it has a 'proper' value, it puts the result into the shared variables. Through these variables it informs the main process about the results. The timer interrupt signal occurs in every 55 ms. Therefore when we design our program, we must

not forget that our interrupt service must be finished before the next service-request occurs.

Under Windows the DOS method is not applicable, because every resource, device in the system are controlled by Windows Kernel and they communicate with the applications through the message queue. In this way we cannot achieve the real time character (a, b). Therefore under Windows we have to choose another way. We make an environment part as part of system Kernel, which provide the real-time character of the control. This task controls the ion beam, as the interrupt handler above under DOS, and the results pass through the normal message queue to the main application. When occuring an external event (passage of time or data ready signal), the environment part being an additional system device services it immediately. This driver can be created under Windows as a device driver (usually called virtualized device driver (VDD)). These types of driver are used e.g. by the National Instruments Inc. in its hardware-software package. However there is another way to solve this problem. We put the environment part driver into a dynamically linked, do not movable library (DLL). This method is often useful, because in this case this library is loaded into the memory only when we use the measurement application. The scheme of realised application is shown in Figure 3.

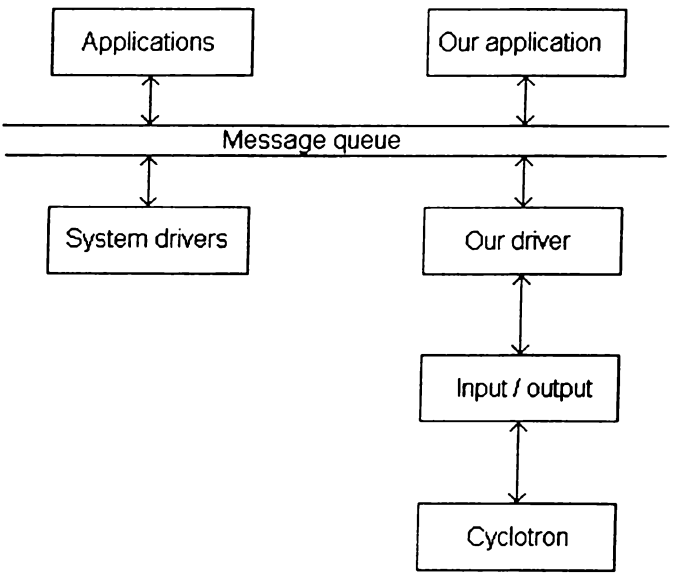

Figure 3.

3. Discussion

We chose for making the main part a complex object-oriented (OOP) development environment for Windows, the Microsoft Visual C++ ver. 1.5 (VCC) [5]. It gives the full OOP features for application designing, makes the program code shorter and more efficient. The CASE tools (AppWizard, App Studio, Class Wizard) are powerful framework system which lets us to concentrate on designing the functional code.

The DLL is written in assembly, because the VCC do not provide under Windows the real time event handling facilities. For the future interesting task would be to design the real-time event handling, as a part of system exception handling mechanism.

References

[1] **Bennett S.,** *Real-time Computer Control,* Prentice Hall, 1988.

[2] **Pyle I.C.,** Methods for the design of control software, *Software for Computer Control. Second IFAC/IFIP Symposium on Software for Computer Control,* Prague, 1979.

[3] **Illés Z., Sándor, A., Illés Á., Havancsák K. and Szenes Gy.,** *Data Acquition and Control of the Experiment Employing C-64 Microcomputer for Ion Beam Diagnostics,* P10-89-674, JINR, Dubna, 1989.

[4] **Milesz S., Molnár J., Illés Z. and Sándor A.,** Determination of Ion Mobilities by IBM PC Controlled Nuclear Electromigration Method, *J. Radional. Nucl. Chem. Lett.,* **135** (4) (1989), 231-236.

[5] Microsoft Visual C++, Microsoft Foundation Class User's Guide, Reference Guide 1.5, 1994.

Z. Illés
Dept. of General Computer Science
Eötvös Loránd University
VIII. Múzeum 6-8.
H-1088 Budapest, Hungary

K. Havancsák
Inst. for Solid State Physics
Eötvös Loránd University
VIII. Múzeum krt. 6-8.
H-1088 Budapest, Hungary