USB Based Image Acquisition System for Real-Time Applications

Mihai R. Dumitrean*, Daniel Moga*, Gabriel Lazar*, Florin Hurgoi*, Mihai Munteanu*, Gabriel Calin Cret**

******Technical University of Cluj-Napoca, Romania,*
***University of Oradea*

E-mails: mihai.dumitrean@tedelco.ro, daniel.moga@aut.utcluj.ro, gabriel.lazar@com.utcluj.ro, h.florin@email.ro, mihai.munteanu@mas.utcluj.ro, gcret@uoradea.ro

Abstract: This paper presents some design ideas that should guide the designer toward the specification of an USB based image acquisition system. The techniques for interfacing multiple USB cameras are examined and solutions suitable for real-time applications are presented. An implementation for a two camera system based on these ideas is described.

Keywords: Image Acquisition, Real-time, USB, multiple cameras system

1. Introduction

Only one decade ago, designing and implementing an image acquisition system for a real-time application required high-tech hardware and software solutions. These solutions were expensive and not widely available.

The main drawback was that implementing a real-time acquisition system required the use of an expensive acquisition board, since the CPU power at that time barely covered the processing power required by such an application.

But the use of an acquisition board often meant that the application designer was forced to use a compatible camera for that board. The higher the needed quality of the captured images, the more expensive was the final solution.

Today, the world of image acquisition systems is completely changed: the cost of the hardware has dropped to that limit that building an image acquisition system is no longer restricted to high-budgeted projects.

In this paper we present an USB based low-cost image acquisition system together with technical details related both to the hardware and software implementation.

2. The hardware

A common solution in the past was to use a CCD or CMOS camera together with an acquisition board which led to complex and expensive implementation

solutions. Today this is no longer the case, since the offer of USB connected CMOS or CCD cameras is very diverse. In the market one can find, from the common web-cams to thermal cameras, with price ranging from few tens to thousands of dollars.

From the variety of the camera types available on the market it is possible to choose the solution that suits best the specific application: B/W or color cameras, with CMOS or CCD sensor. In this way the cost of the final product can be lowered.

The USB cameras open new implementation possibilities because, to nowadays PC, it is possible to connect several cameras. In this way, with appropriate operating system and device drivers it is rather simple to implement a multiple camera system.

There are two ways to support multiple USB cameras connected to the same PC. One is to make use of multiple USB cameras each with its own device driver. Then the application's software must provide a mean to control the devices, especially from the resource usage point of view. Also the device driver must allow the OS to use all the connected cameras.

Another way to support multiple USB cameras is to use a device driver that can handle multiple cameras. In this way the control over the resources is made inside the device driver and the application's software does not need to deal with cameras' resources control.

3. The software

Once the cameras are connected to the PC, and the OS has identified and installed the cameras properly, the task left to the designer is to decide the API that will be used to interface the cameras to the actual application.

In Windows operating systems, one common API for image acquisition applications is the Video for Windows (VfW). Although it is rather old, VfW is supported by most of the device drivers. For the new device drivers that are based on the (Windows Driver Model) WDM**,** in Windows 98, Me and Windows 2000, Microsoft provides a VfW-to-WDM mapper. In this way an application developed using the VfW API, and that used hardware with VfW device drivers, can be easily interfaced with hardware that uses WDM device drivers.

A real-time application that uses an image acquisition system has, generally, the following main modules:

- the acquisition module: which is responsible of interfacing the application with the image acquisition system
- the processing module: which is responsible of applying application specific operations on the acquired images.
- the control module: which is responsible of issuing application specific commands, using the results from the previous module.

An important aspect of the real-time application is that the operations from the different component modules must be synchronized and made in real-time (that means that data buffering is acceptable only for a well defined and application specific time).

Having these constraints, the interface between the acquisition and the processing module must be very careful designed.

The first problem that appears is the synchronization between the two modules. There are two time intervals that are of interest: the first one is the time difference between to successive frames (the inverse of the frame rate), and the second one is the

time needed for the processing module to complete its task. In order to keep the modules synchronized, some restrictions must be applied. In order to not have unprocessed frames, a restriction must be imposed on the processing time, and that is that the processing time must be less than the time interval between two successive frames. This restriction can be relaxed only by increasing the time interval between frames, that means either lowering the frame rate or by allowing that part of the frames are dropped, only each n-th frame being processed.

