TSPD-ADQ EPICS device support

1.2.0

Generated by Doxygen 1.8.16

Chapter 1

TSPD-ADQ EPICS device support

TSPD-ADQ EPICS device support is a NDS3 based library that allows to work with ADQ7 and ADQ14 digitizers on EPICS. See also NDS3 API reference manual

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 ADQAIChannel Class Reference

This class handles channel specific parameters and pushes acquired data to appropriate data PVs.

`#include <ADQAIChannel.h>`

Public Member Functions

- ADQAIChannel (const std::string &name, nds::Node &parentNode, int32_t channelNum, ADQInterface ∗&adqInterface, nds::PVDelegateIn< std::string > logMsgPV)

  ADQAIChannel class constructor.
- void setInputRange (const timespec &pTimestamp, const double &pValue)

  Sets the channel's input range.
- void getInputRange (timespec ∗pTimestamp, double ∗pValue)

  Gets the channel's input range.
- void setDcBias (const timespec &pTimestamp, const int32_t &pValue)

  Sets the channel's DC bias.
- void getDcBias (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the channel's DC bias.
- void setChanDec (const timespec &pTimestamp, const int32_t &pValue)

  Sets the channel's sample decimation.
- void getChanDec (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the channel's sample decimation.
- void setState (nds::state_t newState)

  Sets a new state to the ADQAIChannel class' state machine.
- void readData (short ∗rawData, int32_t sampleCnt)

  This method passes the acquired data to appropriate data PV.
- void getDataPV (timespec ∗pTimestamp, std::vector< int32_t > ∗pValue)

  This is a dummy method held by the data PV for appropriate work in NDS3.
- void commitChanges (bool calledFromDaqThread=false)

  This method processes changes are applied to channel specific parameters.

Public Attributes

- int32_t m_channelNum

  Number of channel.

5.1.1 Detailed Description

This class handles channel specific parameters and pushes acquired data to appropriate data PVs.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ADQAIChannel()

```
ADQAIChannel::ADQAIChannel (
            const std::string & name,
            nds::Node & parentNode,
            int32_t channelNum,
            ADQInterface *& adqInterface,
            nds::PVDelegateIn< std::string > logMsgPV )
```

ADQAIChannel class constructor.

Parameters

name	a name with which this class will register its child node.
parentNode	a name of a parent node to which this class' node is a child.
channelNum	a number of channel which a constructed class represents.
adqInterface	a pointer to the ADQ API interface created in the ADQDevice class.
logMsgPV	process variable for sending the log messages (shared with the ADQAIChannelGroup class).

5.1.3 Member Function Documentation

5.1.3.1 commitChanges()

```
ADQAIChannel::commitChanges (
            bool calledFromDaqThread = false )
```

This method processes changes are applied to channel specific parameters.

Parameters

calledFromDaqThread	a flag that prevents this function to be called when set to false.

The documentation for this class was generated from the following files:

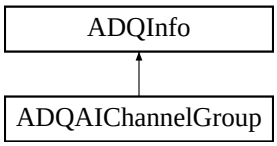
- ADQAIChannel.h
- ADQAIChannel.cpp

5.2 ADQAIChannelGroup Class Reference

This class handles majority of parameters for correct setup of each data acquisition mode. Data acquisition is handled in this class. The state machine of the device is defined here. Each digitizer's channel gets a representation by calling ADQCHannel constructor for N amount of physical channels.

`#include <ADQAIChannelGroup.h>`

Inheritance diagram for ADQAIChannelGroup:



Public Member Functions

- ADQAIChannelGroup (const std::string &name, nds::Node &parentNode, ADQInterface *&adqInterface, void *adqCtrlUnit)

  ADQAIChannelGroup class constructor.
- template<typename T >
  void createPv (const std::string &name, nds::PVDelegateIn< T > &pvRb, std::function< void(ADQAIChannelGroup *, const timespec &, const T &)> setter, std::function< void(ADQAIChannelGroup *, timespec *, T *)> getter)

  This function creates the most common type of PV and sets it readback PV to interrupt mode.
- template<typename T >
  void createPvEnum (const std::string &name, nds::PVDelegateIn< T > &pvRb, nds::enumerationStrings_t enumList, std::function< void(ADQAIChannelGroup *, const timespec &, const T &)> setter, std::function< void(ADQAIChannelGroup *, timespec *, T *)> getter)

  This function creates the Enumeration type of PV and sets it readback PV to interrupt mode.
- template<typename T >
  nds::PVDelegateIn< T > createPvRb (const std::string &name, std::function< void(ADQAIChannelGroup *, timespec *, T *)> getter)

