ControlAPI.h File Reference

```
#include <afxwin.h>
#include <fstream>
#include <cstdint>
```

Go to the source code of this file.

Macros

#define	**_AFXDLL**
#define	**API_EXPORT**
#define	**API_EXPORT_CLASS**
#define	**API_EXPORT_CLASS**
#define	**ERROR_CODE_TYPE** bool
#define	**CLA_FN**(name)
#define	**CLA_FNDEF**(name)

Functions

ERROR_CODE_TYPE	**AddErrorMessage** (const std::string &error_message)
API_EXPORT ERROR_CODE_TYPE CLA_FN	**Create** (bool InitializeAfx, bool InitializeAfxSocket) Initializes the ControlAPI. If you use a bare function C API, then this function must be called before using any other functions in the API. If you didn't load MFC yet (which is the case if you use Qt), then set.
API_EXPORT void **CLA_FN**	**Cleanup** () This function must be called when you are finished using the API. The class wrapped version of this API calls this function in its destructor.
API_EXPORT bool **CLA_FN**	**DidErrorOccur** () Check if an error has occurred since the last call to GetLastError.
API_EXPORT const char ***CLA_FN**	**GetLastError** () returns the last error messages
API_EXPORT void **CLA_FN**	**Configure** (bool _DisplayErrors) Configures if the ControlAPI should display error messages.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**LoadFromJSONFile** (const char *filename) Loads a configuration from a JSON file. Before you can use the control system, you either must read a json

configuration file, or define the devices in the API.

API_EXPORT ERROR_CODE_TYPE CLA_FN	**Initialize** () After Configuring the hardware, e.g. by loading a json configuration file, this function must be called to initialize the hardware.
API_EXPORT void **CLA_FN**	**SwitchDebugMode** (bool OnOff, const char *FileName) Switches Debug mode on. In Debug mode, the sequence of each FPGA sequencer is written to a human readable ASCII file before being sent to the FPGA. In addition, the FPGA sequencers display more information on their USB-UART port, being slowed down a bit by that.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**IsReady** () Checks if the ControlAPI is ready to be used.
API_EXPORT void **CLA_FN**	**StartAssemblingSequence** () Starts assembling a sequence. Clears any previous sequence. Must be called before adding commands to the sequence.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetRegister** (const unsigned int &Sequencer, const unsigned int &Address, const unsigned int &SubAddress, const uint8_t *Data, const unsigned long &DataLength_in_bit, const uint8_t &StartBit=0) This function sets a register for a device on the sequencer. What this means depends on the device type. This function gives us an easy way to add new functionality to a device without having to programm new DLL functions. SetRegister maps to the register map given in the device's datasheet. Note: this is not currently implemented in the API.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetValue** (const unsigned int &Sequencer, const unsigned int &Address, const unsigned int &SubAddress, const uint8_t *Data, const unsigned long &DataLength_in_bit, const uint8_t &StartBit=0) This function sets a value for a device on the sequencer. What this means depends on the device type. This function gives us an easy way to add new functionality to a device without having to programm new DLL functions. In contrast to SetRegister, SetValue can execute more complex operations, such as calculating a DDS frequency tuning word from the given frequency and then programming that. There can be several SetValue functions for the same SetRegister function. There can be SetValue functions that

	set some parameter, or that trigger some action, not even necessarily related to the registers of the device.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetValueSerialDevice** (const unsigned int &Sequencer, const unsigned int &Address, const unsigned int &SubAddress, const uint8_t *Data, const unsigned long &DataLength_in_bit, const uint8_t &StartBit=0) As SetValue, but for serial devices.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetRegisterSerialDevice** (const unsigned int &Sequencer, const unsigned int &Address, const unsigned int &SubAddress, const uint8_t *Data, const unsigned long &DataLength_in_bit, const uint8_t &StartBit=0) As SetRegister, but for serial devices.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**Wait_ms** (double time_in_ms) Wait for a given time in ms.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**GetTime_ms** (double &time_in_ms) Get the current time in the currently assembled sequence ms.
API_EXPORT ERROR_CODE_TYPE CLA_FNDEF	**GetTimeOfSequencer_ms** (const unsigned int &Sequencer, double &time_in_ms) Get the current time of a specific sequencer in the currently assembled sequence ms.
API_EXPORT ERROR_CODE_TYPE CLA_FNDEF	**GetTimeDebtOfSequencer_ms** (const unsigned int &Sequencer, double &time_debt_in_ms) Get the time debtof a specific sequencer in the currently assembled sequence ms.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetVoltage** (const unsigned int &Sequencer, const unsigned int &Address, double Voltage) Sets the voltage of an analog output device.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetDigitalOutput** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t BitNr, bool OnOff) Sets a digital output to high or low.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetStartFrequency** (const unsigned int &Sequencer, const unsigned int &Address, double Frequency) Sets the start frequency of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetStopFrequency** (const unsigned int &Sequencer, const unsigned int &Address, double Frequency) Sets the stop frequency of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetModulationFrequency** (const unsigned int &Sequencer, const unsigned int &Address, double Frequency)

