Online TDU/DCM Timing Calibration Procedure

Evan Niner, Indiana University

July 8, 2014

Abstract

This documentation describes the design, implementation, and procedures for calibrating the timing chains at the Near and Far NOνA detectors. Steps to perform the calibration, validate results, load into the database, and retrieve results will be described. The document will also outline current monitoring tools and suggestions for future improvements.

Contents

1 Overview of Timing System

In the NOνA experiment a hierarchical system is used to distribute timing commands from a master timing distribution unit (MTDU) to each individual electronic component of the detector [1]. The Near and Far Detectors employ the same timing technology and each detector has two redundant timing chains. To meet the physics goals the timing system must be capable of synchronizing both detectors to better than 10 ns.

The Far Detector timing layout is shown in figure 1. The MTDU is connected by a single optical fiber to the first slave timing distribution unit (STDU) in the chain. Fourteen STDUs are daisy chained together, one per kiloton of detector, with copper connections providing 4 LVDS lines, master clock, command, SYNC and SYNC return. Each STDU supports two branches of six data concentrator modules DCMs, one for the horizontal planes and one for vertical. The DCM branch is terminated with a loopback connector so delay calibration can be performed. Each DCM fans out to 64 front end boards (FEBs) with copper links carrying master clock, command and SYNC lines. The SYNC return is replaced with a high speed data line to transmit hit information from the cells, meaning the delay cannot be individually calibrated and the length of all FEB cables must be uniform. The

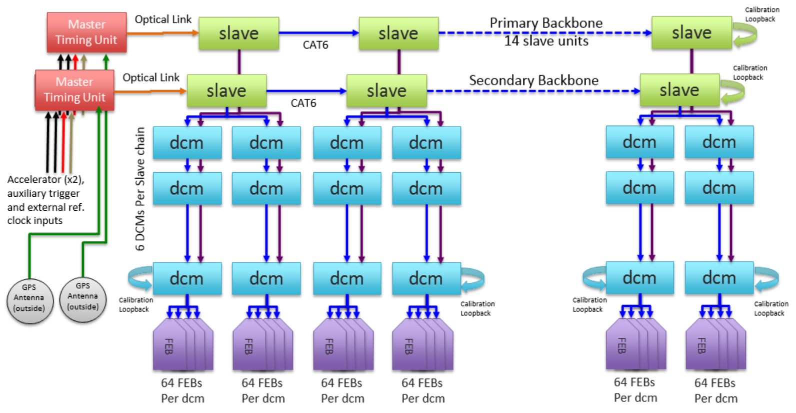


Figure 1: Schematic of the NOνA distributive timing system deployed at the Far Detector.

Near Detector is smaller in size and consists of three diblocks and a muon catcher. Each diblock has four DCMs and the muon catcher consists of two. There are two STDUs which each timing out branch being connected to one whole diblock. The timing runs from the side of the diblock up and across the top. This is done so that each diblock can be separated during electronics installation. These design changes are shown in figure 2.

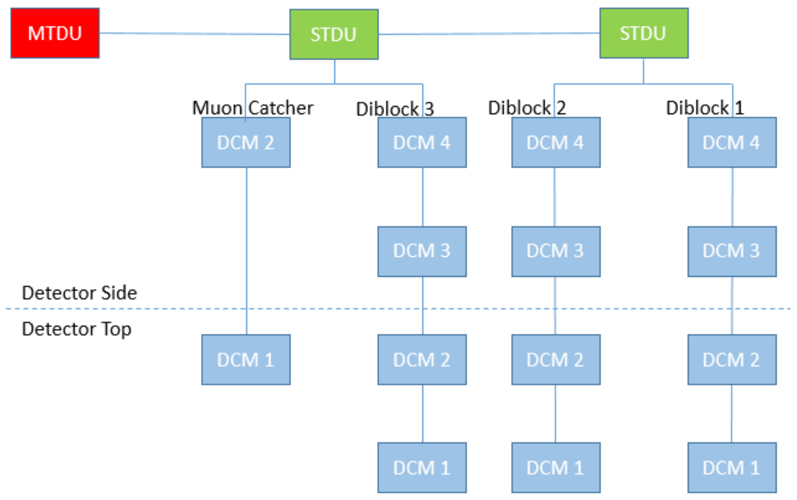


Figure 2: Schematic of the NOνA distributive timing system deployed at the Near Detector. Note the main difference that each diblock is on one branch of an STDU and the detector top and side is not separated.

To precisely synchronize the time stamp counters to the NOνA time, the timing signal propagation delays between each component of the system must be calibrated. The delay calibration is initiated by setting the "learn enable" bit in the control register of the MTDU [2]. This initiates the delay calibration in each STDU and DCM. The delay offset value, which is arbitrary but must be larger then the maximum travel time from the MTDU to the farthest element in the system, is loaded into each unit. Upon receipt of the next SYNC signal sent from the MTDU, every element of the chain clears its time register and initiates a counter. The counter is stopped when the unit receives a return SYNC from the loopback. Each STDU then loads one half of the time-of-flight (TOF) value into its delay register. Each slave keeps an independent counter for the delay value down each DCM branch. The delay offset value is added to the delay calculated for the slave backbone and then one half the TOF for the DCM branch is subtracted to compute the value loaded into the STDU delay register for each DCM branch. The MTDU loads its delay value, which is the total delay down the STU backbone plus the delay offset value into the "early SYNC" register. This is the time prior to a GPS 1 second boundary that a sync procedure needs to begin to ensure all units see the same time. This delay calibration can be performed periodically to monitor stability and seasonal temperature variations. Figures 3 and 4 illustrate the stability of the calibration within one 128 MHz clock tick. The higher precision counter is used so the delay is known more precisely then the digitization clock sampling rate used on the FEB at the Far Detector (16 MHz) or Near Detector (64 MHz).

