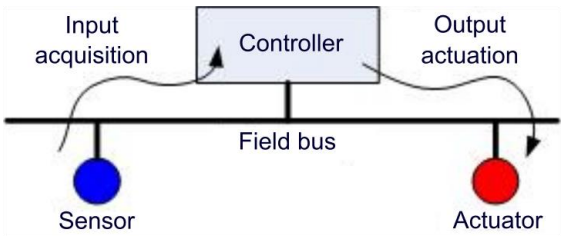
Master in Electrical and Computer Engineering

Test and Evaluation of Network Response Time

Version 1.3 (2021/11/24)

1. Introduction

In most industrial processes the input signal acquisition and the outputs actuation is performed by a controller that cyclically executes the control algorithm. When using a (fieldbus) network the delays associated with reading the sensors and changing the actuators are impacted in such a way that, if not considered, can disrupt the performance of the process controller.



A process' control rules and requirements generally impose some temporal constraints:

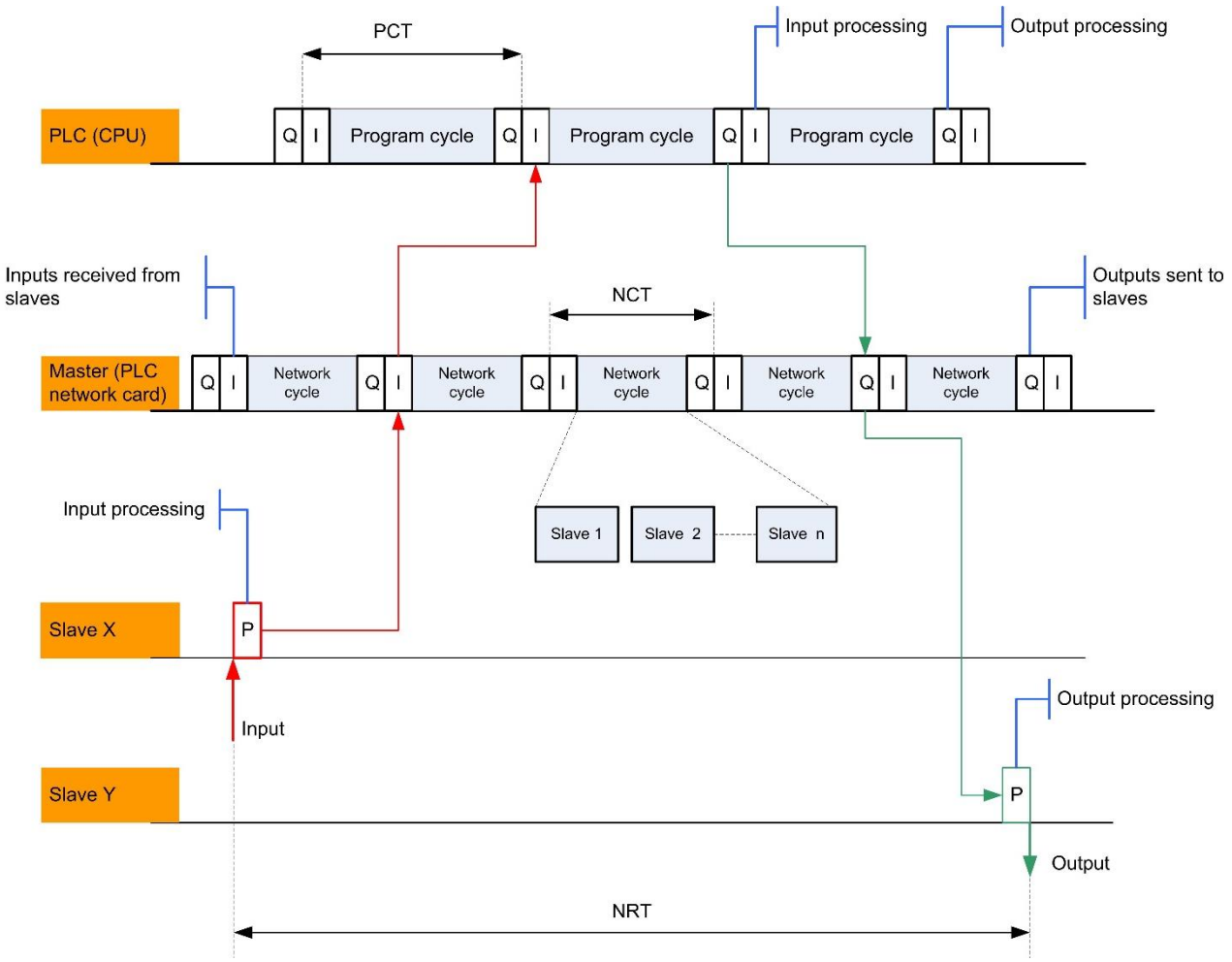
- **Discrete contro**l systems generally impose a maximum response time on event occurrence (i.e. maximum delay between sensor actuation and corresponding actuator actuation). Variations in response times to events is tolerated, as long as these are bounded (I.e. they do not exceed a certain value)..For example, we may have a requirement that: motor X must be turned off at most 10ms after the limit switch Y actuates;
- **Continuous controllers** (e.g. PID) are simpler if the interval between consecutive acquisitions (or actuations) is assumed fixed. This greatly eases design and increases control effectiveness. The variation of this interval is called jitter. When jitter reaches larger values (typically >10% of the acquisition period) it becomes detrimental to process control, particularly for the dynamic response.