Sets the modulation frequency of a DDS (for now a AD9854 DDS).

API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetPower** (const unsigned int &Sequencer, const unsigned int &Address, double Power) Sets the power of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetAttenuation** (const unsigned int &Sequencer, const unsigned int &Address, double Attenuation) Sets the attenuation of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetStartFrequencyTuningWord** (const unsigned int &Sequencer, const unsigned int &Address, uint64_t FrequencyTuningWord) Sets the start frequency tuning word of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetStopFrequencyTuningWord** (const unsigned int &Sequencer, const unsigned int &Address, uint64_t FrequencyTuningWord) Sets the stop frequency tuning word of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFSKMode** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t mode) Sets the FKS mode of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetRampRateClock** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t rate) Sets the ramp rate clock of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetClearACC1** (const unsigned int &Sequencer, const unsigned int &Address, bool OnOff) Clears the ACC1 bit of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetTriangleBit** (const unsigned int &Sequencer, const unsigned int &Address, bool OnOff) Sets the triangle bit of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFSKBit** (const unsigned int &Sequencer, const unsigned int &Address, bool OnOff) Sets the FSK bit of a DDS (for now a AD9854 DDS).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFrequency** (const unsigned int &Sequencer, const unsigned int &Address, double Frequency) Sets the frequency of a DDS.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFrequencyTuningWord** (const unsigned int &Sequencer, const unsigned int &Address, uint64_t FrequencyTuningWord)

Sets the frequency tuning word of a DDS.

API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFrequencyOfChannel** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t channel, double Frequency) Sets the frequency of a multi channel DDS (for now the AD9958).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetFrequencyTuningWordOfChannel** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t channel, uint64_t FrequencyTuningWord) Sets the frequency tuning word of a multi channel DDS (for now the AD9958).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetPhaseOfChannel** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t channel, double Phase) Sets the phase of a multi channel DDS (for now the AD9958).
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetPowerOfChannel** (const unsigned int &Sequencer, const unsigned int &Address, uint8_t channel, double Power) Sets the phase of a multi channel DDS (for now the AD9958).
API_EXPORT void **CLA_FN**	**ExecuteSequence** () Executes the sequence that was previously assembled.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**GetSequenceExecutionStatus** (bool &running, unsigned long long &DataPointsWritten) Get the current status of the sequence execution.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**WaitTillEndOfSequenceThenGetInputData** (uint8_t *&buffer, unsigned long &buffer_length, unsigned long &EndTimeOfCycle, double timeout_in_s) Waits until the sequence is finished, then gets the input data from the FPGA sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SetTimeDebtGuard_in_ms** (const double &MaxTimeDebt_in_ms) Sets a guard time. If sequencer commands make the time advance more than the guard time beyond what's allowed by Wait_ms, an error will be recorded (check with **DidErrorOccur()** if that happened) or thrown.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerStartAnalogInAcquisition** (const unsigned int &Sequencer, const uint8_t &ChannelNumber, const uint32_t &NumberOfDataPoints, const double &DelayBetweenDataPoints_in_ms) Start analog acquisition on the specified channel. This function places a commandto for the FPGA in the

sequencer buffer. The analog in acquisition will start when this command is executed by the FPGA. Only one analog in acquisition can hapen at each moment. A new one can start right after the previous one is finished. The data will be returned at the end of a sequence when using CLA_WaitTillEndOfSequenceThenGetInputData.