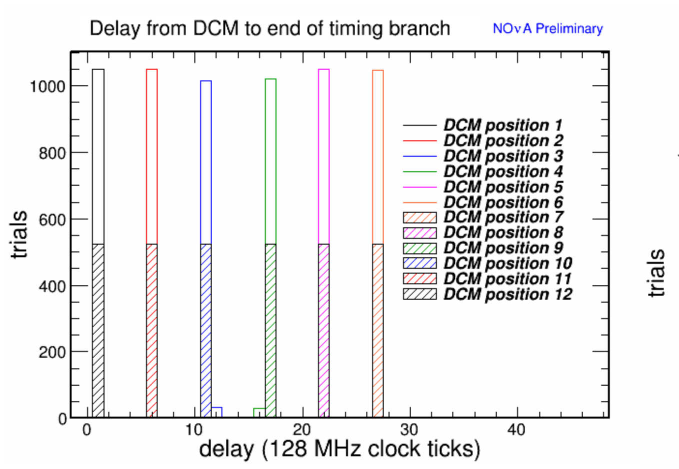


Figure 3: Delay measured from a DCM to the end of the timing branch. DCMs 1-6 are in a branch on top of the detector with 6 being closest to the STDU backbone. DCMs 7-12 are located in a branch on the side, with 7 being closest to the backbone. Statistics are for one timing chain and are summed over the first 8 kilotons of detector instrumented.

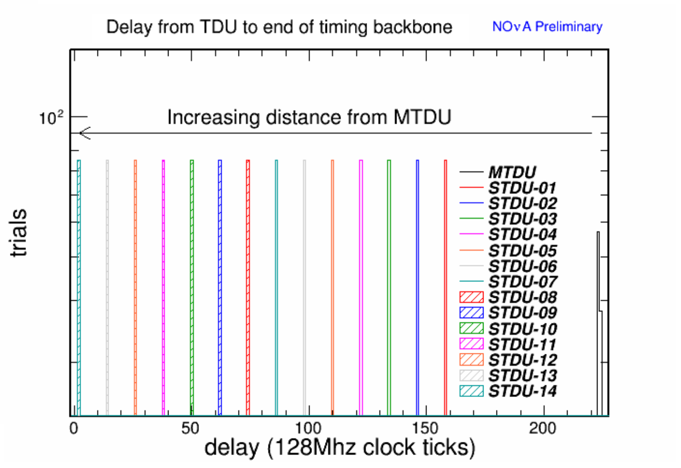


Figure 4: Delay calibrated from a timing unit to the end of the backbone. The MTDU is located off of the detector hall and has a longer delay. Each STDU is uniformly spaced 12 clock ticks apart. One timing chain is pictured, delay values are identical for the second chain.

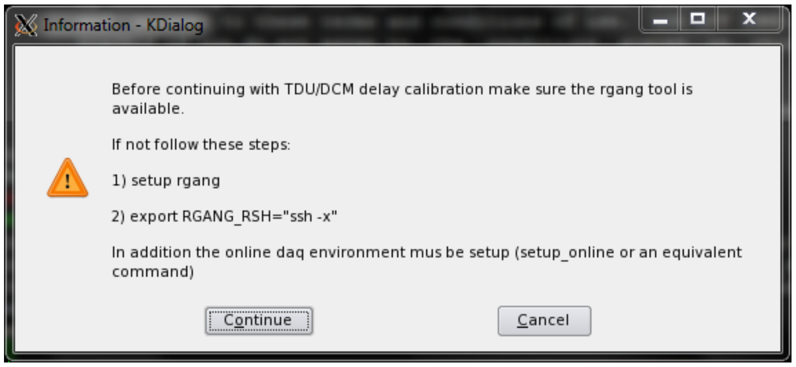
The timing system uses the scheme "At the tone the time will be..." to synchronize the detector. When a SYNC is requested the master timing unit looks at the current time and determines how close it is to the next 1 second GPS boundary. It then uses the delay loaded in the "early SYNC" register to calculate the next 1 second boundary sufficiently far in the future to complete transmission of the sync to all regions of the detector. This new NOνA time is sent out and pre-loaded in to the registers of each system component. The SYNC pulse is sent prior to the upcoming 1 second boundary calculated so the pulse reaches all elements before the designated time. When a sync is received by an electronics component (TDU, DCM, FEB) it is placed in a delay loop buffered with the calibrated value. With proper calibration the entire detector will exit the buffer loop and begin counting from the new NOνA time simultaneously. After the SYNC is complete the time stamp register in each device runs free, driven by the 10 MHz reference clock.

2 Timing Calibration Procedure

Scripts have been designed to perform the TDU delay-learn calibration procedure described in the previous section in an automated fashion with the output recorded in text files to be uploaded to the online database. The timing calibration script is timing_chain_delay_calibration.sh and is found in the online package *TDUUtilities*. The calibration is performed in the following steps:

1. Log into a machine on the network for the detector the calibration will be performed on (typically novadaq-near-master or novadaq-far-master-02). Setup the online software environment and also make sure the rgang utility is available.

