FLOPSYNC-2 Configuration tool

A. Leva F. Terraneo {alberto.leva,federico.terraneo}@polimi.it

Introduction

This tool allows to easily tune the α and T parameters of the FLOPSYNC-2 controller

$$R(z) = \frac{3(1-\alpha)z^2 + 3(\alpha^2-1)z + 1 - \alpha^3}{(z-1)^2} \tag{1}$$

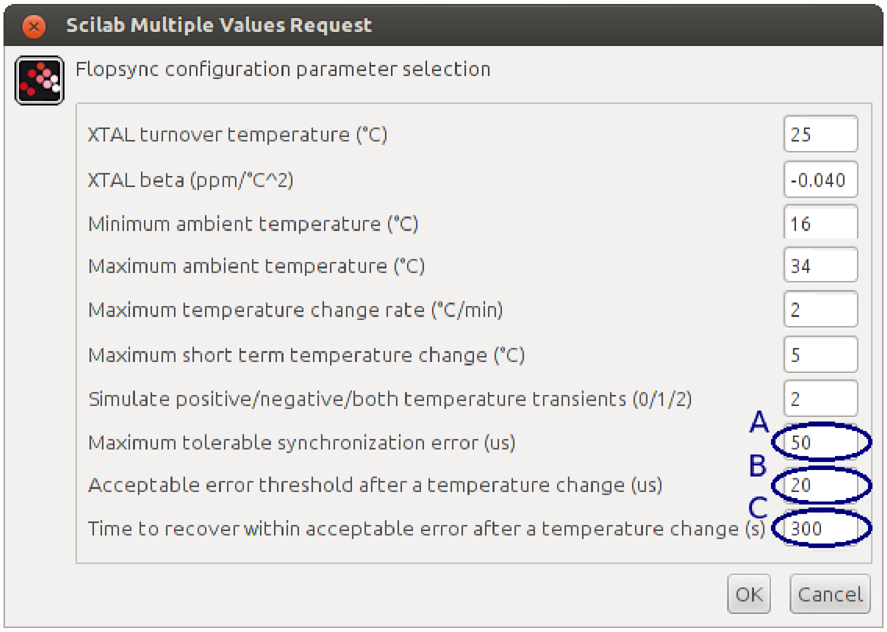
based on requirements on the absolute maximum synchronisation error that is tolerable in a given application. Note that relative requirements (i.e, the time to recover within a given percentage of the maximum error during a temperature transient) can be expressed in an even simpler manner, without depending on the disturbance model on which this tool concentrates.

Launching the tool

This tool requires Scilab to run. Once you have installed and opened it, use the *cd* command within the Scilab prompt to change the current directory to the one containing the *ParameterConfigurationTool.sci* and *param.dat* files. You also can use the *ls* and *pwd* commands. Then, simply type the command *exec('ParameterConfigurationTool.sci');*

Input data

Once started, the tool presents you a dialog where you can input the data needed to select the controller parameters.



The needed data belongs to three categories: nominal data for the quartz crystal, that can be taken from its datsheet, information on the ambient in which the sensor node is going to operate and synchronization error requirements.

The first category includes the crystal turnover temperature and β parameter. The default values are common for a wide range of 32768Hz watch crystals, including but not limited to the MC306-G-06Q-32.768 used in the stm32vldiscovery, ECX-306X and other.

The ambient information are

- Minimum ambient temperature: The minimum temperature in which the WSN node is going to operate. For example, the minimum temperature during winter for outdoor apllications.

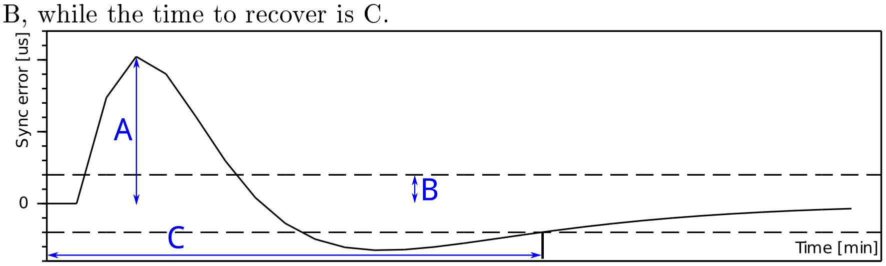
- Maximum ambient temperature: The maximum temperature in which the WSN node is going to operate. For example, the maximum temperature during summer for outdoor applications.

- Maximum temperature change rate: The maximum change rate of the *crystal* temperature. This depends on environmental conditions, usually reaching its maximum during a shade/sunlight transition where temperature changes very fast. Also, note that the enclosure in which the WSN note is placed has a smoothing effect on this value. The default value based on experiments is between 2 and 3 $^{\circ}$C/min for a node in an enclosure, and higher for a bare PCB node.