API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerWriteInputMemory** (const unsigned int &Sequencer, unsigned long input_buf_mem_data, bool write_next_address=1, unsigned long input_buf_mem_address=0) Writes a value to the input memory of the sequencer. This is useful to mark the start or end of a data acquisition, or to mark the type of experimental run in the full fledged version of the ControlAPI, which can cycle sequences automatically in the background. To execute this function, a command for the FPGA must be placed in the sequencer buffer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerWriteSystemTimeToInputMemory** (const unsigned int &Sequencer) Writes the current FPGA clock ticks to the input memory of the sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerSwitchDebugLED** (const unsigned int &Sequencer, unsigned int OnOff) Switches the debug LED of the FPGA on or off.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerIgnoreTCPIP** (const unsigned int &Sequencer, bool OnOff) Switches the sequencer to ignore TCP/IP commands. This is useful to prevent the sequencer from being interrupted by TCP/IP commands while executing a timing critical task, such as transferring input data from the FPGA BRAM to the DDR. This can be useful if lots of data is acquired at a high rate. Lot's of meaning: more than the size of the BRAM input buffer. How much that is depends how you configured the Vivado project that generates the FPGA sequencer firmware.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**SequencerAddMarker** (const unsigned int &Sequencer, unsigned char marker) Places a marker in the sequencer buffer. The FPGA SOM's CPU will see this marker and can react to it, e.g. write it to the USB-UART for debugging.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceSequencer** (unsigned int id, const char *type, const char *ip, unsigned int port, bool master, unsigned int startDelay, double clockFrequency, unsigned long

	FPGAClockToBusClockRatio, bool useExternalClock, bool useStrobeGenerator, bool connect) Add a device sequencer to the list.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceAnalogOut16bit** (unsigned int sequencer, unsigned int startAddress, unsigned int numberChannels, bool signedValue, double minVoltage, double maxVoltage) Add a 16 bit analog output device to the sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceDigitalOut** (unsigned int sequencer, unsigned int address, unsigned int numberChannels) Add a digital output device to the sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceAD9854** (unsigned int sequencer, unsigned int address, unsigned int version, double externalClockFrequency, uint8_t PLLReferenceMultiplier, unsigned int frequencyMultiplier) Add a AD9854 device to the sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceAD9858** (unsigned int sequencer, unsigned int address, double externalClockFrequency, unsigned int frequencyMultiplier) Add a AD9858 device to the sequencer.
API_EXPORT ERROR_CODE_TYPE CLA_FN	**AddDeviceAnalogIn12bit** (unsigned int sequencer, unsigned int chipSelect, bool signedValue, double minVoltage, double maxVoltage) Add a 12 bit analog input device to the sequencer.

Variables

constexpr int	**MaxLastError** = 10
std::string	**LastError** [**MaxLastError**]
int	**LastErrorIndex**
bool	**ErrorListWrapAround**

Macro Definition Documentation

◆ _AFXDLL

```
#define _AFXDLL
```

◆ API_EXPORT

```
#define API_EXPORT
```

◆ API_EXPORT_CLASS [1/2]

```
#define API_EXPORT_CLASS
```

◆ API_EXPORT_CLASS [2/2]

```
#define API_EXPORT_CLASS
```

◆ CLA_FN

```
#define CLA_FN ( name )
```

Value:

```
CLA_##name
```

◆ CLA_FNDEF

```
#define CLA_FNDEF ( name )
```

Value:

```
CLA_##name
```

◆ ERROR_CODE_TYPE

```
#define ERROR_CODE_TYPE   bool
```

Function Documentation

◆ AddDeviceAD9854()

API_EXPORT ERROR_CODE_TYPE CLA_FN AddDeviceAD9854 (unsigned int sequencer,
unsigned int address,
unsigned int version,
double externalClockFrequency,
uint8_t PLLReferenceMultiplier,
unsigned int frequencyMultiplier)

Add a AD9854 device to the sequencer.

Parameters

sequencer	the sequencer to use.
address	the address of the device to add.
version	the version of the device to add.
externalClockFrequency	the external clock frequency of the device to add.
PLLReferenceMultiplier	the PLL reference multiplier of the device to add.
frequencyMultiplier	the frequency multiplier of the device to add.

Returns

◆ AddDeviceAD9858()

API_EXPORT ERROR_CODE_TYPE CLA_FN AddDeviceAD9858 (unsigned int sequencer,
unsigned int address,
double externalClockFrequency,
unsigned int frequencyMultiplier)

Add a AD9858 device to the sequencer.

