Threshold minimization procedure based on SCurveSmartRange calibration

Introduction

The procedure described in this document aims to lower the thresholds on each ROC to just above the failing point. The SCurveSmartRange calibration is used in place of Standard SCurve calibration in order to check the status of each ROC after having decreased the threshold at each iteration.

In the following description it is assumed that good signal properties have been already obtained running basic calibrations. Good address levels are also required, so an AddressLevel calibration is strongly recommended before starting, if not done yet. Threshold minimization procedure starts from Vana settings obtained doing Vana calibration, so a CalDelCalibration is assumed to be done already, but if this is not the case, it is better to do it in order to have a good starting point for threshold minimization.

An overview about the SCurveSmartRange calibration and the entire method is given in the following talk:
https://indico.cern.ch/event/316543/contribution/3/material/slides/0.pdf
The code for the automatic procedure is available in github: https://github.com/decosa/PixelCalibrations.
Use it under PixelAnalysisTools/test/additionalTools/SCurveSmartRange in POS.

For impatient users

This is the short description of the procedure, for a detailed one, please, go to the next paragraph.

1) VcThrCalibration - Begin the procedure by doing a VcThrCalibration targetting 50 VCal. This gives the starting point for the VcThr setting of each ROC.

2) VcThrVcal - Restart supervisors and do a VcThrVcal calibration considering the bunch crossing next to the InTime one (WBC = 155). Then run the SCurveSR.py scripts

```
python2.7 SCurveSR.py -i 0 -r <VcThrVcal_runNmber>
```

3) SCurveSmartRange - Do an SCurveSmartRange and as soon as it is finished run the script SCurveSR.py:

```
python2.7 SCurveSR.py -i <iteration> -r <SCurveSR_runNmber>
```

Start from iteration 1. Restart supervisor before moving at the next iteration. Repeat until you get 0 succeeding ROCs.

4) Final checks - At this point the minimized threshold for all the ROCs has been set. A standard SCurve to get the final value of the mean threshold is recommended together

with a PixelAlive with all pixels enabled. If there are ROCs failing this check, the threshold for them should be raised by hand

Full description

1) VcThrCalibration - begin the procedure by doing a VcThrCalibration targetting 50 VCal. This gives the starting point for the VcThr setting of each ROC.
Use the following *calib.dat*

```
Mode: VcThrCalibration
Rows:
10 | 30
Cols:
10 | 30
VcalLow
Scan: WBC 154 156 1
Scan: VcThr 20 170 1
Set: Vcal 50
Repeat: 10
ToCalibrate:
all
```

2) VcThrVcal - Restart supervisors and do a VcThrVcal calibration considering the bunch crossing next to the InTime one (WBC = 155). Tha *calib.dat* should be:

```
Mode: 2DEfficiencyScan
Rows:
10
Cols:
10
VcalLow
Scan: Vcal  0 150 2
Scan: VcThr 0 150 2
Set: WBC 155
Repeat: 5
ToCalibrate:
+ all
```

Then run the SCurveSR.py scripts from PixelAnalysisTools/test/additionalTools/SCurveSmartRange folder. The iteration number to specify is 0:

```
python2.7 SCurveSR.py -i 0 -r <VcThrVcal_runNmber>
```

This will analyze the results from VcthrVcal Calibration, fitting the VcThr Vcal plot and finding the parameters which describes the linear dependence of VcThr on Vcal. A text file called mapROCVcThrVcal.txt is created storing per each ROC the fit parameters. If there are failing fits, the fitting range should be changed in analysisCalibFuncs.py in the function fitVcThrVcal.py (a failure in the fits happens really rearely).

It is possible to save the fit plots setting to True the save option by doing:

```
    python2.7 SCurveSR.py -i 0 -r <VcThrVcal_runNmber> -s True
```

3) SCurveSmartRange - Run a series of iterations based on SCurvaAMartRange calibration. The configuration to use for this calibration is the following *calib.dat*:

Mode: SCurveSmartRange
```
Rows:
0 |
9 |
18 |
27 |
36 |
45 |
54 |
63 |
72
Cols:
3 16 29 |
42 4 17 |
30 43 5
VcalLow
Scan: Vcal 40 70 1
Scan: WBC 155 156 1
Repeat:
15
ToCalibrate:
all
```

Do an SCurveSmartRange and as soon as it is finished run the script SCurveSR.py:

```
    python2.7 SCurveSR.py -i <iteration> -r <SCurveSR_runNmber>
```

The first iteration should be 1 and not 0.
The scripts run the SCurve analysis and also look at results to identify succeeding and failing ROCs.

Selection: a ROC is considered as failing if its mean threshold is lower than 35 or if there are more than 2 pixels with threshold lower than 30 or greater than 120.

Threshold settings: threshold for failing ROCs is increased by 4 units, while threshold for succeeding ROCs are decreased by 2 units. ROCs already stabilized in previous iterations are not changed further in threshold.

New dac settings are created by the scripts according to these criteria.
Restart supervisor before moving at the next iteration. Repeat until you get 0 succeeding ROCs. The procedure should converge in about 10/11 iterations.

4) Final checks - At this point the minimized thresholds for all the ROCs have been set. A standard SCurve to get the final value of the mean threshold is recommended together

with a PixelAlive with all pixels enabled. If there are ROCs failing this check, the threshold for them should be raised by hand.