Typically, the PLC (Programmable Logic Controller) has a CPU that executes the control program cyclically, and a network card (for the fieldbus/network) that executes the network protocol and taking on the responsibilities of the *network master*. The network card is responsible for transparently (to the user) performing all communication tasks with the *slaves* (remote I/O devices). Once the network is configured, the status of the inputs and outputs in the slaves become available in a shared memory area of the PLC. This memory is shared by the network card and the PLC's CPU. In reality, two process maps are created:

- the **input map** containing a copy of the values read off the slaves' inputs;
- the **output map**, with a copy of the last values sent to the slaves' outputs.

NOTE: In addition to these maps (memory areas) there are also others associated with configuration, status, error codes, etc. of each slave.

NOTE: From the point of view of the program running on the PLC's CPU, access to these memory areas is usually transparent, for example using the %Mwxx syntax. In some PLCs (such as those from Schneider) the inputs and outputs on the slaves are accessed as normal (local) inputs and outputs, using the %Ixxxx and %Qxxxx syntax.



When the PLC's network card operates in cyclic mode, the input and output maps are updated in each cycle of the fieldbus/network. The network card cyclically interrogates each of the slaves on the network, sending and receiving data. Note however that the network cycle time may have be different to the CPU's cycle time, and that these two cycles are not always synchronized with each other.

It should also be noted that the PLC's network card can only access the shared memory areas at the beginning or end of each program cycle, respectively to read and write data, In this case, the interaction is said to be synchronous. Since the program's execution cycle (in the PLC's CPU) does not necessarily coincide with the network cycle, this can lead to mismatches between the cycles – for e.g. the CPU can either read a value just recently updated by the network card, or a more stale value which will be updated by the network immediately after being read.

The following figure represents an example of the possible sequences and delays in the interaction between the PLC's CPU, the network card (acting as network master), and the network slaves. It should be noted that this behavior is not exactly the same for all networks - there are many variations. The delays in reading a remote input and writing to a remote output are as follows (times in sequential order):

- The delay between the slave physically acquiring the values and making them available to the network (i.e., storing the data in the slave's local memory);
- **Network Cycle Time** (**NCT**) - The period with which the master reads the data of all slaves;

- The time the master's card takes to make this information available to the PLC's CPU (due to synchronization between cycles);
- **Program Cycle Time** (**PCT***)* - The period of the PLC's program execution;
- The time that the PLC's CPU takes to make this information available to the fieldbus/network card (due to synchronization between cycles);
- The period of the network master sending the data to all slaves – **NCT**;
- The time that elapses between the slave receiving the data and physically making it available at its outputs.