2. Launch the timing_chain_delay_calibration.sh script and acknowledge the pop-up instructions in figure 5.



Before continuing with TDU/DCM delay calibration make sure the rgang tool is available.

If not follow these steps:

1) setup rgang

2) export RGANG_RSH="ssh -x"

In addition the online daq environment mus be setup (setup_online or an equivalent command)

Figure 5:

3. When prompted select the detector type the calibration will be performed on shown in figure 6. If the selected detector does not agree with the novadaq environment setup an error message will appear and the program will exit, seen in figure 7.

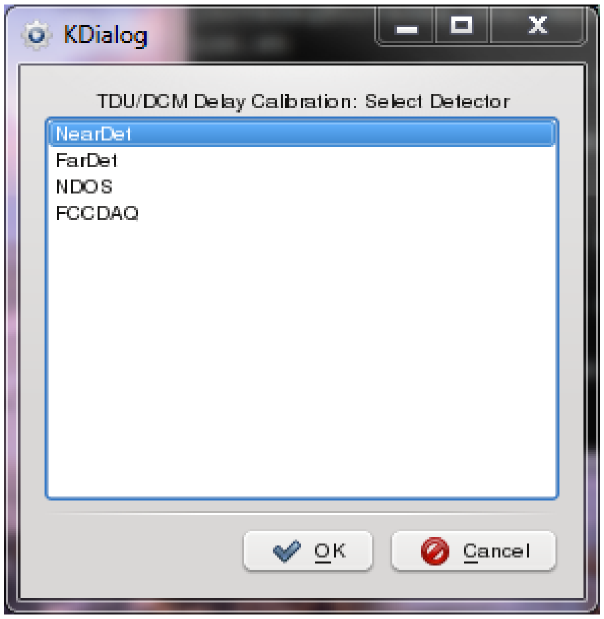
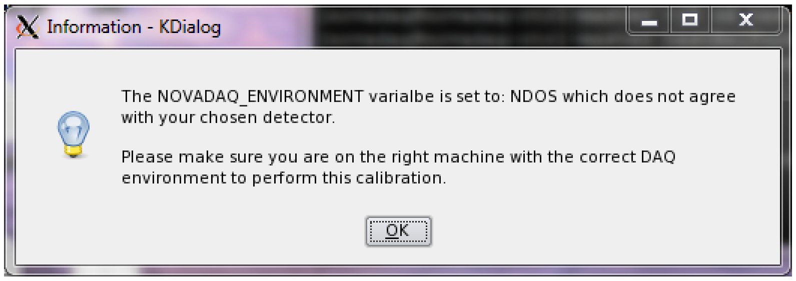


Figure 6:



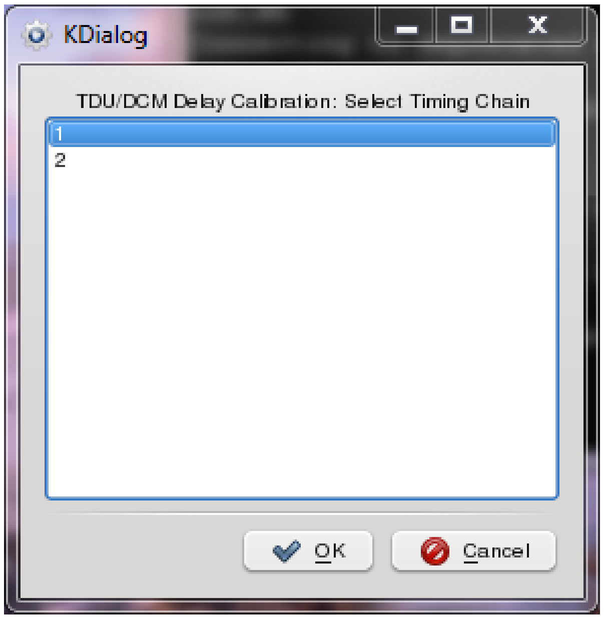
The NOVADAQ_ENVIRONMENT varialbe is set to: NDOS which does not agree with your chosen detector.

Please make sure you are on the right machine with the correct DAQ environment to perform this calibration.

Figure 7:

4. Next select the timing chain to calibrate, either 1 or 2, shown if figure 8. Note that DCMs must be setup to listen to the selected timing chain in order to record the calibration values. A DCM can toggle

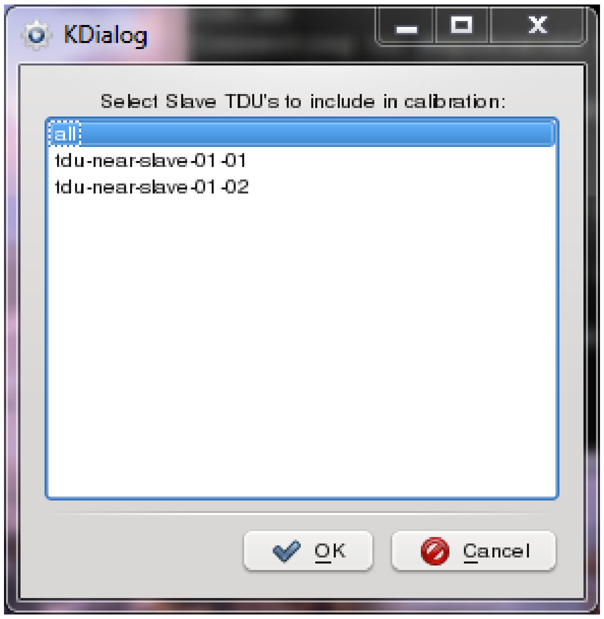
between external and internal timing modes, but once a chain has been selected a DCM can only switch to the other chain after a reboot [3]. If all DCMs are currently using timing chain 1 and the user wishes to calibrate timing chain 2, all DCMs should be rebooted for a clean state before starting this procedure.