  This function creates and returns the readback PV.
- void setDaqMode (const timespec &pTimestamp, const int32_t &pValue)

  Sets the data acquisition mode.
- void getDaqMode (timespec *pTimestamp, int32_t *pValue)

  Gets the data acquisition mode.
- void setTrigMode (const timespec &pTimestamp, const int32_t &pValue)

  Sets the trigger mode.
- void getTrigMode (timespec *pTimestamp, int32_t *pValue)

  Gets the trigger mode.
- void setDbsBypass (const timespec &pTimestamp, const int32_t &pValue)

  Sets if the DBS settings is bypassed (1) or not (0).

- void getDbsBypass (timespec *pTimestamp, int32_t *pValue)
  Gets if the DBS settings is bypassed (1) or not (0).
- void setDbsDc (const timespec &pTimestamp, const int32_t &pValue)
  Sets the DC target for the DBS.
- void getDbsDc (timespec *pTimestamp, int32_t *pValue)
  Gets the DC target for the DBS.
- void setDbsLowSat (const timespec &pTimestamp, const int32_t &pValue)
  Sets the lower saturation level for the DBS.
- void getDbsLowSat (timespec *pTimestamp, int32_t *pValue)
  Gets the lower saturation level for the DBS.
- void setDbsUpSat (const timespec &pTimestamp, const int32_t &pValue)
  Sets the upper saturation level for the DBS.
- void getDbsUpSat (timespec *pTimestamp, int32_t *pValue)
  Gets the upper saturation level for the DBS.
- void setPatternMode (const timespec &pTimestamp, const int32_t &pValue)
  Sets the pattern mode.
- void getPatternMode (timespec *pTimestamp, int32_t *pValue)
  Gets the pattern mode.
- void setChanActive (const timespec &pTimestamp, const int32_t &pValue)
  Sets which channels should be active for data acquisition.
- void getChanActive (timespec *pTimestamp, int32_t *pValue)
  Gets which channels should be active for data acquisition.
- void getChanMask (timespec *pTimestamp, int32_t *pValue)
  Gets the channel mask accordingly to chosen active channels.
- void setRecordCnt (const timespec &pTimestamp, const int32_t &pValue)
  Sets the number of records to acquire.
- void getRecordCnt (timespec *pTimestamp, int32_t *pValue)
  Gets the number of records to acquire.
- void setRecordCntCollect (const timespec &pTimestamp, const int32_t &pValue)
  Sets the number of records to pass to the device (Multi-Record mode).
- void getRecordCntCollect (timespec *pTimestamp, int32_t *pValue)
  Gets the number of records to pass to the device (Multi-Record mode).
- void setSampleCnt (const timespec &pTimestamp, const int32_t &pValue)
  Sets the number of samples per record.
- void getSampleCnt (timespec *pTimestamp, int32_t *pValue)
  Gets the number of samples per record.
- void getSampleCntMax (timespec *pTimestamp, int32_t *pValue)
  Gets the maximum number of samples per record accordingly to the number of records (Multi-Record mode).
- void getSamplesTotal (timespec *pTimestamp, int32_t *pValue)
  *Gets a total number of samples to acquire (Number of records * Number of samples).*
- void setSampleSkip (const timespec &pTimestamp, const int32_t &pValue)
  Sets the sample skip.
- void getSampleSkip (timespec *pTimestamp, int32_t *pValue)
  Gets the sample skip.
- void setSampleDec (const timespec &pTimestamp, const int32_t &pValue)
  Gets the data acquisition mode..
- void getSampleDec (timespec *pTimestamp, int32_t *pValue)
  Sets the sample decimation (-FWSDR digitizers only).
- void setPreTrigSamp (const timespec &pTimestamp, const int32_t &pValue)
  Sets the number of pre-trigger samples (Multi-Record and Triggered streaming mode).
- void getPreTrigSamp (timespec *pTimestamp, int32_t *pValue)

Gets the number of pre-trigger samples (Multi-Record and Triggered streaming mode).

