WaveCatcher64Ch

Control and Readout Library

User's Manual

Date: 1/12/2016	WaveCatcher64Ch Library User's Manual
By D.Breton & J.Maalmi, LAL Orsay	V/Ref. : 1.1.9

PURPOSE OF THIS MANUAL

This User's Manual contains the full description of the WaveCatcher64Ch Control and Readout Library. It is compatible with the 8-Channel WaveCatcher module, the 16-Channel WaveCatcher boards, and the 64-Channel WaveCatcher Crate.

EVOLUTIONS OF THE DOCUMENT

DATE	REVISION	SOFTWARE VERSION	CHANGES
07/30/2014	V1.1.1	UP TO 2.4.5	FIRST VERSION OF THE DOCUMENT
09/02/2014	V1.1.2	2.4.6	Introduction of the coincidence masks for rate counters
10/21/2014	V1.1.3	2.4.12	Addition of the functions: WAVECAT64CH_StartRatecounters and WAVECAT64CH_ReadRatecounters
12/02/2014	V1.1.4	2.4.14	Addition of the functions: WAVECAT64CH_EnableRatecounters and WAVECAT64CH_DisableRatecounters
01/09/2015	V1.1.5	2.4.18	Addition of the functions: WAVECAT64CH_EnableRateCounterVeto, WAVECAT64CH_DisableRateCounterVeto, WAVECAT64CH_SetRateCounterVetoLength and WAVECAT64CH_GetRateCounterVetoLength
03/23/2015	V1.1.6	2.5.1	Addition of the functions: WAVECAT64CH_GetWaveCatDevices, WAVECAT64CH_OpenSelectedDevice, WAVECAT64CH_SetChannelState and WAVECAT64CH_GetChannelState
06/05/2015	V1.1.7	2.5.3	Addition of the field LibraryVersion in the WAVECAT64CH_DeviceInfoStruct Addition of the functions: WAVECAT64CH_ SetTriggerPrimitivesPairedOption, WAVECAT64CH_ GetTriggerPrimitivesPairedOption, WAVECAT64CH_SetExternalClockOption, WAVECAT64CH_GetExternalClockOption, WAVECAT64CH_ SetTriggerPrimitivesOnExtTrigOutOption, WAVECAT64CH_ GetTriggerPrimitivesOnExtTrigOutOption
06/17/2015	V1.1.8	2.5.3	Few corrections in the description of various items
09/21/2015	V1.1.9	2.6.1	- Correction of the wrong polarity of the FixedThresholdOrCFD bit in the WAVECAT64CH_FirmwareMeasurementParameters structure - Addition of all the error return codes in all the functions accessing to hardware - New version of the

			WAVECAT64CH_FirmwareMeasurementParameters structure wrt the new firmware for the measurement block (concerns a better calculation of the edges' time and the introduction of the dynamic charge) - Correlated introduction of the charge overflow bit in WAVECAT64CH_ChannelDataStruct
01/11/2016	V1.1.10	2.6.6	Adding of a scheme describing the measurement parameters in 4.7.

REFERENCE DOCUMENTS

[RD1] USB_WaveCatcher V6 Documentation V/Ref 1
[RD2] LALUsb 2.0 library - Software version 1.0.0.1

INDEX

1 Introduction

1.1 Short description of the WaveCatcher systems

The WaveCatcher64Ch acquisition library is intended to permit an easy development of acquisition software for control and readout of the various WaveCatcher systems. The latter currently range between 8 and 64 (+8) channels.
They all make use of the **SAMLONG** analog memory chips which permit sampling the input signal between 400 MS/s and 3.2 GS/s over 12 bits.
There are 3 different types of systems:

- 8-channel (autonomous desktop), composed of a motherboard equipped with two 4-channel mezzanines
- 16-channel (6U board)
- 64-channel (mini crate). This crate can actually house between 1 and 4 16-channel boards, thus providing 16, 32, 48 or 64 channels.

From the second version of the 16-channel boards on, 2 extra channels have been added in the back of the board. They can be digitized together with the other channels. When the board is used in standalone mode, these channels correspond to the external trigger and the external sync. Otherwise, they are equivalent to other channels.

The systems are currently interfaced with a 480 MBits/s USB link. A UDP interface (8 and 64-channel systems) and an optical link (all systems) will soon be put into function.

1.2 Computer system requirements

Host PC requirements:
Linux, Windows XP and above.

The WaveCatcher64Ch Acquisition Library:
the WaveCatcher64Ch acquisition library is a package of three files: **WaveCat64Ch_Dll.lib**, **WaveCat64Ch_Dll.Dll** and **WaveCat64Ch_Dll.h**

Third party required library:
LALUsbML Library: download and install the **LAL UsbML library** package:

http://electronique.lal.in2p3.fr/echanges/LALUsb/software.html

1.3 Return Codes

Error Code	Value	Meaning
WAVECAT64CH_Success	0	Operation completed successfully
WAVECAT64CH_CommError	-1	Communication error
WAVECAT64CH_GenericError	-2	Unspecified error
WAVECAT64CH_DeviceNotFound	-3	Unable to open device
WAVECAT64CH_DeviceAlreadyOpen	-4	Device already open
WAVECAT64CH_NoDeviceConnected	-5	No device is connected
WAVECAT64CH_InvalidParam	-6	One or more parameters are

		invalid, mostly meaning absent of a list
WAVECAT64CH_No_Event	-7	No event is ready to be read
WAVECAT64CH_IncompleteEvent	-8	Event is incomplete
WAVECAT64CH_ReadoutError	-9	Readout error
WAVECAT64CH_ErrorAcquisitionRunning	-10	Accessing the device while acquisition is running is not allowed
WAVECAT64CH_EventHeaderError	-11	Wrong event header
WAVECAT64CH_EventIdInconsistenceError	-12	Inconsistent event ID between blocks
WAVECAT64CH_SourceIdInconsistenceError	-13	Inconsistent source ID between blocks
WAVECAT64CH_EventTrailerError	-14	Wrong event trailer
WAVECAT64CH_ExtendedReadoutError	-15	Error during extended event readout
WAVECAT64CH_OutOfRangeParam	-16	One of the parameter values is out of range
WAVECAT64CH_NotYetImplemented	-99	The function is not yet implemented

