Bekkeng, 24.6.2013

Exam solutions FYS3240/4240 June 2013

Problem 1

a) Explain how memories are used when data are transferred from a DAQ-card and into the DAQ software program. The DAQ-card is assumed to be configured for continuous sampling.

See Figure 1. Data samples from the DAQ-card ADC are added to the **FIFO buffer on the DAQ-card**, and then transferred to the **PC buffer** (using DMA). The PC buffer is implemented in computer **RAM** as a **circular buffer (ring buffer),** due to continuous sampling configuration. (In LabVIEW the DAQmx driver creates this circular buffer). The software DAQ-program is running in the application memory in RAM, and the program read the data samples from the PC buffer memory and into the application memory.

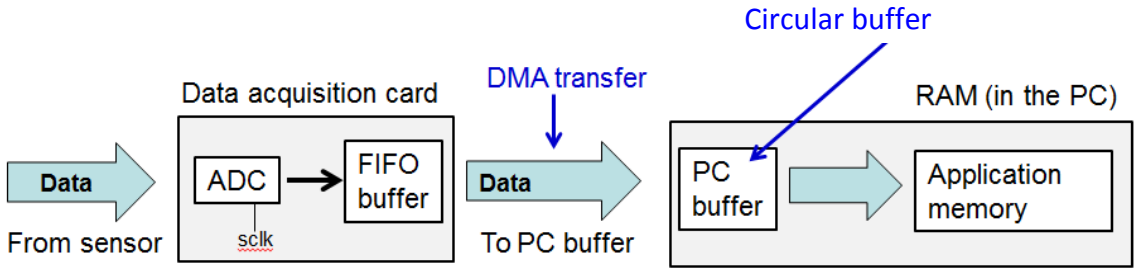


Figure 1: Memory use during DAQ.

b) Explain overwrite and overflow errors when reading from a DAQ-card.

1) An **overwrite error** indicates that information is lost and occurs when the software program does not read data from the PC buffer quickly enough. Samples that are written to the circular PC buffer are overwritten before they are read into the application memory.

2) An **overflow error** indicates that information has been lost earlier in the data acquisition process. Overflow errors indicate that the First In First Out (FIFO) memory buffer onboard the data acquisition card has reached its maximum capacity for storing acquired samples and can no longer accept new samples. An overflow error is symptomatic of a bus transfer rate that is to slow for the requested DAQ-card sample rate.

c) How can the errors in problem 1b) be avoided?

1) Use **Producer-Consumer** architecture in the DAQ software program. The producer loop then read data from the DAQ-card and adds the data to a queue (FIFO in RAM). The Consumer loop then read data out of the queue for further processing and storage. This structures makes sure that the producer loop can read the data out of the circular buffer as fast as possible, to avoid overwrite.
2) Use a **Direct Memory Access (DMA)** transfer mechanism between the DAQ-card and the computer. DMA is the fastest data transfer mechanism, and helps avoiding that the DAQ-card FIFO buffer reached its maximum capacity. (In addition, using DMA allows the CPU to work on other tasks).

d) Given that a 100 Hz sinus signal is sampled by a DAQ-card with a sample frequency of 50 kHz, and the buffer size of the DAQ-card is 1000 samples. If you plot the 1000 samples in a graph, how many periods of the 100 Hz signal do you see?

Number of periods = buffer size * signal frequency /sample frequency = 1000*100/50000 = 2

e) Explain the term oversampling, and why it is useful in data acquisition.

The Nyquist sampling theorem says that we have to sample a signal with a sample frequency greater than at least two times the maximum frequency component of the signal. The frequency $f_{sampling} = 2 * f_{signal}$ is called the Nyquist frequency. Oversampling means sampling faster than the Nyquist frequency. Advantages of oversampling are:

- Increased signal to noise ratio (SNR) for a given number of ADC bits.
- Possible to use a simple RC anti-aliasing filter before the ADC
- Better representation of the signal shape.

Problem 2

a) Explain the differences between software timing and hardware timing, and give examples of how both timing techniques can be used in a software program for data acquisition.

Hardware PC clocks

- Real Time Clock (RTC) is an integrated circuit on the motherboard.
- The RTC has a battery backup power so that it tracks the time even while the computer is turned off.
- Based on a 32.768 kHz quartz crystal oscillator.
- Maximum resolution of 1 millisecond (1 kHz).