- void setTrigHoldOffSamp (const timespec &pTimestamp, const int32_t &pValue)

  Sets the number of hold-off samples (Multi-Record and Triggered streaming mode).
- void getTrigHoldOffSamp (timespec *pTimestamp, int32_t *pValue)

  Gets the number of hold-off samples (Multi-Record and Triggered streaming mode).
- void setClockSrc (const timespec &pTimestamp, const int32_t &pValue)

  Sets the clock source.
- void getClockSrc (timespec *pTimestamp, int32_t *pValue)

  Gets the clock source.
- void setClockRefOut (const timespec &pTimestamp, const int32_t &pValue)

  Enables (1) or disables (0) clock reference output.
- void getClockRefOut (timespec *pTimestamp, int32_t *pValue)

  Gets the clock reference output.
- void setTimeout (const timespec &pTimestamp, const int32_t &pValue)

  Sets the flush timeout (Triggered streaming mode).
- void getTimeout (timespec *pTimestamp, int32_t *pValue)

  Gets the flush timeout (Triggered streaming mode).
- void setStreamTime (const timespec &pTimestamp, const double &pValue)

  Sets the streaming time (Continuous streaming mode).
- void getStreamTime (timespec *pTimestamp, double *pValue)

  Gets the streaming time (Continuous streaming mode).
- void setSWTrigEdge (const timespec &pTimestamp, const int32_t &pValue)

  Sets the SW trigger edge.
- void getSWTrigEdge (timespec *pTimestamp, int32_t *pValue)

  Gets the SW trigger edge.
- void setLevelTrigLvl (const timespec &pTimestamp, const int32_t &pValue)

  Sets the trigger level of Level trigger.
- void getLevelTrigLvl (timespec *pTimestamp, int32_t *pValue)

  Gets the trigger level of Level trigger.
- void setLevelTrigEdge (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Level trigger edge.
- void getLevelTrigEdge (timespec *pTimestamp, int32_t *pValue)

  Gets the Level trigger edge.
- void setLevelTrigChan (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Level trigger channel.
- void getLevelTrigChan (timespec *pTimestamp, int32_t *pValue)

  Gets the Level trigger channel.
- void getLevelTrigChanMask (timespec *pTimestamp, int32_t *pValue)

  Gets the Level trigger channel mask.
- void setExternTrigDelay (const timespec &pTimestamp, const int32_t &pValue)

  Sets the External trigger delay.
- void getExternTrigDelay (timespec *pTimestamp, int32_t *pValue)

  Gets the External trigger delay.
- void setExternTrigThreshold (const timespec &pTimestamp, const double &pValue)

  Sets the External trigger treshold.
- void getExternTrigThreshold (timespec *pTimestamp, double *pValue)

  Gets the External trigger treshold.
- void setExternTrigEdge (const timespec &pTimestamp, const int32_t &pValue)

  Sets the External trigger edge.
- void getExternTrigEdge (timespec *pTimestamp, int32_t *pValue)

  Gets the External trigger edge.

- void setInternTrigHighSamp (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Internal trigger high sample length.
- void getInternTrigHighSamp (timespec *pTimestamp, int32_t *pValue)

  Gets the Internal trigger high sample length.
- void setInternTrigLowSamp (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Internal trigger low sample length.
- void getInternTrigLowSamp (timespec *pTimestamp, int32_t *pValue)

  Gets the Internal trigger low sample length.
- void setInternTrigFreq (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Internal trigger frequency.
- void getInternTrigFreq (timespec *pTimestamp, int32_t *pValue)

  Gets the Internal trigger frequency.
- void setInternTrigEdge (const timespec &pTimestamp, const int32_t &pValue)

  Sets the Internal trigger edge.
- void getInternTrigEdge (timespec *pTimestamp, int32_t *pValue)

  Gets the Internal trigger edge.
- void getLogMsg (timespec *pTimestamp, std::string *pValue)

  Gets the log messages.
- void onSwitchOn ()

  Sets the state machine to state ON.
- void onSwitchOff ()

  Sets the state machine to state OFF.
- void onStart ()

  Sets the state machine to state RUNNING and starts data acquisition.
- void onStop ()

  Stops the data acquisition and sets the state machine to state ON.
- void recover ()

  State machine function. Not supported.
- bool allowChange (const nds::state_t currentLocal, const nds::state_t currentGlobal, const nds::state_t nextLocal)

  Allows the state machine to switch to a new state.
- void daqTrigStream ()

  This method processes Triggered streaming data acquisition.
- void daqMultiRecord ()

  This method processes Multi-Record data acquisition.
- void daqContinStream ()

  This method processes Continuous streaming data acquisition.
- void daqRawStream ()

  This method processes Raw streaming data acquisition.

Public Attributes

- nds::Port m_node

  ADQAIChannelGroup class node that connects to the device.
- nds::StateMachine m_stateMachine

  State machine of this class. Attached to the node.
- std::vector< std::shared_ptr< ADQAIChannel > > m_AIChannelsPtr

  Vector of pointers to ADQAIChannel class instances.