Tab. 1 Return Code Table

2 Definition of constants and enums

```
#define  WAVECAT64CH_MAX_DATA_SIZE 1024
```

Info

This corresponds to the maximum number of samples which can be stored inside the SAMLONG chips.

```
#define  WAVECAT64CH_INFO_NOT_AVAILABLE -1
```

```
#define  WAVECAT64CH_MAX_NB_OF_FE_BOARDS 4
```

Info

This corresponds to the maximum number of front-end boards in a system.

```
#define  WAVECAT64CH_NB_OF_CHANNELS_IN_SAMBLOCK 2
```

```
#define  WAVECAT64CH_MAX_TOTAL_NB_OF_SAMBLOCKS 36
```

Info

This corresponds to the maximum possible number of SAMLONG chips in a system (each chip houses 2 channels).

```
#define  WAVECAT64CH_MAX_TOTAL_NB_OF_CHANNELS  72
```

Info

This corresponds to the maximum possible number of channels in a system (of any type => see below for channel type).

```
#define  WAVECAT64CH_ADCTOVOLTS 0.00061
```

Info

This defines the conversion of the ADC count into Volts.

```
typedef enum
{
        WAVECAT64CH_SYSTEM_IS_WAVECAT64CH_8CH,
        WAVECAT64CH_SYSTEM_IS_WAVECAT64CH_16CH,
        WAVECAT64CH_SYSTEM_IS_WAVECAT64CH_64CH

} WAVECAT64CH_TypeOfSystem;
```

```
typedef enum
{
        WAVECAT64CH_EVENT_WITH_WAVEFORM =0,
        WAVECAT64CH_EVENT_WITHOUT_WAVEFORM =1

} WAVECAT64CH_DataType;
```

Info

This type is not yet in use.

```
typedef enum
{
	WAVECAT64CH_SYSTEM_FPGA,
	WAVECAT64CH_CTRL_FPGA,
	WAVECAT64CH_FE_FPGA

} WAVECAT64CH_FpgaType;
```

Info

This enumeration is used only for the low level access to the board.

```
typedef enum
{
	WAVECAT64CH_THRESHOLDDACS_NOT_AVAILABLE = -1,
	WAVECAT64CH_THRESHOLDDACS_LOADED_FROM_EEPROM = 0,
	WAVECAT64CH_THRESHOLDDACS_LOADED_FROM_FILES = 1,

} WAVECAT64CH_ThresholdDacs_INFO;
```

Info

This enumeration is about the correction of the Trigger Thresholds from their calibration data.

```
typedef enum
{
	WAVECAT64CH_SAMDACOFFSETS_NOT_AVAILABLE = -1,
	WAVECAT64CH_SAMDACOFFSETS_LOADED_FROM_EEPROM = 0,
	WAVECAT64CH_SAMDACOFFSETS_LOADED_FROM_FILES = 1,

} WAVECAT64CH_SamDacOffsets_INFO;
```

Info

This enumeration is about the correction of the line offsets inside the SAMLONG chips from their calibration data.

```
typedef enum
{
	WAVECAT64CH_PEDESTALS_NOT_AVAILABLE = -1,
	WAVECAT64CH_PEDESTALS_LOADED_FROM_EEPROM = 0,
	WAVECAT64CH_PEDESTALS_LOADED_FROM_FILES = 1,

} WAVECAT64CH_Pedestals_INFO;
```

Info

This enumeration is about the correction of remaining individual pedestals from their calibration data.

```
typedef enum
{
	WAVECAT64CH_INL_NOT_AVAILABLE = -1,
	WAVECAT64CH_INL_LOADED_FROM_EEPROM = 0,
	WAVECAT64CH_INL_LOADED_FROM_FILES = 1

} WAVECAT64CH_INL_INFO;
```

Info

This enumeration is about the correction of the time INL from their calibration data.

```
typedef enum
{
	WAVECAT64CH_TRIGGER_SOFT = 0,
	WAVECAT64CH_TRIGGER_NORMAL = 1,
	WAVECAT64CH_TRIGGER_INTERNAL = 2,
	WAVECAT64CH_TRIGGER_EXTERNAL = 3,
	WAVECAT64CH_TRIGGER_COINCIDENCE = 4,
	WAVECAT64CH_TRIGGER_NORMAL_PAIRED = 5,
	WAVECAT64CH_TRIGGER_COINCIDENCE_WITH_EXT_SIG = 6

} WAVECAT64CH_TriggerType;
```

Info

WAVECAT64CH_TRIGGER_COINCIDENCE means coincidence between input channels.
WAVECAT64CH_TRIGGER_NORMAL_PAIRED means coincidence between pairs of consecutive channels (0&1, 2&3, …)
WAVECAT64CH_TRIGGER_COINCIDENCE_WITH_EXT_SIG means that the external trigger is added into the coincidence of the other channels.

```
typedef enum
{
	WAVECAT64CH_FRONT_CHANNEL = 0,
	WAVECAT64CH_BACK_EXTRA_CHANNEL = 1

} WAVECAT64CH_ChannelType;
```