Parameters

sequencer	the sequencer to use.
address	the address of the device to add.
externalClockFrequency	the external clock frequency of the device to add.
frequencyMultiplier	the frequency multiplier of the device to add.

Returns

◆ AddDeviceAnalogIn12bit()

API_EXPORT ERROR_CODE_TYPE CLA_FN AddDeviceAnalogIn12bit (unsigned int sequencer,
unsigned int chipSelect,
bool signedValue,
double minVoltage,
double maxVoltage)

Add a 12 bit analog input device to the sequencer.

Parameters

sequencer	the sequencer to use.
chipSelect	the chip select of the device to add.
signedValue	true if the device is signed, false if it is unsigned.
minVoltage	the minimum voltage of the device to add.
maxVoltage	the maximum voltage of the device to add.

Returns

◆ AddDeviceAnalogOut16bit()

API_EXPORT ERROR_CODE_TYPE CLA_FN AddDeviceAnalogOut16bit (unsigned int sequencer,
unsigned int startAddress,
unsigned int numberChannels,
bool signedValue,
double minVoltage,
double maxVoltage)

Add a 16 bit analog output device to the sequencer.

Parameters

sequencer	the sequencer to use.
startAddress	the start address of the device to add.
numberChannels	the number of channels of the device to add.
signedValue	true if the device is signed, false if it is unsigned.
minVoltage	the minimum voltage of the device to add.
maxVoltage	the maximum voltage of the device to add.

Returns

◆ AddDeviceDigitalOut()

API_EXPORT ERROR_CODE_TYPE CLA_FN AddDeviceDigitalOut (unsigned int sequencer,
unsigned int address,
unsigned int numberChannels)

Add a digital output device to the sequencer.

Parameters

sequencer	the sequencer to use.
address	the address of the device to add.
numberChannels	the number of channels of the device to add.

Returns

◆ AddDeviceSequencer()

API_EXPORT ERROR_CODE_TYPE CLA_FN

```
AddDeviceSequencer ( unsigned int   id,
                     const char *   type,
                     const char *   ip,
                     unsigned int   port,
                     bool           master,
                     unsigned int   startDelay,
                     double         clockFrequency,
                     unsigned long  FPGAClockToBusClockRatio,
                     bool           useExternalClock,
                     bool           useStrobeGenerator,
                     bool           connect )
```

Add a device sequencer to the list.

Parameters

id	the id of the device sequencer to add (this is the number that all other commands are using when sending data to a device on this sequencer).
type	the type of the device sequencer to add.
ip	the ip address of the device sequencer to add.
port	the port of the device sequencer to add.
master	true if the device sequencer is a master, false if it is a slave.
startDelay	the start delay of the device sequencer to add.
clockFrequency	the clock frequency of the device sequencer to add.
FPGAClockToBusClockRatio	the FPGA clock to bus clock ratio of the device sequencer to add.
useExternalClock	true if the device sequencer should use an external clock, false if it should use the internal clock.
useStrobeGenerator	true if the device sequencer should use a strobe generator, false if it should not.
connect	true if the device sequencer should connect to the device, false if it should not.

Returns

◆ AddErrorMessage()

ERROR_CODE_TYPE AddErrorMessage (const std::string & error_message)

◆ Cleanup()

API_EXPORT void **CLA_FN** Cleanup ( )

This function must be called when you are finished using the API. The class wrapped version of this API calls this function in its destructor.

Parameters

**
**

Returns

◆ Configure()

API_EXPORT void **CLA_FN** Configure (bool _DisplayErrors)

Configures if the ControlAPI should display error messages.

Parameters

_DisplayErrors true to display error messages, false to suppress them.

Returns

◆ Create()

API_EXPORT ERROR_CODE_TYPE CLA_FN Create (bool InitializeAfx,
bool InitializeAfxSocket)

Initializes the ControlAPI. If you use a bare function C API, then this function must be called before using any other functions in the API. If you didn't load MFC yet (which is the case if you use Qt), then set.

Parameters

InitializeAfx to true to initialize MFC core

InitializeAfxSocket to true to initialize MFC socket layer The class wrapped version of this API calls this function in its constructor.