Using this approach, the design of the application require multiple capture threads, one for each camera, and one thread for processing. The capture threads are running free, the acquisition data is stored in a thread safe manner in an input buffer, and the processing thread read the data stored from that buffer. It is worth to mention the following problem encountered with this approach: if the cameras are not synchronized, then the captured frames from the multiple connected cameras may not have the same time stamp. Because of this the information from the cameras cannot be used to accurately describe the same scene.

For an application that requires the description of a scene using multiple cameras, a more suitable approach is to synchronize the acquisition module with the processing module. This can be done by controlling the acquisition. In this case a separate control thread in required, thread that will be responsible for controlling both the processing thread and the capture threads, and in this way the synchronization, both between the acquisition module and the processing module, and, inside the acquisition module (between the cameras), can be achieved.

The controlling thread (see figure 1) commands (using the VfW API) when the capture should begin and, when the capture completes (the captured frames from all the cameras are ready for processing), triggers the start of the processing thread. After the processing is completed a new capture command can be issued to the cameras. In this way, the frame rate at which the images are acquired is given by the time needed to process the captured frames.

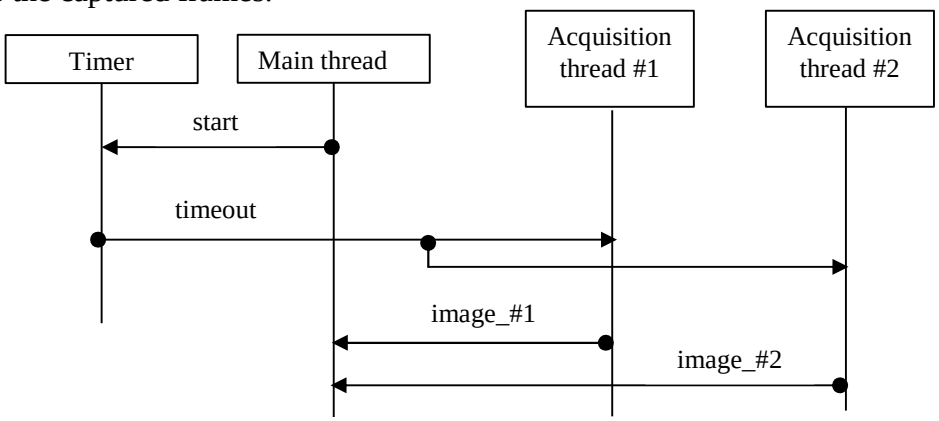

Figure 1. Threads setup

4. Results

Next, based on the facts presented so far, we will describe an USB based image acquisition system for real-time application. The acquisition module is based on 2 USB CCD cameras; the processing module will implement basic operations on the acquired images. The control module is not described in this paper.

The format of the acquired images is: 320x240 pixels, 8bit B/W.

Each of the cameras has an associated preview window, a video settings panel and uses a separate thread for acquisition. The control of the acquisition and of the processing is done in the application's main thread.

After the capture is completed, the captured frames can be viewed one at a time. The steps presented in the next paragraph describe the application flow.

1. Start.
2. Set the priority of the main thread, create and start the acquisition threads.
3. Query de VfW compatible device list and populate the pop-down lists.