Info

On the 8-channel module, there is no extra channel.
From the second version of the 16-channel front-end boards on, 2 extra channels have been added in the back of the board. They can be digitized together with the other channels. When the board is used in standalone mode, these channels correspond to the external trigger and the external sync. Otherwise, they are equivalent to other channels. Their number (2 to 8) depends on the number of 16-channel boards in the system.

```
typedef enum
{
	WAVECAT64CH_EXT_TRIG = 0,
	WAVECAT64CH_EXT_SYNC = 1

} WAVECAT64CH_ExternalSignalType;
```

```
typedef enum
{
	WAVECAT64CH_LVTTL = 0,
	WAVECAT64CH_NIM = 1

} WAVECAT64CH_ExternalSignalNormType;
```

```
typedef enum
{
```

```
        WAVECAT64CH_3_2GHZ = 3200,
        WAVECAT64CH_2_13GHZ = 2133,
        WAVECAT64CH_1_6GHZ = 1600,
        WAVECAT64CH_1_28GHZ = 1280,
        WAVECAT64CH_1_07GHZ = 1067,
        WAVECAT64CH_800MHZ = 800,
        WAVECAT64CH_533MHZ = 533,
        WAVECAT64CH_400MHZ = 400

} WAVECAT64CH_SamplingFrequencyType;
```

```
typedef enum
{
        WAVECAT64CH_STATE_ON = 1,
        WAVECAT64CH_STATE_OFF = 0

} WAVECAT64CH_StateType;
```

```
typedef enum
{
 WAVECAT64CH_POS_EDGE = 0,
 WAVECAT64CH_NEG_EDGE = 1

} WAVECAT64CH_TriggerEdgeType;
```

```
typedef enum
{
 WAVECAT64CH_STANDARD_READOUT = 0,
 WAVECAT64CH_SHORT_READOUT = 1

} WAVECAT64CH_ReadoutLatencyType;
```

Info

Readout and ADC conversion can be performed either at 10 MHz (standard mode) or at 20 MHz (short mode). The short mode reduces the latency by a factor close to 2 but the noise then increases by a few %.

```
typedef enum
{
 WAVECAT64CH_PULSER_SOURCE_IS_USB =0 ,
 WAVECAT64CH_PULSER_SOURCE_IS_EXT_SIG =1

} WAVECAT64CH_PulserSourceType;
```

Info

This describes the source of the individual pulsers located on each channel. By default, it is a 375 kHz random clock.

```
typedef enum
{
 WAVECAT64CH_NO_CORRECTION,
 WAVECAT64CH_PEDESTAL_CORRECTION_ONLY,
 WAVECAT64CH_ALL_CORRECTIONS
```

} **WAVECAT64CH_CorrectionModeType**;

Info

By default, all corrections are active.

3 Definition of structures

```
typedef struct{

char    LibraryVersion[8]; // Version of the library used by the software

int     BoardConnected; //Board connection status 0 = Not connected, 1 = Connected

int     ControllerBoardVersion;     // ControllerBoard: for 64-channel systems
int     ControllerBoardSerialNumber;
int     ControllerBoardFPGAVersion;
int     ControllerBoardFPGAEvolution;

int     MotherBoardVersion;         // MotherBoard: for 8-channel systems
int     MotherBoardSerialNumber;
int     MotherBoardFPGAVersion;
int     MotherBoardFPGAEvolution;

int     BoardVersionFromFPGA;   // Front-end board (16-channel or more) or mezzanine (8-channel systems)
int     FPGAVersion;
int     FPGAEvolution;

WAVECAT64CH_ThresholdDacs_INFO   ThresholdDacs_Info; // Trigger threshold DACs calibration status (cf.enum type description)
WAVECAT64CH_SamDacOffsets_INFO   SamDacOffsets_Info; // SAM DACs calibration status (cf.enum type description)
WAVECAT64CH_Pedestals_INFO       Pedestals_Info; //Pedestal calibration status (cf.enum type description)
WAVECAT64CH_INL_INFO             INL_Info; // INL calibration status (cf.enum type description)

WAVECAT64CH_TypeOfSystem         SystemType; // See different system types in enum

int     NbOfLayers; // Number of layers in the USB_ML architecture (2 for 8-channel and 16-channel, 3 for 64-channel systems)
int     NbOfFeBoards; // For 64-channel systems: between 1 and 4
int     FE_Paths[MAX_NB_OF_FE_BOARDS]; // Paths in the USB_ML architecture towards the different front-end boards
int     NbOfFeFPGAs; // Total number of Front-end FPGAs in the system.
int     NbOfFeSamBlocks; // Total number of Front-end SAMLONG chips in the system.
int     NbOfBackExtraSamBlocks; // Total number of SAMLONG chips for extra channels in the system.
int     NbOfFeChannels; // Total number of Front-end channels in the system.
int     NbOfBackExtraChannels; // Total number of extra channels in the system.
char    BoardFullSerialNumber[MAX_NB_OF_FE_BOARDS][8]; // Can be used for 8-channel and 64-channel systems

} WAVECAT64CH_DeviceInfoStruct;
```

```
typedef struct{

        WAVECAT64CH_ChannelType  ChannelType; // Type of channel (front/back)
        int     Channel; // Channel number for the specified type
        int     TrigCount; // For channel rate counter (* see info below)
        int     TimeCount; // For channel rate counter (* see info below)
```

```
        int     WaveformDataSize; // In samples
        float   *WaveformData; // Event data for specified channel
// All measurements below performed by firmware, as defined by the
// WAVECAT64CH_FirmwareMeasurementParameters structure
        float   Baseline; // Signal baseline in ADC counts
        float   Peak; // Peak of signal in ADC counts
        int     PeakCell; // Peak location. Units are in samples from the beginning of the waveform
        float   Charge; // Bits 0 to 22 : charge extracted from signal (sum of sample contributions in
        ADC counts). Bit 23 at 1 means charge overflow and charge value is then forced to 0x3FFFFF
        float   CFDRisingEdgeTime; // Rising edge threshold crossing location. Units are in
samples from the beginning of the waveform => values between 0.0 and 1024.0
        float   CFDFallingEdgeTime; // Falling edge threshold crossing location. Units are in
samples from the beginning of the waveform => values between 0.0 and 1024.0
        int     FCR; // First cell read in the SAMLONG chip

} WAVECAT64CH_ChannelDataStruct;
```