TDU/DCM Delay Calibration: Select Timing Chain

1
2

Figure 8:

5. Select which STDUs to record calibration results for, as shown in figure 9. Note that if a unit is not included in the calibration its delay value will not be recorded but it still must be on unless the timing chain termination has changed. There can be no missing links in the middle of the calibration.

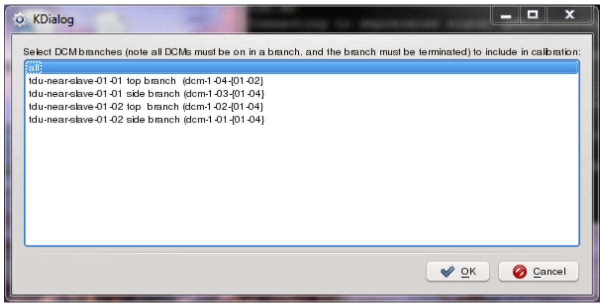


Select Slave TDU's to include in calibration:

```
all
tdu-near-slave-01-01
tdu-near-slave-01-02
```

Figure 9:

6. Select which DCM branches to record calibration results for, shown in figure 10. There is no functionality to select individual DCMs within a branch. The reason is that if the timing of a specific DCM changed the entire DCM branch could be effected and thus calibration constants should only be recorded as a group. All DCMs in a selected branch must be turned on or the calibration will record timeout values.



Select DCM branches (note all DCMs must be on in a branch, and the branch must be terminated) to include in calibration:

```
all
tdu-near-slave-01-01 top branch  (dcm-1-04-{01-02}
tdu-near-slave-01-01 side branch (dcm-1-03-{01-04}
tdu-near-slave-01-02 top  branch (dcm-1-02-{01-04}
tdu-near-slave-01-02 side branch (dcm-1-01-{01-04}
```

Figure 10:

7. Next the calibration is performed. Each DCM is switched into external

timing mode on the selected timing chain and any existing delays are cleared. Then on the MTDU the delay-learn procedure is activated by writing the value 0x0010 to the control register (0x0000). The procedure takes 10 seconds. After that the delay value from each TDU and DCM is read and recorded. The utility module tduDelayLearn.cc handles the TDU programming and recording. The script then reads the DCMs with rgang. The final step in the procedure is to clear the delay values from the DCM so they are not set permanently in case something went wrong. A pop-up window indicates that the procedure is finished.

8. The calibration results are stored in the directory path:

   /daqlogs/${DetID}/TDUDelayCalibrationConstants/TimingChain${chainNum}/

   There are four files produced from each calibration:

   - TDU_delay_total_output_${DetID}_${chainNum}_YYYYMMDD_hhmmss.out
   - TDU_delay_side_output_${DetID}_${chainNum}_YYYYMMDD_hhmmss.out
   - TDU_delay_top_output_${DetID}_${chainNum}_YYYYMMDD_hhmmss.out
   - DCM_delay_output_${DetID}_${chainNum}_YYYYMMDD_hhmmss.out

   NOTE: The delay values stored are in an integer number of 128 MHz clock ticks. To load the value into a DCM or TDU it must be converted back into hexadecimal.

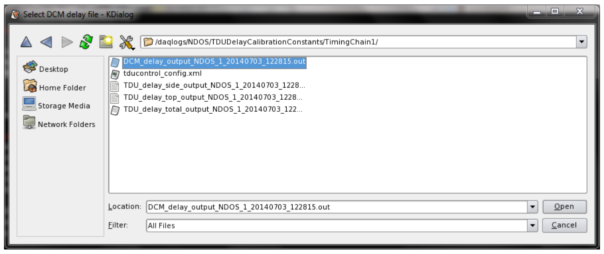
NOTE: After the completion of these steps, the timing calibration has been set for the TDUs. This takes out the cable delay between diblocks at the Far Detector. The DCMs have not had a calibration applied and none of the results are in the database yet. To load results proceed to the next section.

3 Timing Calibration Validation and Database Loading

After a completed timing calibration the next step is to verify that the results are valid and then load into the database. This is done with the timing_chain_delay_validation.sh script also found in the TDUUtilities package. This procedure is done interactively in the following steps:

1. Log into a machine on the network for the detector the calibration will be performed on (typically novadaq-near-master or novadaq-far-master-02). Setup the online software environment and launch the timing_chain_delay_validation.sh script.

2. A window will pop-up prompting the user to select the desired timing delay file. The user is expected to select the DCM delay output file and then the TDU delay files with the corresponding time stamp are found automatically, shown if figure 11.