Network Response Time (**NRT**) is defined as the time interval between a value changing at a remote input and the consequential changing of a remote output.

Note that the delay between an input value changing and the "arrival" of the new value at the PLC is variable. This variation depends on the values of the PLC's **Program Cycle Time** and the **Network Cycle Time,** as well as the phase (offset) between these two cycles. The same can be said for the propagation of an output value, from the PLC to the remote output.

The minimum **NCT** depends on several factors: namely the characteristics of the fieldbus/network protocol, the number of slaves on the network, the amount of data exchanged with each slave, and the network transmission rate. Some of these parameters are intrinsic to the type of network and cannot be modified.

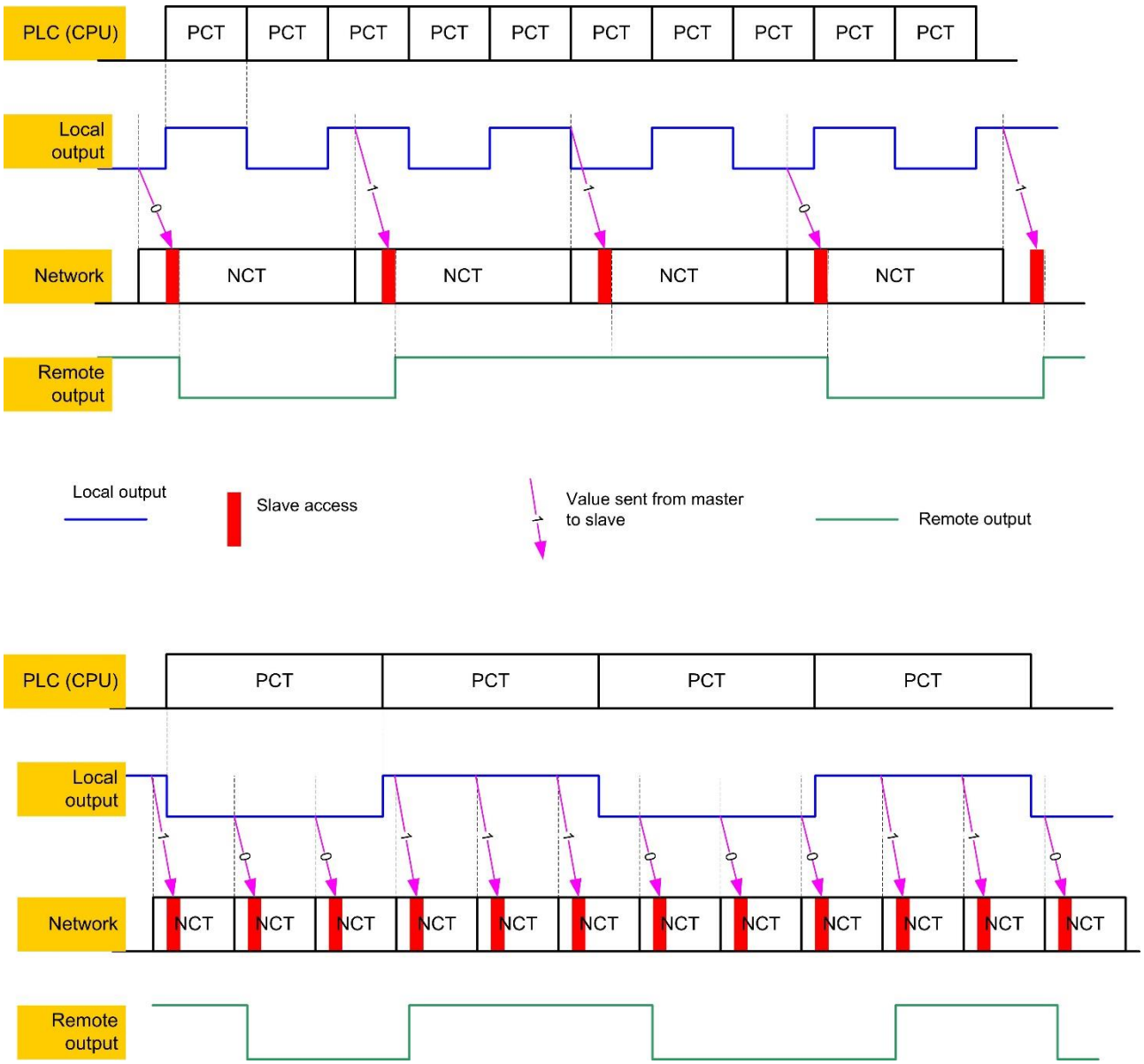
The minimum **PCT** depends on the number of I/Os read/written, the number and type of instructions executed by the program, as well as the processing speed of the CPU. The PCT may too be variable (for example, if the number of executed instructions varies between consecutive cycles due to an IF clause).