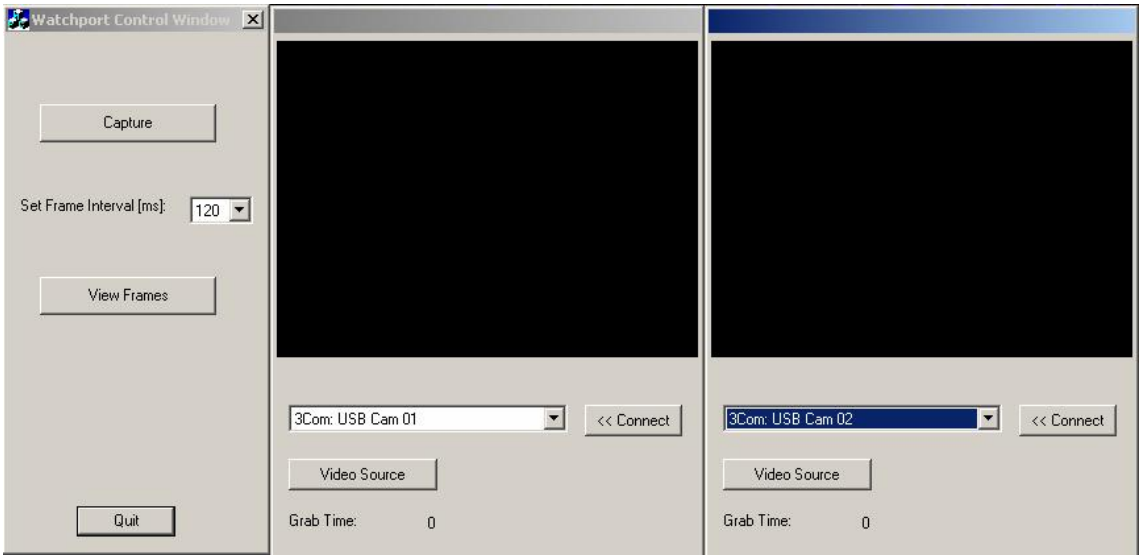

Figure 2.Select and connect the devices

4. Connect the device to the preview window and set the parameters of the capture (dimension, color, etc.). (using the Connect button)