**
**

Returns

◆ DidErrorOccur()

API_EXPORT bool **CLA_FN** DidErrorOccur ( )

Check if an error has occurred since the last call to GetLastError.

Parameters

**
**

Returns

true if an error has occurred since the last call to GetLastError

◆ ExecuteSequence()

API_EXPORT void **CLA_FN** ExecuteSequence ( )

Executes the sequence that was previously assembled.

Parameters

**
**

Returns

◆ GetLastError()

API_EXPORT const char ***CLA_FN** GetLastError ( )

returns the last error messages

Parameters

**
**

Returns

◆ GetSequenceExecutionStatus()

API_EXPORT ERROR_CODE_TYPE CLA_FN
GetSequenceExecutionStatus (bool & running,
unsigned long long & DataPointsWritten)

Get the current status of the sequence execution.

Parameters

running	returns true if the sequence is running, false if it is not.
DataPointsWritten	returns number of data points that were already written out by the FPGA sequencer(s).

◆ GetTime_ms()

API_EXPORT ERROR_CODE_TYPE CLA_FN GetTime_ms (double & time_in_ms)

Get the current time in the currently assembled sequence ms.

Parameters

time_in_ms	the current time in ms.

Returns

◆ GetTimeDebtOfSequencer_ms()

API_EXPORT ERROR_CODE_TYPE CLA_FNDEF
GetTimeDebtOfSequencer_ms (const unsigned int & Sequencer,
double & time_debt_in_ms)

Get the time debtof a specific sequencer in the currently assembled sequence ms.

Parameters

Sequencer	the sequencer to use.
time_debt_in_ms	the current time in ms.

Returns

◆ GetTimeOfSequencer_ms()

API_EXPORT ERROR_CODE_TYPE CLA_FNDEF
GetTimeOfSequencer_ms (const unsigned int & Sequencer,
double & time_in_ms)

Get the current time of a specific sequencer in the currently assembled sequence ms.

Parameters

Sequencer the sequencer to use.

time_in_ms the current time in ms.

Returns

◆ Initialize()

API_EXPORT ERROR_CODE_TYPE CLA_FN Initialize ( )

After Configuring the hardware, e.g. by loading a json configuration file, this function must be called to initialize the hardware.

Parameters

**
**

Returns

◆ IsReady()

API_EXPORT ERROR_CODE_TYPE CLA_FN IsReady ( )

Checks if the ControlAPI is ready to be used.

Parameters

IsReady true if the ControlAPI is ready to be used, false otherwise.

◆ LoadFromJSONFile()

API_EXPORT ERROR_CODE_TYPE CLA_FN LoadFromJSONFile (const char * filename)

Loads a configuration from a JSON file. Before you can use the control system, you either must read a json configuration file, or define the devices in the API.

Parameters

filename the name of the json file to load.

◆ SequencerAddMarker()

API_EXPORT ERROR_CODE_TYPE CLA_FN SequencerAddMarker (const unsigned int & Sequencer, unsigned char marker)

Places a marker in the sequencer buffer. The FPGA SOM's CPU will see this marker and can react to it, e.g. write it to the USB-UART for debugging.

Parameters

**
**

Returns

◆ SequencerIgnoreTCPIP()

API_EXPORT ERROR_CODE_TYPE CLA_FN SequencerIgnoreTCPIP (const unsigned int & Sequencer, bool OnOff)

Switches the sequencer to ignore TCP/IP commands. This is useful to prevent the sequencer from being interrupted by TCP/IP commands while executing a timing critical task, such as transferring input data from the FPGA BRAM to the DDR. This can be useful if lots of data is acquired at a high rate. Lot's of meaning: more than the size of the BRAM input buffer. How much that is depends how you configured the Vivado project that generates the FPGA sequencer firmware.

Parameters

**
**

Returns

◆ SequencerStartAnalogInAcquisition()

API_EXPORT ERROR_CODE_TYPE CLA_FN
SequencerStartAnalogInAcquisition (const unsigned int & Sequencer,
const uint8_t & ChannelNumber,
const uint32_t & NumberOfDataPoints,
const double & DelayBetweenDataPoints_in_ms)

Start analog acquisition on the specified channel. This function places a commandto for the FPGA in the sequencer buffer. The analog in acquisition will start when this command is executed by the FPGA. Only one analog in acquisition can hapen at each moment. A new one can start right after the previous one is finished. The data will be returned at the end of a sequence when using CLA_WaitTillEndOfSequenceThenGetInputData.

