Computer Science Journal of Moldova, vol.**7**, no.1(19), 1999

Data acquisition with direct memory access

Iu. Guzun

Abstract

This paper is concerned about using Direct Memory Access (DMA) transfer for real-time data acquisitions from a plug-in data acquisition board Aquarius DSP-1. Since the conversion speed of the modern Analog to Digital Converters is increasevealy, the using of the interrupts for reading of the conversion results becomes impossible. For the data acquisition system, presented in this paper, this speed is up to 100 kS/s. This problem can be successfully solved using DMA technique. We subsequently describe the hardware of the data acquisition board, which has been used to acquire data. Next, have been presented the functions, in C programming language, for programming a data acquisition via DMA. Finally, we present some results obtained using this kind of data acquisitions, and some conclusions.

Keywords and Phrases: real-time data acquisition systems, driver software development and DMA transfer mode.

1 Introduction

Current PCs are high-performance computers that can handle the most demanding data acquisition and control applications. Plug-in data acquisition boards and instrument interfaces are available now for all the popular platforms - PCs, PS/2, Macintosh, Sun Workstations. In this context in a DSP Labs of "Politehnica" University of Timisoara has been designed and implemented a data acquisition board - Aquarius DSP-1. Our goal is to implement some driver software and a Graphical User's Interface which can be used for data acquisitions from plug-in board outlined above. So far have been implemented drivers for programmed I/O and data acquisition via interrupts.

2 Data Acquisition Board Aquarius DSP-1

2.1 An overview of the data acquisition board Aquarius DSP-1

Data acquisition system Aquarius DSP-1 is a plug-in board that can be used in a PC computers with a 16-bit ISA-AT bus connector. This board has been implemented for studying of the electrical machines.

The hardware of the data acquisition board has been described in [MGG97]. Here will be introduced only some aspects of hardware organization for data path using DMA. First of all, board uses DMA controller of the host computer, channels 5 - 7. Secondly, the number of samples that will be acquired should be loaded in channel 1 of the timer from data acquisition board. When the acquisition is completed, and the counter 1 of the timer becomes equal to zero, an interrupt is generated. Interrupt routine moves the stored data from the DMA buffer in other part of memory or writes them to the disk. Finally, the "Single Transfer Mode" of the DMA controller was used.

In the following paragraph only the address map and registers configuration of data acquisition board will be described.

The data acquisition board (DAQ) acquires data in two modes:

via interrupts and using DMA, executing continuous acquisition of a single channel, or multichannel acquisition with continuos scanning.

Driver for data acquisition using interrupts has been presented in [GG98].

2.2 Address map of a data acquisition board

A decoder of port addresses can address the programmable elements of the board. Each port address can be obtained by adding an offset address from address map to the board's base address (100h, 120h, or 140h jumper selectable), noted by BAdd in address map. The address map of a data acquisition board is shown in Table 2.1.

Notice that the I/O address of data acquisition board is not selected by decoding the address on the address bus []. During the DMA transfer, the address bus contains the memory address and cannot contain

the I/O port address.

Table 2.1 Address map of a data acquisition board AQUARIUS-DSP

	Read/Write	Base Address [BAdd] +
Command Register	W	+ 00h
Acquisition Results + Status Register	R	+ 04h
Start A/D Conversion in Software Mode	W	+08h
Timer i8253	R/W	+10h
Programmable Paralel Interface i8255	R/W	+14h
Data Load for D/A Conversion	W	+18h

2.3 Data Acquisition Driver Software for data acquisition using DMA

2.3.1 Data transfers via DMA

DMA means Direct Memory Access. Lets consider the problem of moving a large amount of data to or from an I/O device. This requirement is commonly encountered in the operation of real-time data acquisition systems that are constantly moving large amount of data into memory. An obvious way of making the transfer would be a short program which for an input operation might read a byte (word) from the I/O port into the accumulator (AX register). of the 8088 processor, and then move data from the AX register to a memory location to store it. In addition, we have to keep track of the memory locations where the data is going. By far the simplest way of handling this is to lay the data down in continuous blocks locations within a block of memory, using one of the processor's index registers to control the address. Each time a byte is transfered, the index register (usually the DI or Destination Index register) is incremented or decremented to point to the next location. A typical example assembly language program to do this might be as follows:

```
SETUP:
    MOV      AX, SEGMENT   ; setup segment of memory transfer
    MOV      DS, AX
    MOV      DI, OFFSET     ; setup start address within segment
    MOV      CX, COUNT      ; setup # of bytes
    MOV      DX, IOPORT     ; DX = I/O port address
READ:
    IN        AL, DX          ; read byte from I/O port        (8)
    MOV     [DI], al         ; store data                     (10)
    INC      DI               ; increment index                (2)
    LOOP    READ             ; continue till CX = 0           (17)
CONT:        .....               ; yes, continue with program
```

The numbers in parentheses following the READ: label are the number if the processor clock cycles required to execute the entire read segment. On a standard IBM PC, the clock runs at 4.77 MHz, corresponding to a clock cycle period of 210 nanoseconds, and the loop takes 37 cycles or 7.8 microseconds to execute. This would be the fastest we could transfer each byte. Note also the following:

1. The processor is tied up 100% of the time in transferring data; it cannot execute any other part of the program while the transfer is underway.