It is usually possible to configure the NCT and the PCT to be perfectly periodic, regardless of the time taken to execute the program or transmit the slaves' data. In this case, it is necessary to configure the network master and PLC to execute their cycles periodically and the period value must be larger than the maximum time needed to transmit all slaves' data or execute the PLC program, respectively.

The **NRT** is dependent on the values and phases of **PCT** and **NCT**.

2. Example

Let us consider an example wherein we wish to write to a remote output (in a slave) as fast as possible (or alternatively to read a remote input at the highest possible frequency.) We wish to analyze the maximum frequency with which we can activate a remote output, and the variations on this frequency.



One possible way to accomplish this task is to have the PLC simultaneously toggle a local and a remote output (change from 0 → 1, and 1 → 0) at the highest possible frequency. The highest frequency is achieved by executing the program in *freewheeling* mode (each cycle starts immediately after the end of the previous one) and toggling the outputs each time the program is run (i.e. at each program cycle).

NOTE: Although the program's execution time may vary, let's assume for now that it is constant (or approximately so).

In a first approach it would be logical to assume that the remote and the local outputs will behave similarly. This, however, is not always the case. For an example see the previous figure where two distinct situations are highlighted:

- PCT << NCT
- PCT >> NCT

When analyzing this figure keep in mind that the PLC reads the inputs at the beginning of each cycle and writes to the outputs at the end of each cycle. Remember too that the fieldbus master (i.e. the PLC's network card) works independently and asynchronously with the PLC's CPU.

In the first situation (PCT << NCT), the remote output's cycle is close to NCT. It also suffers from a high variation (jitter). If the NCT is much higher than PCT it is easy to conclude that the remote output's cycle will be at most very close to NCT, and that the relative jitter will decrease with an increasing value of NCT. That is, in these situations the NCT will dominate.

In the second situation (PCT >> NCT), the remote output's cycle is close to PCT. Additionally, it presents a smaller variation (jitter). If PCT is much higher than the NCT it is easy to extrapolate that the relative jitter will decrease with an increasing value of PCT, and that the remote output's cycle time will be very close to PCT. That is, in these situations PCT will dominate.

In conclusion, the relative value of PCT and NCT will influence the maximum frequency at which an output can be activated, or that an input can be acquired.

3. Experimental Validation

You will have available an Arduino based circuit that allows you to apply a signal (S) to the remote Input, as well as measure the time it takes until that signal is propagated back to a remote output (A) on the same slave, i.e., a remote I/O device. For the signal (A) to be correctly generated, the PLC must execute the following program, where Sn and An are the signals S and A received and sent through the network by the master, respectively:

An: = Sn;