- Maximum short term temperature change: the maximum temperature change that can occur within minutes to a few hour, in many applications it is way less than the difference between the maximum and minimum ambient temperature. Can be the difference between the initial and final temperature during a shade/sunlight transition or the difference between day and night temperature.

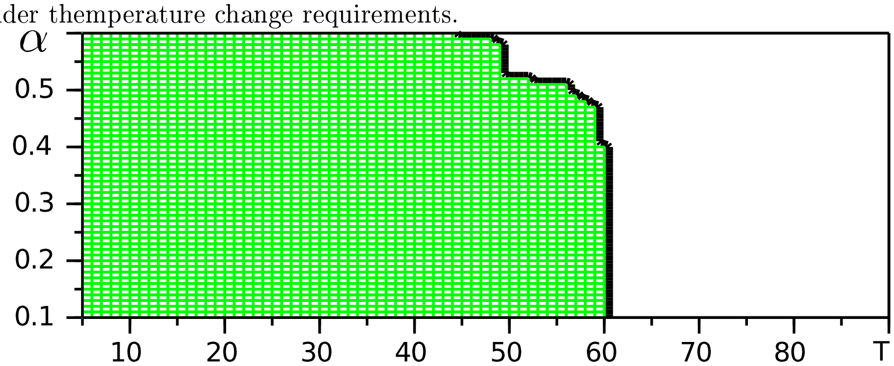
- Whether to simulate only positive temperature steps (i.e, from a lower temperature to a higher one), negative ones, or both. This is useful for asymmetric use cases in environments sthat have fast temperature changes in one direction, but slow changes in the other. The default one is 2, or to simulate both rising and falling temperatures.

Last, there are the shnchronization requirements. Given a typical synchronisation error during a fast temprature change, the maximum tolerable synchronization error is A, the acceptable error threshold after a temperature change is B, while the time to recover is C.



Provided results

The tool provides in graphical form the controller parameters T (synchronization period) and α (controller parameter in (1)) that satisfy the synchronization error under themperature change requirements.



Note that you should aways try to choose the highest possible values of both parameters, as high values of α reduce the controller sensitivity to jitter (the plots are capped at $\alpha = 0.6$, as for higher values the $\mathcal{H}2$ norm increases), thereby resulting in better synchronization error standard deviation, while high values of the synchronization period reduce power consumption by needing less frequen syncronizations. Therefore, the edge of the feasible region, highlighted in red in the previous figure, can be seen as a Pareto curve for the FLOPSYNC-2 controller under the prescribed design parameters, and allows to optimize the jitter versus power tradeoff.

Frequently asked questions

Q: I've input the exact minimum and maximum temperature of a thermal transient experiment I've done and the tool seems to be quite conservative in the required synchronization period to meet the constraints, what is happening?

A: Assuming as an example that you done have a transient experiment between a start temperature of 20°C and an end temperature of 40°C, the tool simulates both positive transient (from 20 to 40°C, and a negative one, from 40 to 20°C). A synchronisation period, to pass the constraint, has to have acceptable error with both transients. Try configuring the tool to simulate only your transient (i.e: only the positive or the negative one) and it won't be conservative in estimating the period. However, note that unless you know that thd opposite transient will never happen in your application, the tool is right, that synchronization period doesn't meet the constraints. Due to the parabolic nature of the temperature to frequency relation, the peak synchronization error is worse if the start temperature of the thermal transient is far from the crystal turnover temperature. So, a transient from 40 to 20°C results in higher peak errors than one from 20 to 40°C.

Configuring FLOPSYNC-2

Once the appropriate parameters are selected, to apply the required changes to the FLOPSYNC-2 source code two changes need to be done:

The synchronization period is set in the *sync_period=60* line of the *experiment.conf* file, which affects the *nominalPeriod* constant in *protocol_constants.h*.

For the α parameter, there is no single place to set it due to the optimized fixed point implementation of the FLOPSYNC-2 controller. You need to edit the `int u=2*uo-uoo+960*e-1320*eo+485*eoo;` line of the member function `pair<int,int> OptimizedRampFlopsync2::computeCorrection(int e)` in *flopsync2.cpp*. The optimized implementation requires α to be a fractional value with a power of two as denominator, and is much simpler if the denominator is 8, thereby choosing values like 1/8, 2/8, 3/8...

If the denominator is 8, the coefficients can be chosen using this line of wxMaxima code, where you can change *\alpha=3/8* to the desired value:

`rat(subst(\alpha=3/8,(3*(1-\alpha)*z^2+3*(\alpha^2-1)*z+1-\alpha^3))*512);`

If the denominator is not 8, you have to consider that an α^3appears in the controller formula (1), so you have to substitute 512 with the third power of your denominator in the formula and also in other places in the *OptimizedRampFlopsync2* class. Also, you have to check that the increased denominator value does not introduce numerical overflows in the synchronization algorithm.