Software PC clocks (gives the system time on the computer)

- Maintained by the operating system, based on the RTC interrupts.
- When the system starts it sets the system time to a value based on the real-time clock of the computer and then regularly updates the time based on interrupts from the RTC.

Software timing attempts to resolve milliseconds (ms) on a computer within the limitations of the operating system. In Windows the 1 kHz clock (maximum) that is available is used for software timing. Using a software timer function (e.g. a **wait** inside a loop) to control the loop rate, see Figure 2, you can expect some differences in the time interval between each iteration of the loop.

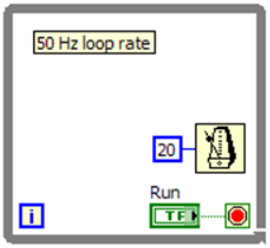


Figure 2: Illustration of Software timing using LabVIEW.

To achieve a better timing **resolution** and **accuracy (determinism)** hardware timing must be used. **Hardware timing** can give **ns** to **µs** accuracy, depending on the hardware and synchronization technology used (e.g. GPS-card or oscillator on the DAQ-card). An example of **Hardware timing** is shown in Figure 3. No wait function is used in the producer (or consumer) loop; the producer loop rate is given by the DAQ-card setup:

$$f_{producer} = \frac{\text{sample rate (Hz)}}{\text{DAQ card buffer size}}$$

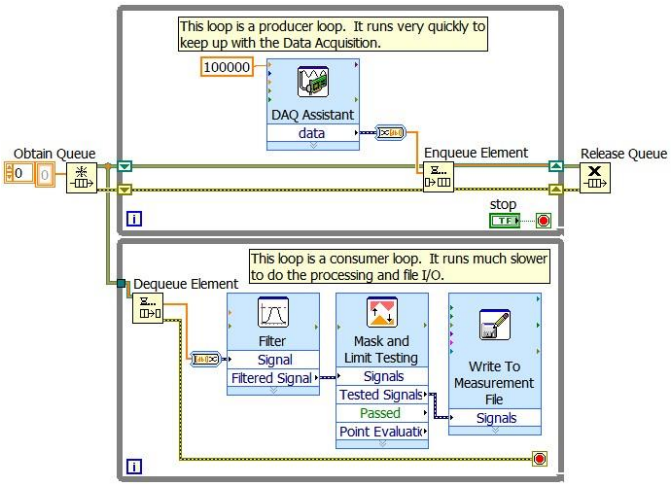


Figure 3: Producer-consumer example for illustration.

Another example of hardware timing would be to timestamp a rising/falling edge of an external signal using a timing card with a GPS receiver (or with an IRIG-B or IEEE-1588 input).

b) Given a clock with an error of 20 ppm (0.002 %), what is the accumulated clock error during one day (24 hours)?

Error = 0.002/100 * 60 * 60 * 24 = 1.73 seconds.

(Note: according to the SI-system the second is the standard for time measurements. However, specifying the error in minutes or hours is also ok, but the answer must be given related to a time unit).

c) What is the typical accuracy of the following time synchronization technologies: GPS, IEEE-1588, IRIG-B, NTP and TCP/IP?

See Figure 4 for time accuracy. GPS is the most accurate, followed by IRIG-B, then IEEE-1588, and then NTP (uses the UDP protocol). TCP/IP is the worst case. Note that IEEE-1588, NTP and TCP/IP can have varying precision due to the network connection. For instance a small LAN in the lab with a direct cable connection to an NTP server gives a simple network, while using an NTP server connected to the internet (going through a huge amount of routers, switches, etc.) is less accurate.