2. The rate at wich the data is transferred is controlled by the processor clock and may not correspond to the rate at wich the I/O device wants to handle the data, This can be circumvented by polling the I/O device to see if it is ready, or having the I/O device generate a hardware interrupt; but either od thise adds further code to the routine which will slows down the transfer rate even further. In practice because of all the sacking of registers that occurs when a hardware interrupt is processed, it is impossible to handle data transfers rates much above 5 to 10 Khz using intrerrupts.

3. If the processor has to handle an intrerrupt from one device while it is involved in handling a data transfer to another device then the delays involved may cause it to miss data, or at least will cause the discontinuitly in the data flow.

It would be nice to have a way of transferring data without involving the processor so it can be freed up as much as possible to attend the transfer rate up and be able to control the rate easily. Since all we want to do is move byte directly to/from an I/O port from/to memory without any sort of instead proving special hardware that will accomplish this commonly required task. The process of connecting an I/O device directly to memory is known as Direct Memory Access (DMA), and the hardware that controls this process is known as the DMA controller, which the case of the IBM PC is the function of the 8237 chip on the system board.

DMA normally occurs between an I/O device and memory without the use of the microprocessor. A DMA read transfers data from the memory to the I/O device. A DMA write transfers data from I/O device to the memory. In our case DMA write has been used.

In order to perfom DMA operations, the peripheral must include hardware that generates the DREQ (DMA request) and responds to the DACK (DMA acknowledge). The DMA contrloller, on the other hand, is a system component that is a standard feature of the IBM PC archtecture. It is important to appreciate that the DMA controller sets the dinamics od the DMA transfer, that there is nothing in the peripheral I/O device that can alter the maximum speed at which the controller will handle data.

To do a DMA transfer it is necessary to know a few things:

1. The address of memory to access
2. The length of data to read/write

This can all be put into a structure:
typedef struct dmainfo
{
char Page;
unsigned Offset,
Length;
}dmainfo;

The Page is the most significant 4 bits from the 20-bit address of the memory location. Note that DMA transfers cannot across 64k page boundaries.

The Length is actually Length-1; sending in a 0 will move one byte, sending a 0xFFFF will move 64k.

2.3.2 Programming DMA channels 5 through 7 of the IBM PC system's DMA controller

To do a successful DMA transfer it is necessary:

1. To set the Mask bit for the channel
2. To clear Byte Ptr
3. To set the DMA transfer mode
4. To set the memory ADDRESS and LENGTH
5. To set the DMA page
6. To clear DMA mask bit

A data acquisition driver includes the following functions:
char MakePage(unsigned *offset, unsigned mem_off, unsigned mem_seg);
void interrupt dma_move_samples(__CPPARGS);
void interrupt new_int60(__CPPARGS);
int dmaacq(int, int);
void dma_movedata(unsigned, unsigned, unsigned, unsigned, unsigned);
int far init_timerdmamode(unsigned port_addr, unsigned const_div,

unsigned const_div1);
char read_status_dma(char);
void set_mask(char);
void clear_mask_bit(char);

3 Results and Conclusions

In this paper the driver software for a data acquisitions using DMA has been presented.

From this project we have gained some experience. First of all, there were a lot of details to be worked out in the specification and implementation phase. After this, we did study the system's DMA controller 8237, and the techniques for using and programming of the DMA controller in real-time data acquisition system. On the other hand, the data acquisition driver using direct memory access has been implemented to allow real-time data acquisition up to conversion speed of the ADC. Using the technique of the interrupts in this case leads to a wrong interpretation of an acquired signal. As a solution, an on-board FIFO memory can be used. This kind of memory will be included in a next version of data acquisition board.

In the immediate future this driver will be used in industrial applications for the study of electrical machines in general and of their transient state in particular.

In table 2.2 we present some experimental results for three methods of acquisition using polling, interrupts and DMA transfer.

Table 2.2. Experimental results

Acquisition Mode	Number of Channels	Number of Samples	Sampling Rate	Average Acquisition Time (sec)
Polling	1	4 kSamples	-	00.06
Interrupts	8	4 kSamples	800 Hz	02.03
DMA	8	4 kSamples	800 Hz	00.48

Note that these experimental results have been achieved using an IBM PC computer 5x86 at 133 MHz.

References

[1] [MGG97] Mihai Micea, Iurie Guzun and Viaceslav Guzun. Performant Multifunction I/O Board for the IBM PC AT, in Proc. Int. Conf. On Microelectronics and Computer Science (ICMCS), vol. 1, (Chisinau, Moldova), pp. 167-171, October, 1997.

[2] [GG98] Viaceslav Guzun, Iurie Guzun. Data acquisition system Aquarius DSP-1, in Proc. Int. Conf. On Technical Informatics (CONTI'98), vol. 2, (Timisoara, Romania), pp. 275- 282, October, 1998.

[3] [MS96] Mark Socos. ISA Technical Connector, 1996, http://www.gl.umbc.edu/~msocos1.

Iurie Guzun,
"Politehnica" University of Timişoara
Department of Computer Science and Engineering
Bd. V.Parvan 2, RO–1900, Timişoara,
România
Phone/Fax: +40-56-192049
e-mail: guzun@cs.utt.ro

Received April 2, 1999