Additional Inherited Members

5.2.1 Detailed Description

This class handles majority of parameters for correct setup of each data acquisition mode. Data acquisition is handled in this class. The state machine of the device is defined here. Each digitizer's channel gets a representation by calling ADQCHannel constructor for N amount of physical channels.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ADQAIChannelGroup()

```
ADQAIChannelGroup::ADQAIChannelGroup (
            const std::string & name,
            nds::Node & parentNode,
            ADQInterface *& adqInterface,
            void * adqCtrlUnit )
```

ADQAIChannelGroup class constructor.

Parameters

name	a name with which this class will register its child node.
parentNode	a name of a parent node to which this class' node is a child.
adqInterface	a pointer to the ADQ API interface created in the ADQDevice class.
adqCtrlUnit	a pointer to the ADQ control unit that sets up and controls the ADQ devices.

The documentation for this class was generated from the following files:

- ADQAIChannelGroup.h
- ADQAIChannelGroup.cpp

5.3 ADQDevice Class Reference

This class creates a device that communicates with a digitizer. ADQ Control Unit is handled by this class. The pointer to ADQAPI interface is also created here.

```
#include <ADQDevice.h>
```

Public Member Functions

- ADQDevice (nds::Factory &factory, const std::string &deviceName, const nds::namedParameters_t &parameters)

  ADQDevice class constructor.

5.3.1 Detailed Description

This class creates a device that communicates with a digitizer. ADQ Control Unit is handled by this class. The pointer to ADQAPI interface is also created here.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ADQDevice()

```
ADQDevice::ADQDevice (
            nds::Factory & factory,
            const std::string & deviceName,
            const nds::namedParameters_t & parameters )
```

ADQDevice class constructor.

Parameters

factory	contains an interface to the control system that requested a creation of the device.
deviceName	a name with which the device should be presented to the control system (root node).
parameters	here a serial number of the requested digitizer is passed to the device.

The documentation for this class was generated from the following files:

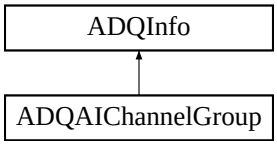
- ADQDevice.h
- ADQDevice.cpp

5.4 ADQInfo Class Reference

This class monitors informative parameters of the connected digitizer.

```
#include <ADQInfo.h>
```

Inheritance diagram for ADQInfo:



Public Member Functions

- ADQInfo (const std::string &name, nds::Node &parentNode, ADQInterface ∗&adqInterface, void ∗adqCtrlUnit)

  ADQInfo class constructor.
- void getProductName (timespec ∗pTimestamp, std::string ∗pValue)

  Gets the digitizer's product name.
- void getSerialNumber (timespec ∗pTimestamp, std::string ∗pValue)

  Gets the digitizer's serial number.
- void getProductID (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's product ID.
- void getADQType (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's type.
- void getCardOption (timespec ∗pTimestamp, std::string ∗pValue)

  Gets the digitizer's card option.
- void getTempLocal (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's PCB temperature.
- void getTempADCone (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's ADC1 temperature.
- void getTempADCtwo (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's ADC2 temperature.
- void getTempFPGA (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's FPGA temperature.
- void getTempDd (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's DCDC2A temperature.
- void getSampRate (timespec ∗pTimestamp, double ∗pValue)

  Gets the digitizer's base sample rate.
- void getSampRateDec (timespec ∗pTimestamp, double ∗pValue)

  Gets the digitizer's decimated sample rate.
- void getBytesPerSample (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the number of bytes needed to store each sample.
- void getBusAddr (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's bus address.
- void getBusType (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the digitizer's type of connection.
- void getPCIeLinkRate (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the PCIe/PXIe generation if the digitizer is connected over this interface.
- void getPCIeLinkWid (timespec ∗pTimestamp, int32_t ∗pValue)

  Gets the PCIe/PXIe width if the digitizer is connected over this interface.

Public Attributes

- nds::Port m_node

  ADQInfo class node that connects to the device.

Protected Attributes

- std::mutex m_adqDevMutex

  Lock guard.
- nds::PVDelegateIn< double > m_sampRateDecPV

  PV fpr sample rate with decimation.

5.4.1 Detailed Description

This class monitors informative parameters of the connected digitizer.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ADQInfo()

```
ADQInfo::ADQInfo (
            const std::string & name,
            nds::Node & parentNode,
            ADQInterface *& adqInterface,
            void * adqCtrlUnit )
```

ADQInfo class constructor.