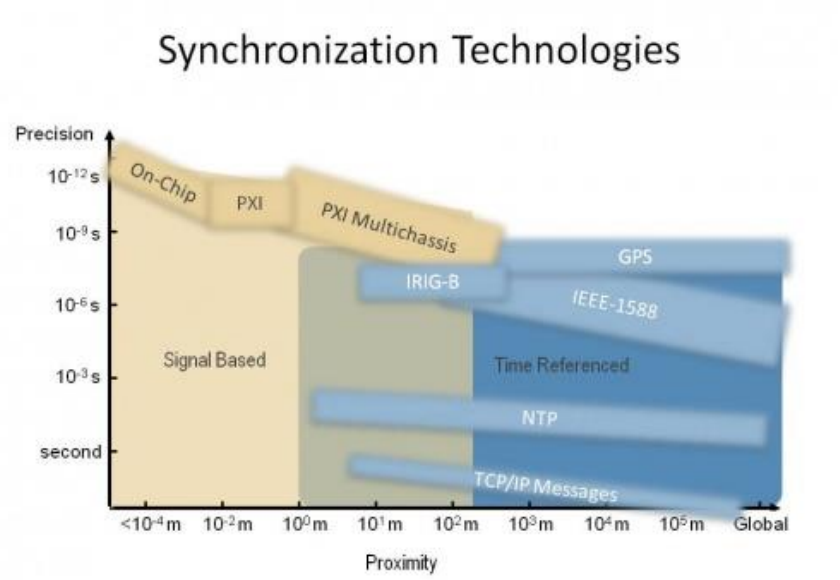


Figure 4: Time accuracy.

d) What is a leap second, and why is it used?

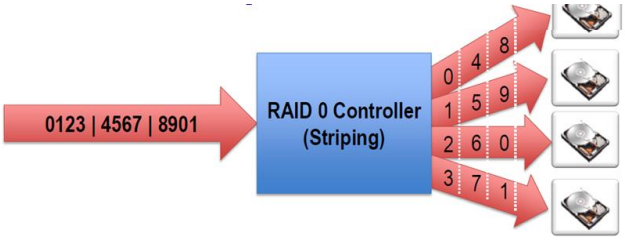
A leap second is a one-second adjustment that is occasionally applied to UTC time in order to keep UTC time of day close to the mean solar time. (Solar time is a reckoning of the passage of time based on the Sun's position in the sky). Leap seconds are necessary because the length of the mean solar day is very slowly increasing (due to slowing of the Earth's rotation), and partly because the atomic fixed-length SI second, when adopted, was already a little shorter than the current value of the second of mean solar time. (Time is now measured using stable atomic clocks, whereas the rotation of Earth is much more variable).

Problem 3

a) Explain RAID-0 and RAID-5.

RAID-0:

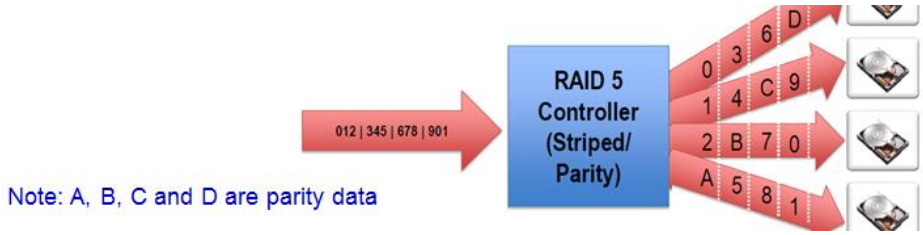
- Striping without redundancy.
- Increased write/read speed.
- Unimproved system reliability.
- The fastest RAID configuration.
- Requires at least two disk drives.



Figur 5: RAID-0

RAID-5:

- Increases write/read performance (but less than for RAID-0) and also increase safety (due to redundancy).
- Parity data distributed on all disk drives (single parity).
- Write overhead because of the additional parity data that has to be created and written to the disk array.
- Can only tolerate one drive failure (array continues to operate with one failed drive).
- As vulnerable to data loss as a RAID-0 array until the failed drive is replaced and its data rebuilt (can be a problem for arrays with large disk drives due to long rebuild time).
- Gives less space for measurement data (due to parity information).
- The minimum numbers of disks are 3.



Figur 6: RAID-5

b) Why is it a limit of about 2.2 TB in disk size for Windows XP 32 bit?