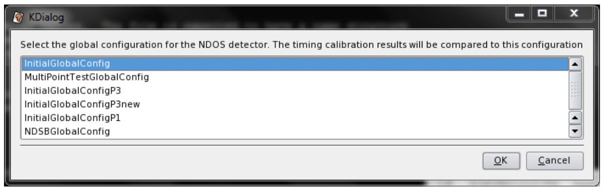


```
Select DCM delay file - KDialog
/daqlogs/NDOS/TDUDelayCalibrationConstants/TimingChain1/

DCM_delay_output_NDOS_1_20140703_122815.out
tducontrol_config.xml
TDU_delay_side_output_NDOS_1_20140703_1228...
TDU_delay_top_output_NDOS_1_20140703_1228...
TDU_delay_total_output_NDOS_1_20140703_122...

Location: DCM_delay_output_NDOS_1_20140703_122815.out
Filter:   All Files
```

Figure 11:

3. An informational box appears to inform the user that the database will be checked for all available global configurations with the current detector ID.

4. User selects the desired global configuration from the list, shown in figure 12.



```
KDialog
Select the global configuration for the NDOS detector. The timing calibration results will be compared to this configuration

InitialGlobalConfig
MultiPointTestGlobalConfig
InitialGlobalConfigP3
InitialGlobalConfigP3new
InitialGlobalConfigP1
NDSBGlobalConfig
```

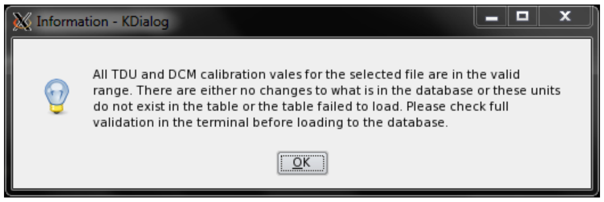
Figure 12:

5. Next the utility module ValidateTimingDelayConstants.cc is used to compare the DCM and TDU delay values to the selected global configuration. The results of this validation are printed to the terminal. If any value deviates by 1 clock tick (at 128 MHz) it is flagged. Values are

also required to be in a valid range (1-254 clock ticks for DCM delays and 1-60000 clock ticks for TDUs). If no database table is available or a unit is missing from it, the value will be flagged as new and only checked that the range is valid. Final results are printed in the the following summary form:

```
***Performing validation of DCM Delay Values***

***Summary of DCM delay validation***
DCMs calibration was performed on: 7
DCMs with a valid calibration result: 7
DCMs with a result outside normal range: 0
DCMs which deviated from value in the database: 0

***Performing validation of TDU Total Delay Values***

***Summary of TDU total delay validation***
TDUs calibration was performed on: 3
TDUs with a valid calibration result: 3
TDUs with a result outside normal range: 0
TDUs which deviated from value in the database: 0

***Performing validation of TDU Top Delay Values***

***Summary of TDU top delay validation***
TDUs calibration was performed on: 2
TDUs with a valid calibration result: 2
TDUs with a result outside normal range: 0
TDUs which deviated from value in the database: 0

***Performing validation of TDU Side Delay Values***

***Summary of TDU side delay validation***
TDUs calibration was performed on: 2
TDUs with a valid calibration result: 2
TDUs with a result outside normal range: 0
TDUs which deviated from value in the database: 0
```

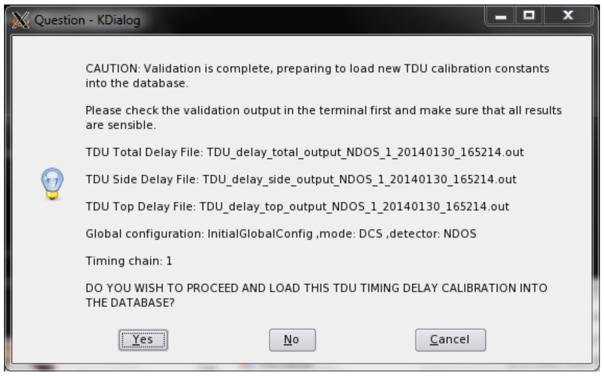
After the validation an informational box will appear indicating if any values changed or were outside the normal range to alert the use, example shown in figure 13.


Information - KDialog

All TDU and DCM calibration vales for the selected file are in the valid range. There are either no changes to what is in the database or these units do not exist in the table or the table failed to load. Please check full validation in the terminal before loading to the database.

Figure 13:

6. Next the TDU delay values will be loaded to the database by the utility module LoadTDUTimingDelayConstants.cc. A window summarizes the database parameters that will be used, shown in figure 14.


Question - KDialog

CAUTION: Validation is complete, preparing to load new TDU calibration constants into the database.

Please check the validation output in the terminal first and make sure that all results are sensible.

TDU Total Delay File: TDU_delay_total_output_NDOS_1_20140130_165214.out

TDU Side Delay File: TDU_delay_side_output_NDOS_1_20140130_165214.out

TDU Top Delay File: TDU_delay_top_output_NDOS_1_20140130_165214.out

Global configuration: InitialGlobalConfig ,mode: DCS ,detector: NDOS

Timing chain: 1

DO YOU WISH TO PROCEED AND LOAD THIS TDU TIMING DELAY CALIBRATION INTO THE DATABASE?