Parameters

Sequencer	the sequencer to use.
ChannelNumber	the channel number to use.
NumberOfDataPoints	the number of data points to acquire.
DelayBetweenDataPoints_in_ms	the delay between data points in ms.

◆ SequencerSwitchDebugLED()

API_EXPORT ERROR_CODE_TYPE CLA_FN SequencerSwitchDebugLED (const unsigned int & Sequencer,
unsigned int OnOff)

Switches the debug LED of the FPGA on or off.

Parameters

Squencer	the sequencer to use.
OnOff	true to switch on the LED, false to switch it off.

Returns

◆ SequencerWriteInputMemory()

API_EXPORT ERROR_CODE_TYPE CLA_FN

SequencerWriteInputMemory (const unsigned int & Sequencer,
unsigned long input_buf_mem_data,
bool write_next_address = 1,
unsigned long input_buf_mem_address = 0)

Writes a value to the input memory of the sequencer. This is useful to mark the start or end of a data acquisition, or to mark the type of experimental run in the full fledged version of the ControlAPI, which can cycle sequences automatically in the background. To execute this function, a command for the FPGA must be placed in the sequencer buffer.

Parameters

Sequencer	the sequencer to use.
input_buf_mem_data	the data to write to the input memory.
write_next_address	true to write the next address, false to write the address given.
input_buf_mem_address	the address to write to, if write_next_address is false.

◆ SequencerWriteSystemTimeToInputMemory()

API_EXPORT ERROR_CODE_TYPE CLA_FN

SequencerWriteSystemTimeToInputMemory (const unsigned int & Sequencer)

Writes the current FPGA clock ticks to the input memory of the sequencer.

Parameters

Sequencer	the sequencer to use.

◆ SetAttenuation()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetAttenuation (const unsigned int & Sequencer,
const unsigned int & Address,
double Attenuation)

Sets the attenuation of a DDS (for now a AD9854 DDS).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the attenuation for.

Attenuation the attenuation to set in dB (-xxx ... 0).

Returns

◆ SetClearACC1()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetClearACC1 (const unsigned int & Sequencer,
const unsigned int & Address,
bool OnOff)

Clears the ACC1 bit of a DDS (for now a AD9854 DDS).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the ACC1 bit for.

OnOff true to set the ACC1 bit, false to clear it.

Returns

◆ SetDigitalOutput()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetDigitalOutput (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t BitNr,
bool OnOff)

Sets a digital output to high or low.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the digital output for.
BitNr	the bit number of the digital output to set.
OnOff	true to set the output to high, false to set it to low.

◆ SetFrequency()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetFrequency (const unsigned int & Sequencer,
const unsigned int & Address,
double Frequency)

Sets the frequency of a DDS.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the frequency for.
Frequency	the frequency to set.

Returns

◆ SetFrequencyOfChannel()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetFrequencyOfChannel (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t channel,
double Frequency)

Sets the frequency of a multi channel DDS (for now the AD9958).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the frequency for.
channel	the channel number to set the frequency for.
Frequency	the frequency to set.

Returns

◆ SetFrequencyTuningWord()

API_EXPORT ERROR_CODE_TYPE CLA_FN
SetFrequencyTuningWord (const unsigned int & Sequencer,
const unsigned int & Address,
uint64_t FrequencyTuningWord)

Sets the frequency tuning word of a DDS.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the frequency tuning word for.
FrequencyTuningWord	the frequency tuning word to set.

Returns

◆ SetFrequencyTuningWordOfChannel()

API_EXPORT ERROR_CODE_TYPE CLA_FN
SetFrequencyTuningWordOfChannel (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t channel,
uint64_t FrequencyTuningWord)

Sets the frequency tuning word of a multi channel DDS (for now the AD9958).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the frequency tuning word for.
channel	the channel number to set the frequency tuning word for.
FrequencyTuningWord	the frequency tuning word to set.

Returns

◆ SetFSKBit()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetFSKBit (const unsigned int & Sequencer,
const unsigned int & Address,
bool OnOff)

Sets the FSK bit of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the FSK bit for.
OnOff	true to set the FSK bit, false to clear it.