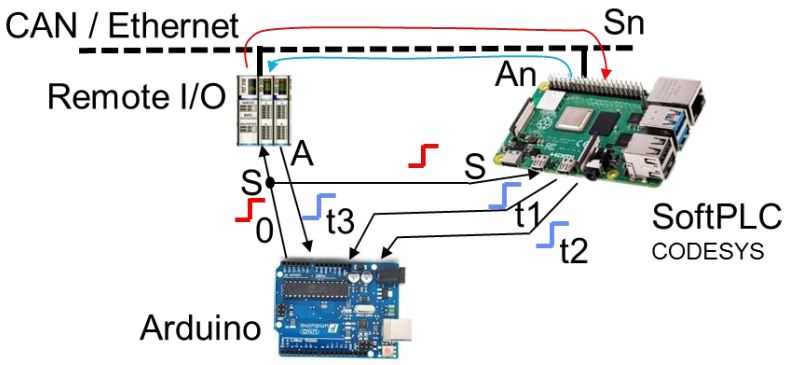
Start by writing the above program on the PLC. Due to the effects described earlier, the observed value of NRT will be variable. You should therefore proceed as follows:

Setup your experiment with known values of PCT and NCT. Use the Arduino to perform multiple measurements of NRT (at least 100 measurements for each configuration to be statistically relevant). Store the measured values in a file and perform some statistical processing, e.g., determine the distribution (trace a histogram) as well as the observed maximum and minimum values. Based on the observed extreme values and the explanations provided earlier, try to develop a model that may be used to determine the maximum and minimum NRT according to the values of NCT and PCT.

Repeat the above with multiple settings of PCT and NCT and use the observed results to confirm the models you develop. See the accompanying slides for the specific values to use.

Note that the implementation of PLC and network master cycles is highly variable across equipment producers, thus your observations will be specific to CODESYS SoftPLCs. Sometimes, as it is the case with this CODESYS SoftPLC, there is a coupling between these cycles, with the PLC cycle being used to generate the network cycle. Consequently, the actual network cycle observed in practice will be an integer multiple of the PLC cycle, whatever the value configured for the network. Other PLCs may use more steps, thus more cycles, in the transfer of information between network card and PLC memory, or simply different phases in the network cycles of different channels. Nevertheless, the concept is the same in all cases, namely a transfer of information across cyclic entities.

In order to have a better understanding of the propagation of the input signal (S) from the slave to the Master and then back to the slave (A), we added a few more connections and intermediate signals. First, we extended the input to arrive directly to the Master. Thus, the Master receives the signal S via two connections, directly (S) and through the network (Sn).