** In order to calculate a rate, two counters are necessary: one is counting the hits (HitCounter over 16 bits) since the last event was read, while the other (TimeCounter over 24 bits) is counting the time in units of 32000 sampling intervals or 200 clock periods (this corresponds for instance to 1 µs at 3.2 GS/s). Reading their values stops both of them, as well as the possible saturation of any of them.* ***The rate is independent from the number of events read per second.***

```
typedef struct{

        int                     EventID;
        unsigned long long      TDC;
        WAVECAT64CH_ChannelDataStruct   *ChannelData;
        int                     NbOfSAMBlocksInEvent; // 2 channels per SAMBlock

} WAVECAT64CH_EventStruct;
```

```
typedef struct{

        Boolean         ForceBaseline;
        float           EnforcedBaselineValue;      // in Volts
        Boolean         PeakPolarity;   // 0 => positive, 1 => negative
        Boolean         FixedThresholdOrCFD;        // 0 => Fixed Threshold, 1 => CFD
        float           FixedThresholdValue;        // in Volts
        int             PeakRatio;      // Between 1 and 16 (units of 1/16)
        int             RefCellForCharge;  // In samples from the beginning of the waveform.
        Boolean         EnableDynamicCharge // 1 => enables the dynamic charge mode.
        int             PreCharge;      // Between 1 and 99 (in samples)
        int             ChargeLength; // Between 4 and 63 (in columns => multiply by 16 for the
        equivalent number of samples)

} WAVECAT64CH_FirmwareMeasurementParameters*;
```

** These parameters are used* ***only for measurements performed inside the firmware****. The baseline is calculated over the 16 first samples of the waveform. One can chose using this result or a value fixed by user for baseline subtraction. The peak is relative to the baseline. Its polarity has to be chosen by user. Time measurement of the rising and falling edges relative to the peak can be performed either with a fixed threshold set by user or via a*

constant fraction discriminator (CFD). In the latter case, the ratio to the peak has to be defined. Charge measurement can be performed starting either from a cell defined by user, or automatically (dynamic charge mode). In the latter case, one has to define the distance before the peak at which calculation will start (PreCharge). In both cases, the duration of the charge calculation has to be defined (ChargeLength). Be aware that the calculation may not be over in dynamic mode when one reaches the end of the waveform. In this case, the bit 23 of the charge sent to WAVECAT64CH_ChannelDataStruct will be set to 1 ("charge overflow") and the value of the charge will be forced to 0x3FFFFF.

4 Definition of functions

4.1 Communication

WAVECAT64CH_GetWaveCatDevices

Description
This function looks for all the WaveCatcher devices connected and returns their number and the corresponding arrays of device descriptors and serial numbers.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ GetWaveCatDevices (
char **deviceDescriptor[MAX_DESCRIPTOR_SIZE]**,
char **usbSerNumchar[SERNUM_SIZE]**,
int *** nbOfWaveCatDevices**);

Arguments

Name	Description
deviceDescriptor[MAX_DESCRIPTOR_SIZE]	List of the device descriptors of the WaveCatcher devices connected
usbSerNumchar[SERNUM_SIZE]	List of the serial numbers of the WaveCatcher devices connected
*** nbOfWaveCatDevices**	Total number of WaveCatcher devices connected

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_OpenDevice

Description
This function opens the single WaveCatcher device connected and gets its USB/UDP handle. It sets all parameters (including correction data) to their default value and loads all systems registers. It also resets the board FPGAs but does not reset the registers.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_OpenDevice(
int ***handle**);

Arguments

Name	Description
***handle**	Pointer to the USB device handle returned by the function

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_OpenSelectedDevice

Description
This function opens the selected WaveCatcher device connected (described by its device descriptor and serial number) and gets its USB/UDP handle. It sets all parameters (including correction data) to their default value and loads all systems registers. It also resets the board FPGAs but does not reset the registers.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_OpenDevice(
char deviceDescriptor[MAX_DESCRIPTOR_SIZE],
char usbSerNumchar[SERNUM_SIZE],
int *handle);
```

Arguments

Name	Description
deviceDescriptor[MAX_DESCRIPTOR_SIZE]	List of the device descriptors of the WaveCatcher devices connected
usbSerNumchar[SERNUM_SIZE]	List of the serial numbers of the WaveCatcher devices connected
***handle**	Pointer to the USB device handle returned by the function

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_CloseDevice

Description
This function closes an opened WaveCatcher device.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_CloseDevice(void);
```

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetDeviceInfo

Description
This function returns a structure with all necessary information about the device: connection status, Board Version, Serial Number, Calibration status, etc…