Returns

◆ SetFSKMode()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetFSKMode (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t mode)

Sets the FKS mode of a DDS (for now a AD9854 DDS).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the FKS mode for.

mode the FKS mode to set (0..4).

Returns

◆ SetModulationFrequency()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetModulationFrequency (const unsigned int & Sequencer,
const unsigned int & Address,
double Frequency)

Sets the modulation frequency of a DDS (for now a AD9854 DDS).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the modulation frequency for.

Frequency the frequency to set.

Returns

◆ SetPhaseOfChannel()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetPhaseOfChannel (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t channel,
double Phase)

Sets the phase of a multi channel DDS (for now the AD9958).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the phase for.

channel the channel number to set the phase for.

Phase the phase to set.

Returns

◆ SetPower()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetPower (const unsigned int & Sequencer,
const unsigned int & Address,
double Power)

Sets the power of a DDS (for now a AD9854 DDS).

Parameters

Sequencer the sequencer to use.

Address the address of the device to set the power for.

Power the power to set in % (0...100).

Returns

◆ SetPowerOfChannel()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetPowerOfChannel (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t channel,
double Power)

Sets the phase of a multi channel DDS (for now the AD9958).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the phase for.
channel	the channel number to set the phase for.
Phase	the phase to set.

Returns

◆ SetRampRateClock()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetRampRateClock (const unsigned int & Sequencer,
const unsigned int & Address,
uint8_t rate)

Sets the ramp rate clock of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the ramp rate clock for.
rate	the ramp rate clock to set (1...1048576).

Returns

◆ SetRegister()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetRegister (const unsigned int & Sequencer,
const unsigned int & Address,
const unsigned int & SubAddress,
const uint8_t * Data,
const unsigned long & DataLength_in_bit,
const uint8_t & StartBit = 0)

This function sets a register for a device on the sequencer. What this means depends on the device type. This function gives us an easy way to add new functionality to a device without having to programm new DLL functions. SetRegister maps to the register map given in the device's datasheet. Note: this is not currently implemented in the API.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the value for.
SubAddress	the subaddress of the device to set the value for.
Data	the data to set.
DataLength_in_bit	the length of the data in bits.
StartBit	the start bit of the data to set.
** **	

Returns

◆ SetRegisterSerialDevice()

API_EXPORT ERROR_CODE_TYPE CLA_FN
SetRegisterSerialDevice (const unsigned int & Sequencer,
const unsigned int & Address,
const unsigned int & SubAddress,
const uint8_t * Data,
const unsigned long & DataLength_in_bit,
const uint8_t & StartBit = 0)

As SetRegister, but for serial devices.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the value for.
SubAddress	the subaddress of the device to set the value for.
Data	the data to set.
DataLength_in_bit	the length of the data in bits.
StartBit	the start bit of the data to set.

◆ SetStartFrequency()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetStartFrequency (const unsigned int & Sequencer,
const unsigned int & Address,
double Frequency)

Sets the start frequency of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the start frequency for.
Frequency	the frequency to set.

Returns

◆ SetStartFrequencyTuningWord()

API_EXPORT ERROR_CODE_TYPE CLA_FN
SetStartFrequencyTuningWord (const unsigned int & Sequencer,
const unsigned int & Address,
uint64_t FrequencyTuningWord)

Sets the start frequency tuning word of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the start frequency tuning word for.
FrequencyTuningWord	the frequency tuning word to set.

Returns

◆ SetStopFrequency()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetStopFrequency (const unsigned int & Sequencer,
const unsigned int & Address,
double Frequency)

Sets the stop frequency of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the stop frequency for.
Frequency	the frequency to set.

Returns

◆ SetStopFrequencyTuningWord()

API_EXPORT ERROR_CODE_TYPE CLA_FN

SetStopFrequencyTuningWord (const unsigned int & Sequencer,
const unsigned int & Address,
uint64_t FrequencyTuningWord)

Sets the stop frequency tuning word of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the stop frequency tuning word for.
FrequencyTuningWord	the frequency tuning word to set.

Returns

◆ SetTimeDebtGuard_in_ms()

API_EXPORT ERROR_CODE_TYPE CLA_FN

SetTimeDebtGuard_in_ms (const double & MaxTimeDebt_in_ms)

Sets a guard time. If sequencer commands make the time advance more than the guard time beyond what's allowed by Wait_ms, an error will be recorded (check with **DidErrorOccur()** if that happened) or thrown.