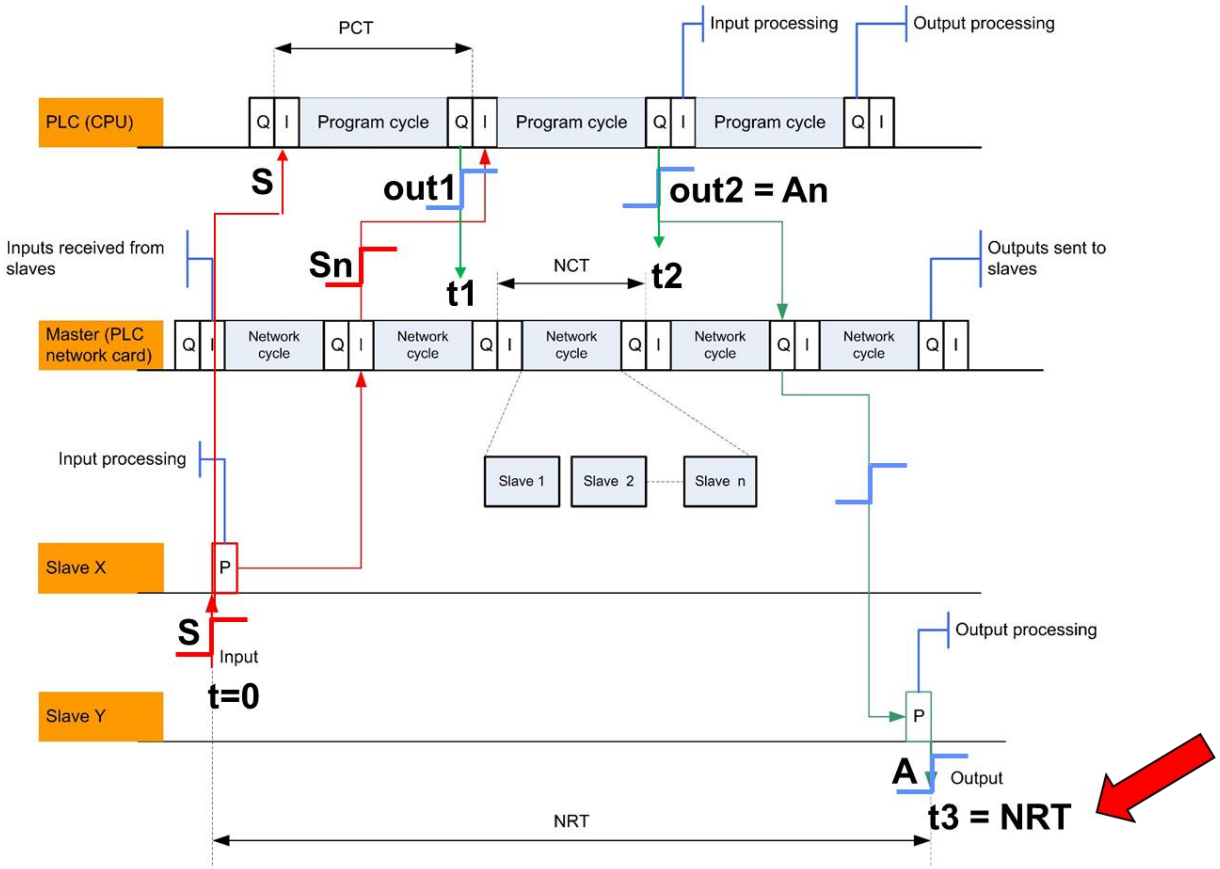
Then, we generate two new signals in the Master that are connected to the Arduino, too. These signals are generated in the cycles in which S becomes available (out1) and in which Sn becomes available (out2) to the Master program. Note that this is also the cycle in which An is generated. Thus, to generate all these signals, the Master program must now be:

```
out1 := S;

out2 := Sn;

An := Sn;
```

The Arduino timestamps these three signals and communicates the three timestamps in a row, <t1, t2, t3>, via the serial connection to the host node. The most relevant timestamp is t3 which corresponds to the NRT as explained before. The timestamp t1 is the time a cyclic program takes to read an input, the difference between t2 and t1 gives the communication delay from the slave to the PLC, and the difference between t3 and t2 gives the communication delay from the PLC to the slave.



4. Exercise

You are controlling one of the 'machines' of the cell floor simulator in the lab. This machine can move the turret (containing tools) up and down. When this turret is lowered or raised, the motor must be switched off whenever its respective limit switch is actuated. If the motor remains switched on for more than 40ms after the limit switch is actuated, there is a risk of damaging the turret.

Let us assume that the control program (Pc) executing this task runs in the PLC, and that the sensor (S) and the actuator (A) are connected to a slave (remote I/O device) connected to the PLC by a (fieldbus) network. The control program is relatively complex because in addition to controlling the turret, it controls many other aspects of the manufacturing cell. There is therefore no absolute certainty that it complies with the safety requirements, including the time requirement mentioned above.

You therefore decide to add another program to execute the interlock (Pi). This program Pi simply shuts off the motor (set output A to 1 – negated logic) whenever sensor S is set to 1.

- Using the models you developed in the previous section, determine the configuration that allows you to meet the time requirement of 40ms considering the PLC is running in freewheel.
- Instead of 40ms, consider a maximum delay of 200ms to switch of the motor. What settings allow you to meet this requirement?
- Consider that the PLC runs periodically with PCT = 20ms. Redo the previous points.

5. Project Grading

The grading of this laboratory project will be done through an individual test. This test may contain questions about the type of distribution observed in the measurements, obtained models, etc. It is particularly important to build the models referred to in the previous sections as well as their experimental validation.