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_GetDeviceInfo(
WAVECAT64CH_DeviceInfoStruct *DeviceInfoStruct);
```

Arguments

Name	Description
***DeviceInfoStruct**	Pointer to a structure containing information about the board as described above.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_WriteAccess

Description
This function permits a generic write access to the device registers.
Notice that the WaveCatcher Control and Readout Library provides high-level functions for most of the device parameter settings. If the user makes use of this function to overwrite some settings, it may cause an inconsistency of the operations.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_WriteAccess(
int handle,
WAVECAT64CH_FpgaType fpga_type,
int board_target,
int front_end_target,
char sub_address,
void* buffer,
int word_count);
```

Arguments

Name	Description
handle	The device handle
fpga_type	The type of FPGA targeted (see enum)
board_target	The number of the board targeted
front_end_target	The number of the front-end FPGA targeted
sub_address	Register sub-address to access
buffer	Data buffer to write
word_count	Number of bytes to write

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_ReadAccess

Description
This function permits a generic read access to any of the device registers.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_ReadAccess(
int handle,
WAVECAT64CH_FpgaType fpga_type,
int board_target,
int front_end_target,
char sub_address,
uchar* buffer,
int word_count);
```

Arguments

Name	Description
handle	The device handle
fpga_type	The type of FPGA targeted (see enum)
board_target	The number of the board targeted
front_end_target	The number of the front-end FPGA targeted
sub_address	Register sub-address to access
buffer	Data buffer to put the read data
word_count	Number of bytes to write

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

4.2 Initialization

WAVECAT64CH_ResetDevice

Description
This function resets the state machines and FIFOs inside the FPGA **but not** the registers of the board (only a “Power On Reset” consecutive to an off/on switching of the device resets the registers of the device to their hardware default values). It also purges the USB buffers of all remaining data.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_ResetDevice(void);
```

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetDefaultParameters

Description
This function initializes all the board parameters in order to run properly with the library. This function is called after opening of the device and should be called each time one wants to reset the system to its default parameters.
To know the default value set by this function for an available parameter (for example Trigger Mode, Trigger Thresholds, etc…), use the corresponding WAVECAT64CH_Get** function.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_SetDefaultParameters(void);
```

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.3 Calibration and correction

WAVECAT64CH_LoadCalibrationParametersFromEEPROM

Description
This function reads all the calibration parameters from the EEPROM and reloads all the structures and boards.

Synopsis
```
WAVECAT64CH_ErrCode
WAVECAT64CH_LoadCalibrationParametersFromEEPROM(void);
```

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_CalibratePedestals

Description
This function calibrates the board pedestals. Pedestal calibration parameters are already stored in the board's EEPROM, but they should be computed again after a firmware update then stored in the EEPROM using *WAVECAT64CH_SavePedestalCalibrationToEEPROM* function.
No signal should be put on the inputs during this calibration.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_CalibratePedestals(
int nbOfRunsForCalibration);
```

Arguments

Name	Description
nbOfRunsForCalibration	Number of runs for the pedestals measurement, which should be ~300

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

Remark
Not yet implemented.

WAVECAT64CH_SavePedestalCalibrationToEEPROM

Description
This function stores the pedestals calibration values into the board's EEPROM.

Synopsis
```
WAVECAT64CH_ErrCode  WAVECAT64CH_SavePedestalCalibrationToEEPROM(void);
```

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Remark
Not yet implemented.

WAVECAT64CH_SetTimeCorrectionMode

Description
This function permits to fix the correction mode. Two types of correction are available, based on the calibration parameters already stored in the boards' EEPROMs:

- pedestal correction
- time INL correction

Based on these two types of correction, 3 modes are defined:

- WAVECAT64CH_NO_CORRECTION
- WAVECAT64CH_PEDESTAL_CORRECTION_ONLY
- WAVECAT64CH_ALL_CORRECTIONS

When time INL correction mode is enabled, the time precision of the device is **< 5ps**. If It is disabled the time precision is <**20 ps**. Due to the online calculation requested, time INL correction may slow down the acquisition.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetCorrectionMode(
WAVECAT64CH_CorrectionModeType **correctionMode**);

Arguments

Name	Description
correctionMode	Correction mode (cf. enum above)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTimeCorrectionMode

Description
This function returns the current time correction mode.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetCorrectionMode(
WAVECAT64CH_CorrectionModeType ***correctionMode**);

Arguments

Name	Description
***correctionMode**	Correction mode returned by the function

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.4 Trigger Configuration

WAVECAT64CH_SetTriggerMode

Description
This function sets the system's trigger mode:
In *Software Triggering Mode*, software sends the trigger to the system.
In *Normal Triggering Mode*, the system makes use of the input channel discriminators and triggers only if the signal on one of the enabled channels crosses the defined trigger threshold with the defined edge.
In *Internal Triggering Mode*, the system triggers by itself using the internal 12MHz random clock.
In *External Triggering Mode*, the system makes use of the External Trigger input with the defined polarity and edge.
In *Coincidence Triggering Mode*, the system performs the coincidence between the defined gated output of the channel discriminators.
In *Normal Paired Triggering Mode*, the system performs the OR of the coincidences between pairs of consecutive gated channels (0&1, 2&3, …).
In *Coincidence With External Signal Triggering Mode*, the system adds the external trigger to the coincidence with the other channels.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerMode(
WAVECAT64CH_TriggerType **triggerType**);

Arguments

Name	Description
triggerType	Trigger type to be chosen between: WAVECAT64CH_TRIGGER_SOFT WAVECAT64CH_TRIGGER_NORMAL WAVECAT64CH_TRIGGER_INTERNAL WAVECAT64CH_TRIGGER_EXTERNAL WAVECAT64CH_TRIGGER_COINCIDENCE WAVECAT64CH_TRIGGER_NORMAL_PAIRED WAVECAT64CH_TRIGGER_COINCIDENCE_WITH_EXT_SIG

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerMode

Description
This function returns the current board's trigger mode.