Parameters

MaxTimeDebt_in_ms	maximum time debt in ms. If the sequencer commands make the time advance more than this, the sequencer will stop and wait for the next command.

Returns

◆ SetTriangleBit()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetTriangleBit (const unsigned int & Sequencer,
const unsigned int & Address,
bool OnOff)

Sets the triangle bit of a DDS (for now a AD9854 DDS).

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the triangle bit for.
OnOff	true to set the triangle bit, false to clear it.

Returns

◆ SetValue()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetValue (const unsigned int & Sequencer,
const unsigned int & Address,
const unsigned int & SubAddress,
const uint8_t * Data,
const unsigned long & DataLength_in_bit,
const uint8_t & StartBit = 0)

This function sets a value for a device on the sequencer. What this means depends on the device type. This function gives us an easy way to add new functionality to a device without having to programm new DLL functions. In contrast to SetRegister, SetValue can execute more complex operations, such as calculating a DDS frequency tuning word from the given frequency and then programming that. There can be several SetValue functions for the same SetRegister function. There can be SetValue functions that set some parameter, or that trigger some action, not even necessarily related to the registers of the device.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the value for.
SubAddress	the subaddress of the device to set the value for.
Data	the data to set.
DataLength_in_bit	the length of the data in bits.
StartBit	the start bit of the data to set.
** **	

Returns

◆ SetValueSerialDevice()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetValueSerialDevice (const unsigned int & Sequencer,
const unsigned int & Address,
const unsigned int & SubAddress,
const uint8_t * Data,
const unsigned long & DataLength_in_bit,
const uint8_t & StartBit = 0)

As SetValue, but for serial devices.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the value for.
SubAddress	the subaddress of the device to set the value for.
Data	the data to set.
DataLength_in_bit	the length of the data in bits.
StartBit	the start bit of the data to set.

Returns

◆ SetVoltage()

API_EXPORT ERROR_CODE_TYPE CLA_FN SetVoltage (const unsigned int & Sequencer,
const unsigned int & Address,
double Voltage)

Sets the voltage of an analog output device.

Parameters

Sequencer	the sequencer to use.
Address	the address of the device to set the voltage for.
Voltage	the voltage to set.

◆ StartAssemblingSequence()

API_EXPORT void **CLA_FN** StartAssemblingSequence ( )

Starts assembling a sequence. Clears any previous sequence. Must be called before adding commands to the sequence.

Parameters

**
**

Returns

◆ SwitchDebugMode()

API_EXPORT void **CLA_FN** SwitchDebugMode (bool OnOff,
const char * FileName)

Switches Debug mode on. In Debug mode, the sequence of each FPGA sequencer is written to a human readable ASCII file before being sent to the FPGA. In addition, the FPGA sequencers display more information on their USB-UART port, being slowed down a bit by that.

Parameters

OnOff true to switch on debug mode, false to switch it off.

FileName the name of the file to write the debug information to without filename extension.

◆ Wait_ms()

API_EXPORT ERROR_CODE_TYPE CLA_FN Wait_ms (double time_in_ms)

Wait for a given time in ms.

Parameters

time_in_ms the time to wait in ms.

Returns

◆ WaitTillEndOfSequenceThenGetInputData()

API_EXPORT ERROR_CODE_TYPE CLA_FN

WaitTillEndOfSequenceThenGetInputData (uint8_t *& buffer,
unsigned long & buffer_length,
unsigned long & EndTimeOfCycle,
double timeout_in_s)

Waits until the sequence is finished, then gets the input data from the FPGA sequencer.

Parameters

buffer	pointer to the input data buffer. Don't delete this buffer, it is managed by the API.
buffer_length	length of the input data buffer in bytes.
EndTimeOfCycle	returns the end time of the cycle in ms.
timeout_in_s	timeout in seconds. If the sequence is not finished within this time, the function returns false or throws an exception (depending on mode selected mode when compiling API).

Variable Documentation

◆ ErrorListWrapAround

bool ErrorListWrapAround extern

◆ LastError

std::string LastError[**MaxLastError**] extern

◆ LastErrorIndex

int LastErrorIndex extern

◆ MaxLastError

int MaxLastError = 10 constexpr