This 2.2 TB limitation dates back to the 1980s and the original IBM PC. This introduced the **master boot record (MBR)** partitioning scheme to describe hard disk partitions. (MBR consists of a sequence of 512 bytes located at the first sector of a data storage device such as a hard disk). BIOS systems with MBR disks use **32-bit values** to describe the starting offset and length of a partition. Due to this size limit, MBR allows a maximum disk size of approximately 2.2 TB due to (2^{32} -1 sectors) * 512 bytes/sector = 2.199TB

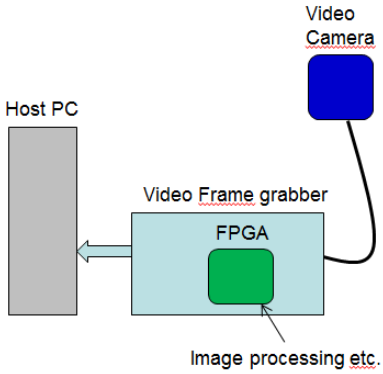
c) Explain the new standard in use to avoid the limitation in problem 3b).

GUID Partition Table (GPT) provides a more flexible mechanism for partitioning disks than the older Master Boot Record (MBR) partitioning scheme. **64-bit values** are used to describe partitions, and GPT can handle disks of up to 9.4 x 10^{21} bytes (2^64 -1 sectors * 512 bytes/sector). In addition, the Unified **Extensible Firmware Interface** (UEFI) is a replacement for the older BIOS firmware interface.

d) Explain the term "Hardware acceleration", and give two examples of how it can be used.

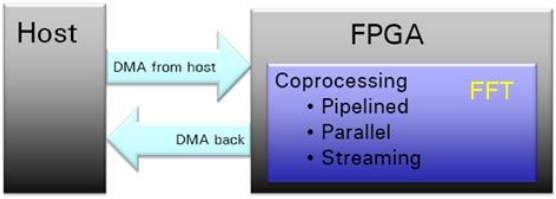
Hardware acceleration is the use of computer hardware to perform some function faster than is possible in software running on the general-purpose CPU. The main difference between hardware and software is concurrency (hardware can be true parallel), allowing hardware to be much faster than software. Hardware accelerators are designed for computationally intensive software code. Typical hardware accelerators are GPUs and FPGAs.

Example 1: Onboard processing and data reduction using FPGA; e.g. video processing (see Figure 7).



Figur 7: Example of Onboard processing.

Example 2: Co-processing; offload the CPU using FPGA (See Figure 8).



Figur 8: Example with Co-processing.

e) What is the problem(s) with the LabVIEW code in *Figure 9* ?

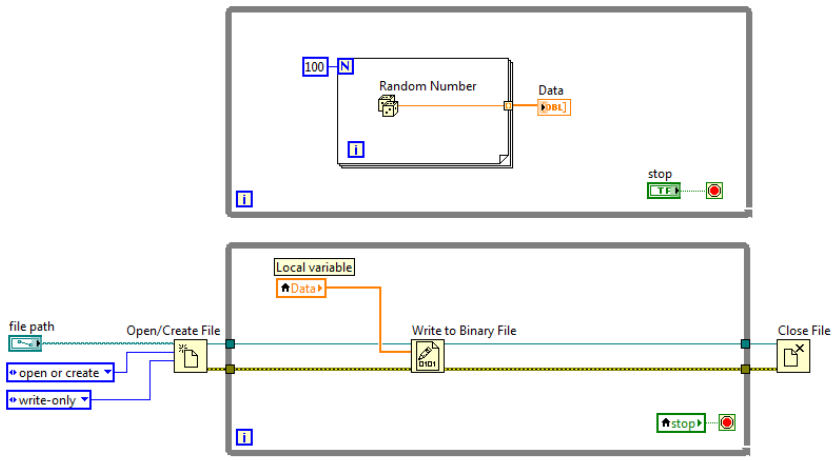


Figure 9: LabVIEW code for problem 3 e)

Main problems with the code in Figure 9:

- No **wait** functions in the loops!
- **Local variable** (for data) used instead of a queue!

Bekkeng, 24.6.2013

This is an example of a producer –consumer structure, where the producer generates random numbers (100 numbers in a 1D array) and the consumer writes the data to disk. Writing to disk is much slower than the random number generation. Therefore, the use of a local variable to transfer data between the two loops will have the effect that several of the random numbers generated in the producer will not be received by the consumer and therefore not written to disk. A data **queue** should have been used, since it gives synchronization of the loops and the queue (FIFO buffer) makes sure that no data are lost.

The problem with using a local variable is increased since no software synchronization (wait functions in both loops) is used. Without a wait function (or some other hardware synchronization) a loop will run as fast as possible and consume most of the CPU power.