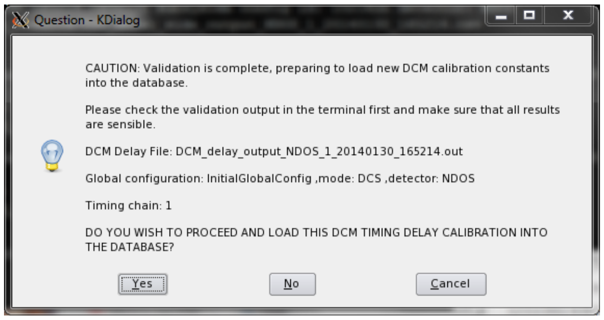
Figure 14:

The commands being sent to the database will be displayed in the terminal and look similar to the following:

```
Table::WriteToDB: Executing PGSQL command:
        INSERT INTO ndos.tdu_timing_delay_settings (tduname,delay,
        delaytype,subsyscfgid,inserttime,insertuser)
        VALUES ('tdu-03',139,'total',10218,'2014-07-03 18:27:42','novadaq')
Table::GetCurrSeqVal: Executing PGSQL command:
        SELECT last_value FROM ndos.tdu_timing_delay_settings_id_seq
```

NOTE: If an error occurs while writing to the database the following wiki can be used as a guide for recovering the table: `https://cdcvs.fnal.gov/redmine/projects/novadaq-config-mgmt/wiki/How_To_recover_from_a_failed_pedestal_upload_-_December_2011`.

7. A pop-up box will appear indicating that the TDU delays have been loaded successfully. At this point a second window will ask for acknowledgement before proceeding to load the DCM delays into the database, shown in figure 15.



CAUTION: Validation is complete, preparing to load new DCM calibration constants into the database.

Please check the validation output in the terminal first and make sure that all results are sensible.

DCM Delay File: DCM_delay_output_NDOS_1_20140130_165214.out

Global configuration: InitialGlobalConfig ,mode: DCS ,detector: NDOS

Timing chain: 1

DO YOU WISH TO PROCEED AND LOAD THIS DCM TIMING DELAY CALIBRATION INTO THE DATABASE?

Figure 15:

The DCM delays are loaded in two steps by the utility module LoadDCMTimingDelayConstants.cc. First the values are loaded into the online database in the same fashion as the TDU values. If this succeeds then the values are also loaded into an sqlite database file. The reason for this is so DCMs can access delay values at boot time without being required to connect to the online database. This file is kept in line with the online version. More information on the location and formatting of these tables can be found in the next section.

8. Finally a pop-up will indicate that the DCM delay values have been loaded successfully. At this point all delay values have now been updated into the database and the procedure is done. NOTE: in the current design these values need to be loaded separately for each global configuration that may use them. The above procedure needs to be repeated as necessary.

4 Database Tables

The TDU timing delays are stored in the table TDUTimingDelaySettings.xml, defined in the NovaDatabase package under *tables/DAQConfig/*. The columns are *tduName*, *delay*, and *delayType*. The *tduName* is the host name of the unit and is a string. The *delay* is the integer number of 128 MHz clock ticks recorded in the delay-learn procedure. The *delayType* is a string representing the type of delay value. Expected types are:

- total: delay value from that TDU down to the end of the TDU backbone.
- side: delay value from that TDU down to the end of the side out dcm branch.
- top: delay value from that TDU down to the end of the top out dcm branch.
- early: For the MTDU only. The number of clock ticks a sync pulse should be issued BEFORE the next 1 second GPS time boundary such that the sync is applied at the 1 second mark.

The combination of *tduName* and *delayType* is unique for a subsystem configuration id. This table is located in the named subsystem configuration TDUManager Hardware.

The DCM timing delays are stored in the online database in the table DCMTimingDelaySettings.xml, defined in the NovaDatabase package under *tables/DAQConfig/*. The columns are *dcmName*, *delay*, and *port*. The *dcmName* is the host name of the unit and is a string. The *delay* is the integer number of 128 MHz clock ticks measured in the calibration. The *port* is an integer (0 or 1) for the timing chain the DCM was listening to. The combination of *dcmName* and *port* is unique for a subsystem configuration id. This table is located in the named subsystem configuration DCMApplication Hardware.

The DCM delays are also loaded into an sqlite database file. This file lives in:

/nova/config/${NOVADAQ_ENVIRONMENT}/DCMTimingDelayCalibration/
and is called DCM_Timing_Delay_Constants_${NOVADAQ_ENVIRONMENT}_table.db. This file maintains a copy of the online DCM database calibrations and is

updated through the same code that loads the online values. The table contains the columns *dcm*, *port*, *delay*, and *time*. Where *dcm* is the host name string, *port* is the timing chain, *delay* is the integer delay value, and *time* is the timestamp when the entry is created. This table is not tied to a specific global configuration since a DCM will not know this when the file is read at boot time. A DCM will by default retrieve the value with the most recent timestamp. If an older value is required it is best to reload that configuration to move it to the front of the table.