Synopsis
WAVECAT64CH *_ErrCode* WAVECAT64CH _GetTriggerMode(
WAVECAT64CH _TriggerType ***triggerType**);

Arguments

Name	Description

***triggerType**	Trigger type returned by the function (see description right above)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerSourceState

Description
This function enables or disables a specified channel as a source of trigger. It has no relationship with the waveform sampling activity of the given channel.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerSourceState(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
WAVECAT64CH_StateType **triggerState**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
triggerState	Source for trigger is ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerSourceState

Description
This function returns the current state (ON/OFF) of a specified channel as a source of trigger.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetTriggerSourceState(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
WAVECAT64CH_StateType ***triggerState**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
***triggerState**	State of the source for trigger:ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerEdge

Description
This function sets the polarity of the trigger edge (negative or positive) for the specified channel as a source of trigger.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerEdge(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
WAVECAT64CH_TriggerEdgeType **triggerEdge**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
triggerEdge	Trigger polarity: positive edge (WAVECAT64CH_POS_EDGE) or negative edge (WAVECAT64CH_NEG_EDGE)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerEdge

Description
This function returns the polarity of the trigger edge (negative or positive) for the specified channel as a source of trigger.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetTriggerEdge(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
WAVECAT64CH_TriggerEdgeType ***triggerEdge**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
***** **triggerEdge**	Trigger polarity: positive edge (WAVECAT64CH_POS_EDGE) or negative edge (WAVECAT64CH_NEG_EDGE)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetExternalSignalNorm

Description
This function sets the signal norm used (NIM or LVTTL or positive) for the specified external signal.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetExternalSignalNorm(
WAVECAT64CH_ExternalSignalType **extSigType**,
WAVECAT64CH_ExternalSignalNormType **extSigNormType**);

Arguments

Name	Description
extSigType	Type of the channel to enable/disable: WAVECAT64CH_EXT_TRIG or WAVECAT64CH_EXT_SYNC
extSigNormType	Norm used: NIM (WAVECAT64CH_NIM) or LVTTL (WAVECAT64CH_LVTTL)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetExternalSignalEdge

Description
This function sets the polarity of the trigger edge (negative or positive) for the specified external signal.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetExternalSignalEdge(
WAVECAT64CH_ExternalSignalType **extSigType**,
WAVECAT64CH_TriggerEdgeType **triggerEdge**);

Arguments

Name	Description
extSigType	Type of the channel to enable/disable: WAVECAT64CH_EXT_TRIG or WAVECAT64CH_EXT_SYNC
triggerEdge	Trigger polarity: positive edge (WAVECAT64CH_POS_EDGE) or negative edge (WAVECAT64CH_NEG_EDGE)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetExternalSignalEdge

Description
This function returns the polarity of the trigger edge (negative or positive) for the specified external signal.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_SetExternalSignalEdge(
WAVECAT64CH_ExternalSignalType **extSigType**,
WAVECAT64CH_TriggerEdgeType ***triggerEdge**);

Arguments

Name	Description
extSigType	Type of the channel to enable/disable: WAVECAT64CH_EXT_TRIG or WAVECAT64CH_EXT_SYNC
***triggerEdge**	Trigger polarity: positive edge (WAVECAT64CH_POS_EDGE) or negative edge (WAVECAT64CH_NEG_EDGE)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerThreshold

Description

This function sets the trigger threshold for the specified channel.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerThreshold(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
float **thresholdValue**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
thresholdValue	Trigger threshold value in Volts for the specified channel. This value is dependent of the DC offset of the specified channel. Their difference must always range between -1.25 and 1.25 V. Otherwise, the function returns the error code: WAVECAT64CH_OutOfRangeParam.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerThreshold

Description

This function returns the current trigger threshold value of the specified channel.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_GetTriggerThreshold(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
float ***thresholdValue**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
***thresholdValue**	Trigger threshold value in Volts for the specified channel. This value is dependent of the DC offset of the specified channel.

Return Values
0: Success; Negative numbers are error codes (see Return Codes)

WAVECAT64CH_SetTriggerDelay

Description
This function sets the delay between the trigger and the stopping of the analog memory, called post-trigger delay (PostTrig).

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerDelay(
unsigned char **triggerDelay**);

Arguments

Name	Description
triggerDelay	Post-trigger delay: 8-bit integer (ranging between 1 and 255) in units of sampling clock period of the analog memory. If this range is not respected, the function returns the error code WAVECAT64CH_OutOfRangeParam. Multiplying this number by **16 times the sampling interval (or by one clock period)** of the analog memory gives the trigger delay in **ns.**

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerDelay

Description
This function returns the current post-trigger delay (see definition right above).

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerDelay(
unsigned char ***triggerDelay**);

Arguments

Name	Description
***triggerDelay**	Trigger delay returned by the function

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerPrimitivesGateLength

Description
This function sets the value of the board's internal gate length for coincidence modes. In coincidence mode, the board triggers if the trigger sources toggle together during the duration of this gate. The gate is opened by the first trigger source that toggles, and ends after the defined gate length.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetTriggerPrimitivesGateLength(
unsigned int **gateLength**);

Arguments

Name	Description
gateLength	The gate length is an unsigned integer **defined in ns**, with a minimum value of 15 ns (hardware offset). It will be converted into clock periods of SAMLONG, with a possible range between 0 and 255 (corresponding to a respective maximum of 1.275 µs @ 3.2 GS/s and up to 10.2 µs @ 400 MS/s). If these conditions are not respected, the function returns the error code: WAVECAT64CH_OutOfRangeParam.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerPrimitivesGateLength

Description
This function returns the current value of the internal gate length (see definition right above).