Parameters

name	a name with which this class will register its child node.
parentNode	a name of a parent node to which this class' node is a child.
adqInterface	a pointer to the ADQ API interface created in the ADQInit class.
adqCtrlUnit	a pointer to the ADQ control unit that sets up and controls the ADQ devices.

5.4.3 Member Data Documentation

5.4.3.1 m_adqDevMutex

```
ADQInfo::m_adqDevMutex  [protected]
```

Lock guard.

It is used to protect ADQAPI library from simultaneous calling from different threads. For example, updating the temperatures (ADQInfo class) and data acquisition (ADQAIChannelGroup).

5.4.3.2 m_sampRateDecPV

```
ADQInfo::m_sampRateDecPV  [protected]
```

PV fpr sample rate with decimation.

Its value is updated when sample skip (ADQAIChannelGroup) is changed.

The documentation for this class was generated from the following files:

- ADQInfo.h
- ADQInfo.cpp

5.5 streamingHeader_t Struct Reference

This record header structure is used in Triggered streaming DAQ mode.

`#include <ADQAIChannelGroup.h>`

Public Attributes

- unsigned char recordStatus
  Record status.
- unsigned char userID
  User ID.
- unsigned char chan
  The name/number of the channel from which the record is acquired.
- unsigned char dataFormat
  Data format of the digitizer.
- unsigned int serialNumber
  Digitizer's serial number.
- unsigned int recordNumber
  The number of the passed record.
- unsigned int samplePeriod
  Sample period (1/rate).
- unsigned long long timeStamp
  Time when record was passed.
- unsigned long long recordStart
  Record start.
- unsigned int recordLength
  Number of samples in the record.
- unsigned int reserved
  Reserved placement.

5.5.1 Detailed Description

This record header structure is used in Triggered streaming DAQ mode.

The documentation for this struct was generated from the following file:

- ADQAIChannelGroup.h

Chapter 6

File Documentation

6.1 ADQAIChannel.h File Reference

This file defines ADQAIChannel class.

```
#include <nds3/nds.h>
```

Classes

- class ADQAIChannel

  This class handles channel specific parameters and pushes acquired data to appropriate data PVs.

6.1.1 Detailed Description

This file defines ADQAIChannel class.

6.2 ADQAIChannelGroup.h File Reference

This file defines ADQAIChannelGroup class and streamingHeader_t struct.

```
#include "ADQAIChannel.h"
#include "ADQDefinition.h"
#include "ADQInfo.h"
#include <mutex>
#include <atomic>
#include <nds3/nds.h>
```

Classes

- struct streamingHeader_t

  This record header structure is used in Triggered streaming DAQ mode.
- class ADQAIChannelGroup

  This class handles majority of parameters for correct setup of each data acquisition mode. Data acquisition is handled in this class. The state machine of the device is defined here. Each digitizer's channel gets a representation by calling ADQCHannel constructor for N amount of physical channels.

6.2.1 Detailed Description

This file defines ADQAIChannelGroup class and streamingHeader_t struct.

6.3 ADQDefinition.h File Reference

This file contains global objects (constants and macros). They are used in classes that include this file.

```
#include <iostream>
#include <nds3/nds.h>
#include <sstream>
#include <unistd.h>
```

Macros

- #define PINI true

  Enable PVs to process at the device initialization.
- #define CELSIUS_CONVERT 1 / 256

  Convert received temperature value into Celsius.
- #define TEMP_LOCAL 0

  Digitizer's device address of PCB.
- #define TEMP_ADC_ONE 1

  Digitizer's device address of ADC1.
- #define TEMP_ADC_TWO 2

  Digitizer's device address of ADC2.
- #define TEMP_FPGA 3

  Digitizer's device address of FPGA.
- #define TEMP_DIOD 4

  Digitizer's device address of DCDC2A.
- #define DATA_MAX_ELEMENTS (4 ∗ 1024 ∗ 1024)

  Maximum number of elements for data PV.
- #define BUFFERSIZE_ADQ14 (512 ∗ 1024)

  Buffersize for data acquisition (ADQ14).
- #define BUFFERSIZE_ADQ7 (256 ∗ 1024)

  Buffersize for data acquisition (ADQ7).
- #define CHANNEL_COUNT_MAX 8

  Maximum allowed amount of channels.
- #define EXTERN_TRIG_COUNT 1

Amount of inputs for external triggering in each device.