5. Set the time interval between consecutive frames in the same thread. (using the Set Frame Interval pop-down list)

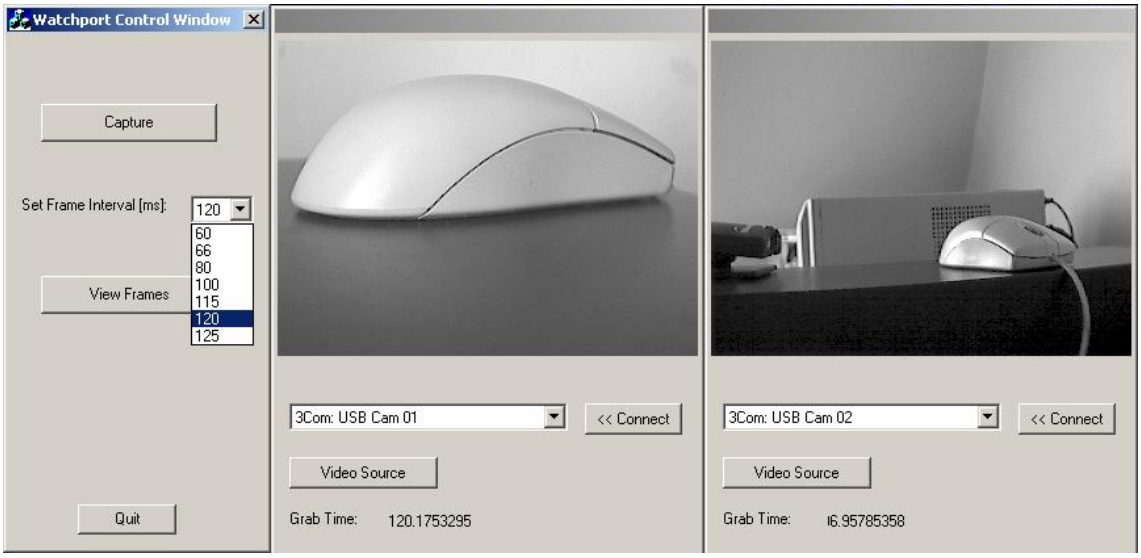

Figure 3. Set the frame interval

6. Start/Stop the capture. (using the Capture button)

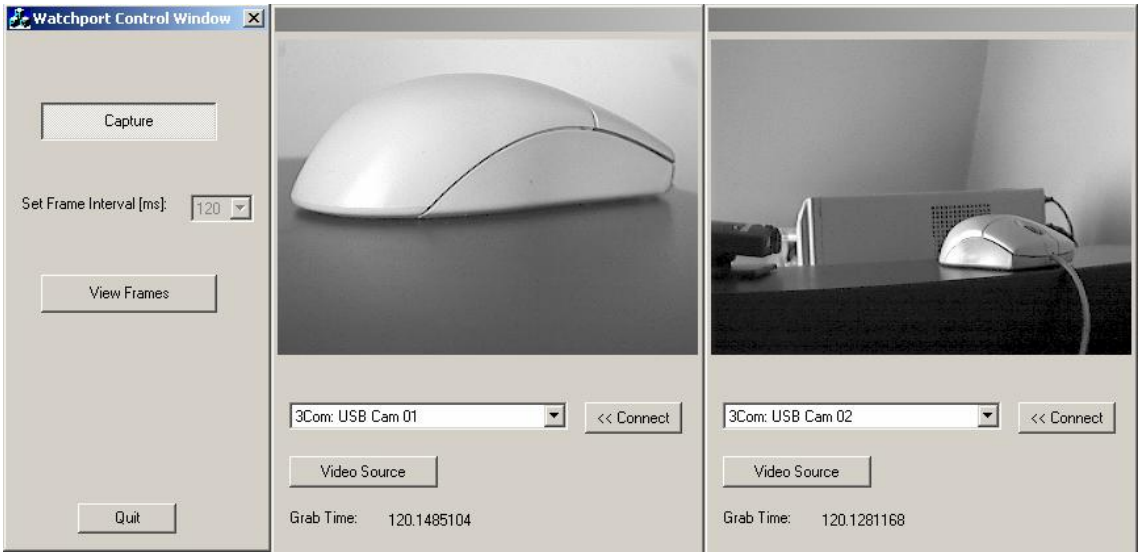

Figure 4. Start/Stop the capture

7. View the captured frames.

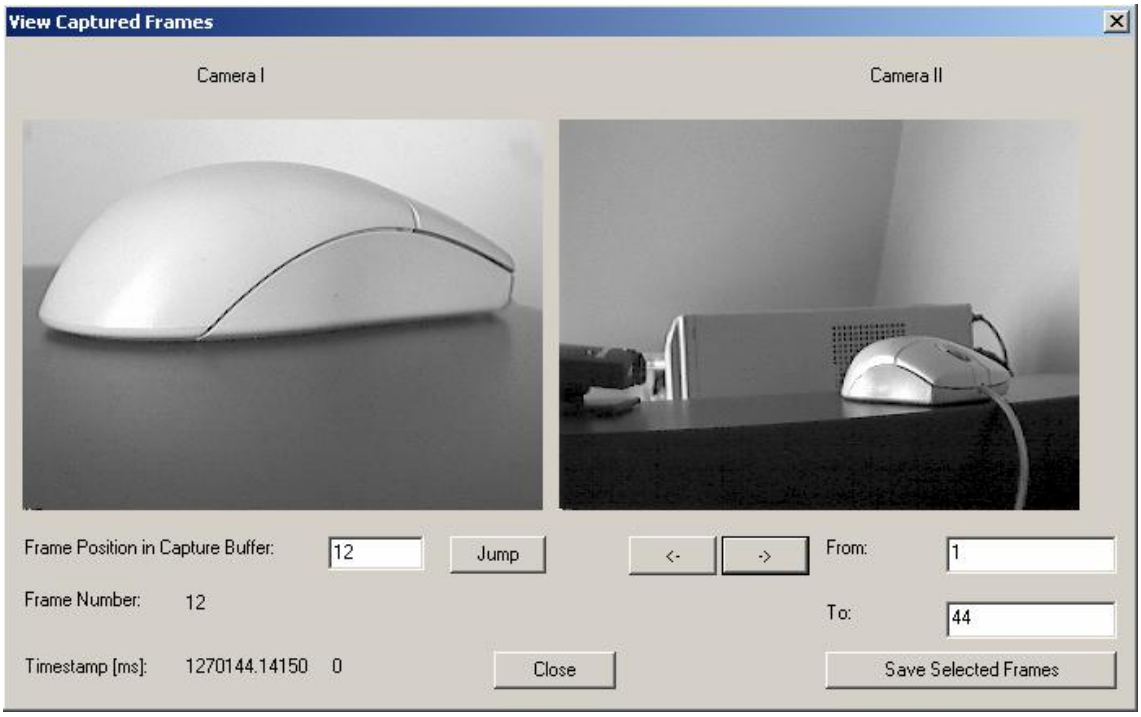

Figure 5. View the captured frames

5. Conclusions

Although Windows does not have a real-time architecture, some real-time control applications based on image acquisition and processing can be developed with quite good performance. On the PC platforms the availability of the USB interfaces can ease the implementation of the acquisition system due to the availability of the USB cameras. The

performance of a multiple cameras application depends very much on choosing a suitable architecture. The architecture demonstrated with the application described could be easily followed or extended with the condition of using "smart enough" drivers for the cameras.

6. References

[1] Microsoft Corporation, 1998, VfW-to-WDM Video Capture Mapper on Windows 98 and Windows 2000, *Device Drivers Technical Article*, MSDN.
[2] Microsoft Corporation, 2002, How Hardware Devices Participate in the Filter Graph, *Microsoft DirectShow for Windows XP Service Pack 1*, MSDN.
[3] Microsoft Corporation, 2002, Video for Windows, *Platform SDK: Windows Multimedia*, MSDN.
[4] Bruce Powell Douglass, 2003, Real-Time Design Patterns, Addison Wesley, 500 pages.
[5] Sam Dekey, Making Windows NT a Real-Time Solution–A Technical Overview, National Instruments, http://zone.ni.com/devzone/conceptd.nsf/webmain/0176ACDE201D5B7986256A63007A310E/$File/WP1639.pdf