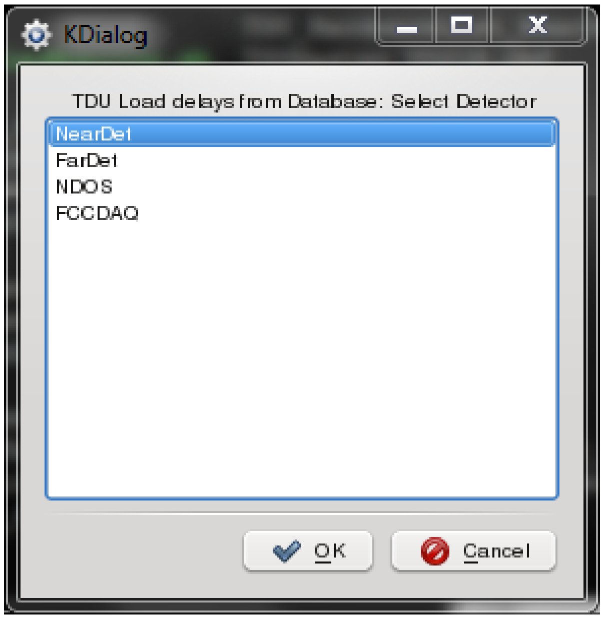
5 Load DCM Delay Values from Database

Utilities have been designed to load the DCM delay values at boot time. This is done so that the delay is set before APD cooling is enabled. After cooling is enabled a change in the timing delay causes a brief clock interruption. This interruption stops the cooling circuits. When cooling resumes all the APDs start at once and the DCM is tripped off from an over current draw. The script dcmLoadDelay.sh in the $DCM_ProgUtils$ package runs this process. This process runs on the DCM and determines the host name and locates the sqlite file discussed in the previous section. The script also looks for a file Timing_Port_${NOVADAQ_ENVIRONMENT}.txt located in the same directory as the sqlite file. This file contains either a 0 or 1, corresponding to the timing port. If the file is not found the default is 0. This is done because a DCM will not have an external timing chain set yet at boot time. Then the utility module dcmDelay.cc is called. This takes the delay value from the sqlite file with the most recent timestamp for the specified dcm name and port and writes it to register 0x4c on the DCM. This register stores manual timing delay values and is separate from the register used when the automatic delay calculation procedure is initiated from the TDU. Next in the control register (0x0) bit 5 (0x20) is flipped high for half a second to load the delay value into the timing circuit. If the dcm is not found in the sqlite table no delay is written. Note that sqlite must be available on the crosscompiler for that DCM. This procedure has been added to the rc.dcm boot script after the IOC has been started. Once this delay is set any soft reset of the FPGA will clear these delay values.

6 Load TDU Delay Values from Database

Utilities also exist to load the TDU delay values from the database without re-running the calibration procedure. In the long term the TDU firmware is being redesigned so that the delays get loaded from memory at boot time. The utility tdu_load_delay_from_database.sh exists in the *TDUUtilies* package. To run this procedure perform the following:

1. From novadaq-near-master or novadaq-far-master-02 setup the online environment and launch the script. The is will open a window prompting the user to select the detector type, shown in figure 16.

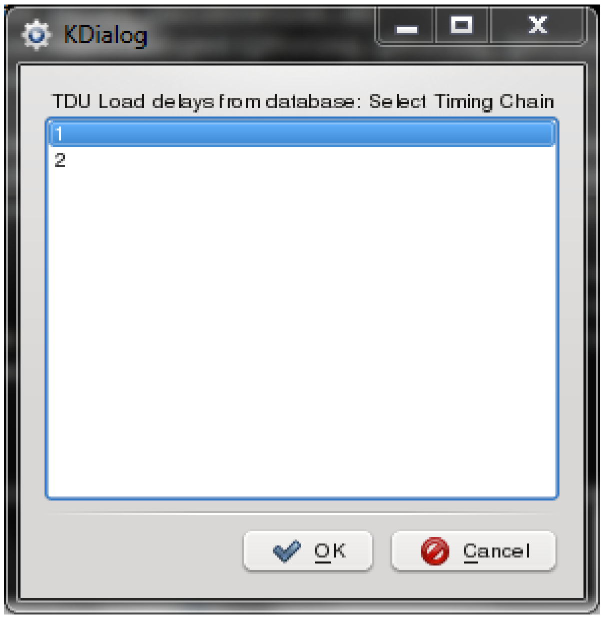


TDU Load delays from Database: Select Detector

NearDet
FarDet
NDOS
FCCDAQ

Figure 16:

2. Then the user selects the timing chain, figure 17.

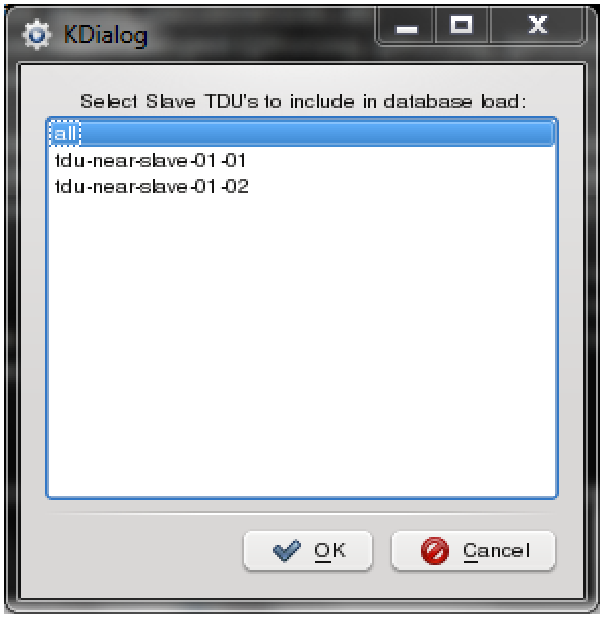


TDU Load delays from database: Select Timing Chain

1
2

Figure 17:

3. The MTDU will always have its delay loaded, the user is then prompted to select which slave units to also apply calibration to, figure 18.