- #define STRING_ENUM 32

  Number of elements for some digitizer's information PVs.

- #define GROUP_CHAN_DEVICE "-ChGrp"

  Append the string to ADQAIChannelGroup node name.

- #define INFO_DEVICE "-Info"

  Append the string to ADQDevice node name.

- #define SLEEP(interval) usleep(1000 ∗ interval)

  Macro for sleeping for 1000∗interval microseconds.

- #define MIN(a, b) ((a) > (b) ? (b) : (a))

  A macro that returns the minimum of a and b.

- #define UNUSED(x) (void)x

  Macro for busying unused parameters in methods.

- #define ADQNDS_MSG_INFOLOG_PV(text)

  Macro for pushing log messages to PV. Used in ADQAIChannelGroup methods.

- #define ADQNDS_MSG_ERRLOG_PV_GOTO_FINISH(status, text)

  Macro for informing the user about occurred major failures and stopping data acquisition. Used in ADQAIChannelGroup methods.

- #define ADQNDS_MSG_WARNLOG_PV(status, text)

  Macro for warning information in case of minor failures. Used in ADQAIChannelGroup methods.

6.3.1 Detailed Description

This file contains global objects (constants and macros). They are used in classes that include this file.

6.3.2 Macro Definition Documentation

6.3.2.1 ADQNDS_MSG_ERRLOG_PV_GOTO_FINISH

```
#define ADQNDS_MSG_ERRLOG_PV_GOTO_FINISH(
              status,
              text )
```

Value:

```
do                                                                  \
    {                                                               \
        if (!status)                                                \
        {                                                           \
            struct timespec now = { 0, 0 };                         \
            clock_gettime(CLOCK_REALTIME, &now);                    \
            m_logMsgPV.push(now, std::string(text));                \
            ndsErrorStream(m_node) « std::string(text) « std::endl; \
            goto finish;                                            \
        }                                                           \
    } while (0)
```

Macro for informing the user about occurred major failures and stopping data acquisition. Used in ADQAIChannelGroup methods.

Parameters

status	status of the function that calls this macro.
text	input information message.

6.3.2.2 ADQNDS_MSG_INFOLOG_PV

```
#define ADQNDS_MSG_INFOLOG_PV(
              text )
```

Value:

```
do                                                                  \
    {                                                                \
        struct timespec now = { 0, 0 };                              \
        clock_gettime(CLOCK_REALTIME, &now);                         \
        m_logMsgPV.push(now, std::string(text));                     \
        ndsInfoStream(m_node) « std::string(text) « std::endl; \
    } while (0)
```

Macro for pushing log messages to PV. Used in ADQAIChannelGroup methods.

Parameters

text	input information message.

6.3.2.3 ADQNDS_MSG_WARNLOG_PV

```
#define ADQNDS_MSG_WARNLOG_PV(
              status,
              text )
```

Value:

```
do                                                                      \
    {                                                                    \
        if (!status)                                                     \
        {                                                                \
            struct timespec now = { 0, 0 };                              \
            clock_gettime(CLOCK_REALTIME, &now);                         \
            m_logMsgPV.push(now, std::string(text));                     \
            ndsWarningStream(m_node) « std::string(text) « std::endl; \
        }                                                                \
    } while (0)
```

Macro for warning information in case of minor failures. Used in ADQAIChannelGroup methods.

Parameters

status	status of the function that calls this macro.
text	input information message.

6.3.2.4 SLEEP

```
#define SLEEP(
              interval ) usleep(1000 * interval)
```

Macro for sleeping for *1000∗interval* microseconds.

Parameters

interval	value that will be multiplied by 1000 microseconds.

6.4 ADQDevice.h File Reference

This file defines ADQDevice class that creates a device.

```
#include "ADQAIChannel.h"
#include "ADQAIChannelGroup.h"
#include "ADQDefinition.h"
#include "ADQInfo.h"
#include <ADQAPI.h>
#include <mutex>
#include <nds3/nds.h>
```

Classes

- class ADQDevice

  This class creates a device that communicates with a digitizer. ADQ Control Unit is handled by this class. The pointer to ADQAPI interface is also created here.

6.4.1 Detailed Description

This file defines ADQDevice class that creates a device.

6.5 ADQInfo.h File Reference

This file defines ADQInfo class.

```
#include <mutex>
#include <nds3/nds.h>
```

Classes

- class ADQInfo

  This class monitors informative parameters of the connected digitizer.

6.5.1 Detailed Description

This file defines ADQInfo class.

Index