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetTriggerPrimitivesGateLength(
unsigned int ***gateLength**);

Arguments

Name	Description
***gateLength**	Gate length returned by the function.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerPrimitivesPairedOption

Description
This function enables or disables the "paired" option, which is the possibility to replace the OR by the AND of the two channels of all SAMBLOCKs in the building of the trigger

primitives. It is intended for instance to be used for triggering on coincidences of both extremities of strip lines, if the latter have systematically been connected to the same SAMBLOCKs.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_ SetTriggerPrimitivesPairedOption (WAVECAT64CH_StateType **pairedOptionState**);

Arguments

Name	Description
pairedOptionState	State of the “paired” option: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerPrimitivesPairedOption

Description

This function returns the current state (ON/OFF) of the “paired” option (see just above).

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_ GetTriggerPrimitivesPairedOption (WAVECAT64CH_StateType ***pairedOptionState**);

Arguments

Name	Description
***pairedOptionState**	State of the “paired” option: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetTriggerPrimitivesOnExtTrigOutOption

Description

This function enables or disables the possibility to replace on the Trig_Out front-panel plug the board trigger by the primitives used to build it. This permits synchronizing the trigger of multiple boards or crates. In order to take the local enable_trigger signal into account, the latter is ANDED with the primitives before the signal is sent to the plug.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_ SetTriggerPrimitivesOnExtTrigOutOption (WAVECAT64CH_StateType **primitiveOptionState**);

Arguments

Name	Description
primitiveOptionState	State of the option:

	ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetTriggerPrimitivesOnExtTrigOutOption

Description
This function returns the current state (ON/OFF) of the option (see just above).

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ GetTriggerPrimitivesOnExtTrigOutOption (WAVECAT64CH_StateType ***primitiveOptionState**);

Arguments

Name	Description
*** primitiveOptionState**	State of the option: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SendSoftwareTrigger

Description
This function sends a software trigger to the system. Before using it, make sure it is necessary, because it can get in conflict with data readout. By default, prefer using the **WAVECAT64CH_PrepareEvent** function to send the software trigger.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SendSoftwareTrigger(void);

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.5 Channel Setup

WAVECAT64CH_SetChannelState

Description
This function enables or disables a specified channel as a source of data. It has no relationship with the trigger activity of the given channel.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_SetChannelState(
WAVECAT64CH_ChannelType channelType,
int channel,
WAVECAT64CH_StateType channelState);
```

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
channelState	Source for data is ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetChannelState

Description
This function returns the current state (ON/OFF) of a specified channel as a source of data.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_GetChannelState(
WAVECAT64CH_ChannelType channelType,
int channel,
WAVECAT64CH_StateType * channelState);
```

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
***triggerState**	State of the source for data: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetChannelDCOffset

Description
This function sets the DC Offset (ranging between -1.25 and 1.25 V) which will be added to the specified channel.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_SetChannelDCOffset(
WAVECAT64CH_ChannelType channelType,
int channel,
float channelDCOffset);
```

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
channelDCOffset	DC Offset value in Volts for the specified channel (float number between -1.25 and 1.25) If this range is not respected, the function returns the error code: WAVECAT64CH_OutOfRangeParam.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetChannelDCOffset

Description
This function returns the current DC Offset value for the specified channel.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_GetChannelDCOffset(
WAVECAT64CH_ChannelType channelType,
int channel,
float *channelDCOffset);
```

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
***channelDCOffset**	DC Offset value in Volts for the specified channel (float number between -1.25 and 1.25).

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.6 Operating channel pulsers

WAVECAT64CH_EnablePulsers

Description
This function enables all the pulsers in the system. In order to mask specific channels, use the WAVECAT64CH_SetPulsePattern function with the **pulsePattern** parameter set to 0**.**

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_EnablePulsers (void);
```

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_DisablePulsers

Description
This function disables all the pulsers in the system. The pulsePattern registers are not modified.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_DisablePulsers (void);
```

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetPulsePattern

Description
This function sets the value of the pulse pattern for the specified channel.

Synopsis
```
WAVECAT64CH_ErrCode WAVECAT64CH_SetPulsePattern(
WAVECAT64CH_ChannelType channelType,
int channel,
unsigned short pulsePattern);
```

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
pulsePattern	Pulse pattern is a 16-bit unsigned short (thus ranging between 0x0000 and 0xFFFF). The value 0x0000 means that the corresponding pulser is OFF. The pulses sent follow the binary format of the pulse pattern, LSB being sent first.

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetPulsePattern

Description
This function returns the current value of the pulse pattern for the specified channel.