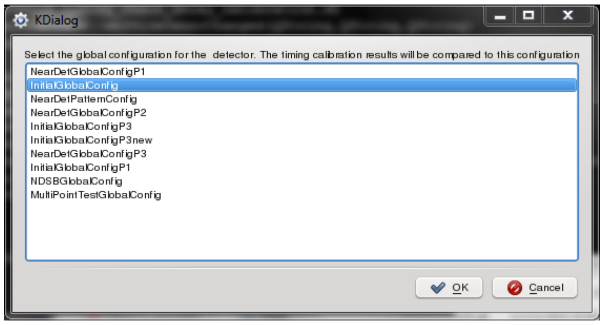

KDialog

Select Slave TDU's to include in database load:

all
tdu-near-slave-01-01
tdu-near-slave-01-02

Figure 18:

4. Finally the user selects which global configuration to retrieve the delay constants from, figure 19.


KDialog

Select the global configuration for the detector. The timing calibration results will be compared to this configuration

NearDetGlobalConfigP1
InitialGlobalConfig
NearDetPatternConfig
NearDetGlobalConfigP2
InitialGlobalConfigP3
InitialGlobalConfigP3new
NearDetGlobalConfigP3
InitialGlobalConfigP1
NDSBGlobalConfig
MultiPointTestGlobalConfig

Figure 19:

5. Next the TDU delay values are programmed by the module LoadTDUDelaysFromDatabase.cc. First the MTDU is programmed. The specific sequence is that the early delay values is set first (register 0xD) followed by the total delay (register 0x2). Then the slaves are set in the

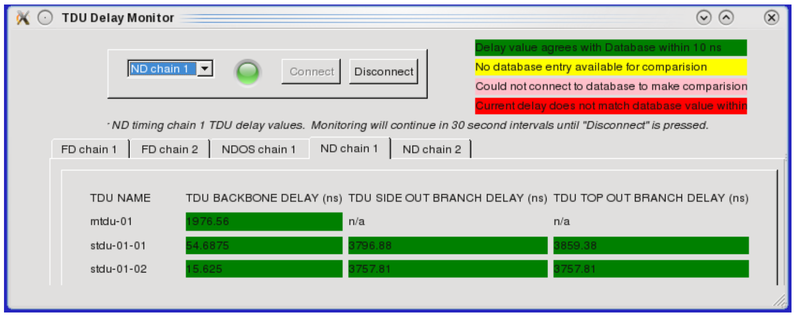
sequence total (register 0x2), side (0x3), top (0x4). Finally, on each slave bit 13 (0x2000) of the control register (0x0) is set high to apply the delay values. The utility tduWriteDelay.cc is used to write one single delay value. Then the utility tduClearErrors.cc is used to reset the error registers on each unit to remove any temporary errors that came from starting and stopping the clocks. While delays are being written, progress is printed to the terminal, similar to the following:

```
For tdu: tdu-near-slave-01-01 with delay: 494
  of type: top attempting to program unit now.
Conntected to: tdu-near-slave-01-01 proceeding to write register
On TDU: tdu-near-slave-01-01 wrote delay: 494 clock ticks at 128 MHZ,
  in hex value of 0x01ee to register: 0x0004
Done programing delay value, exiting
```

7 Monitoring

Currently there is no monitoring in place for the DCM delay values. There has been discussion that the configuration manager will retrieve delay values from the database and pass them to the DCM to verify that the set value agrees with the database. Documentation will be updated when this is complete.

For the TDU delays there is a Qt program TDUDelayMonitor in the *TDUControl* package. This monitor can be launched from desktops on the FD-01 and ND-01 VNC sessions. The monitor connects to a selected timing chain and monitors that the delays stored in the TDU agree with the database in 30 second intervals. Values that agree within 10ns with the database are flagged in green. Larger deviations are shown in red. This monitor is one indicator of an interlock alarm, such as a waterleak. An example of the monitor is shown in figure 20



🟩 Delay value agrees with Database within 10 ns
🟨 No database entry available for comparision
🟥 Could not connect to database to make comparision
🟥 Current delay does not match database value within

ND timing chain 1 TDU delay values. Monitoring will continue in 30 second intervals until "Disconnect" is pressed.

TDU NAME	TDU BACKBONE DELAY (ns)	TDU SIDE OUT BRANCH DELAY (ns)	TDU TOP OUT BRANCH DELAY (ns)
mtdu-01	1976.56	n/a	n/a
stdu-01-01	54.6875	3796.88	3859.38
stdu-01-02	15.625	3757.81	3757.81

Figure 20:

References

[1] Evan Niner, *NOvA timing system procedings for CHEP*. NOvA internal document 10000, `http://nova-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=10000`

[2] Greg Deuerling, Richard Kwarciany and Neal Wilcer, *TDU FPGA Firmware Guide*. NOvA internal document 5370, `http://nova-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=5370`

[3] Richard Kwarciany, *NOvA Data Concentrator Module Hardware User Guide*. NOvA internal document 3989, `http://nova-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=3989`