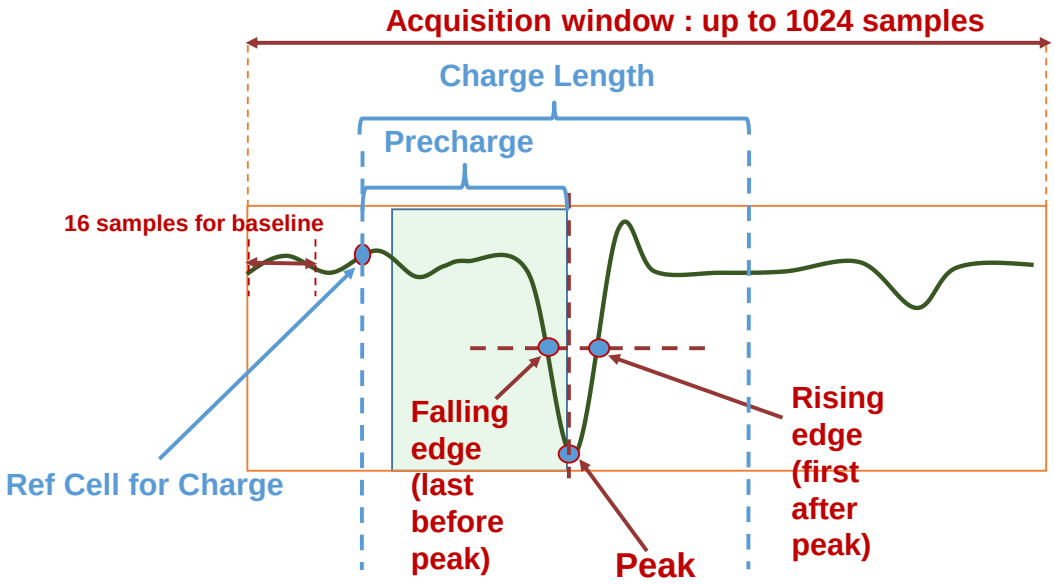
Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetPulsePattern(
WAVECAT64CH_ChannelType **channelType**,
int **channel**,
unsigned short ***pulsePattern**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
channel	Channel number for the given type
pulsePattern	Pulse pattern is a 16-bit unsigned short (thus ranging between 0x0000 and 0xFFFF). The value 0x0000 means that the corresponding pulser is OFF. The pulses sent follow the binary format of the pulse pattern, LSB being sent first.

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

4.7 Configurating the firmware measurements



Programmable options :

- Peak polarity: Pos/Neg
- Forced/Extracted baseline
- CFD/Fixed Threshold
- CFD Ratio (N/16)
- Ref cell for Charge / Start from peak (=> precharge)
- Charge Length

WAVECAT64CH_SetFirmwareMeasurementParameters

Description
This function sets the parameters used by the firmware to perform the real-time measurements on the digitized signal.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetFirmwareMeasurementParameters(WAVECAT64CH_FirmwareMeasurementParameters **measParams**);

Arguments

Name	Description
measParams	See description of the structure in chapter 3

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetFirmwareMeasurementParameters

Description
This function returns the parameters currently used by the firmware to perform the real-time measurements on the digitized signal.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_GetFirmwareMeasurementParameters(
WAVECAT64CH_FirmwareMeasurementParameters ***measParams**);

Arguments

Name	Description
***measParams**	See description of the structure in chapter 3

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

4.8 Rate counters parameters

WAVECAT64CH_EnableRateCounters

Description
This function enables all the front-end rate counters in the system.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_EnableRateCounters (void);

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_DisableRateCounters

Description
This function disables all the front-end rate counters in the system.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_DisableRateCounters (void);

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetRateCounterCoincidenceMask

Description
The channels are naturally grouped by four inside the so-called front-end FPGAs on the boards. This permits defining a coincidence pattern between those 4 channels for feeding the 4 individual rate counters also located inside each front-end FPGA. The gate length defined by the function WAVECAT64CH_SetTriggerPrimitivesGateLength is used for the coincidence window.
This function sets the coincidence mask for the given rate counter inside the given group of 4 channels.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetRateCounterCoincidenceMask (
int **fourChannelGroup**,
int **rateCounter**,
unsigned char **coincidenceMask**);

Arguments

Name	Description
fourChannelGroup	Number of the group of 4 front-end channels in which the rate counter is located (system dependent)
rateCounter	Counter number inside the group (0 to 3)
coincidenceMask	4-bit pattern. Each bit corresponds to an input channel of the above selected group: bit0 => ch0, bit1 => ch1, bit2 => ch2, bit3 => ch3. A

	bit set to 1 means the channel is active in the coincidence. In order to see only the hit rate of a given channel, set to 1 only the bit with the same channel and counter values.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetRateCounterCoincidenceMask

Description
This function returns the current value of the coincidence mask for the given rate counter inside the given group of 4 channels.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetRateCounterCoincidenceMask (
int **fourChannelGroup**,
int **rateCounter**,
unsigned char ***coincidenceMask**);

Arguments

Name	Description
fourChannelGroup	Number of the group of 4 front-end channels in which the rate counter is located (system dependent)
rateCounter	Counter number inside the group (0 to 3)
coincidenceMask	4-bit pattern. Each bit corresponds to an input channel of the above selected group: bit0 => ch0, bit1 => ch1, bit2 => ch2, bit3 => ch3. A bit set to 1 means the channel is active in the coincidence.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_StartRateCounters

Description
This function permits starting all the counters used for calculating the rates inside the given group of 2 channels. Its use is forbidden during standard waveform acquisition runs.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_StartRateCounters(
WAVECAT64CH_ChannelType **channelType**,
int **twoChannelGroup**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
twoChannelGroup	Number of the group of 2 channels for the given channelType in which the rate counter is located (system dependent)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_ReadRateCounters

Description
This function returns the current value of all the counters used for calculating the rates inside the given group of 2 channels. These rates depend on the coincidences defined by the WAVECAT64CH_SetRateCounterCoincidenceMask function. The value of these counters is sent together with waveform events, but they can also be read independently.
In order to calculate a rate, two counters are necessary: one is counting the hits (HitCounter over 16 bits) since the last event was read, while the other (TimeCounter over 24 bits) is counting the time in units of 32000 sampling intervals or 200 clock periods (this corresponds for instance to 1 µs at 3.2 GS/s). Reading their values stops both of them, as well as the possible saturation of any of them. Therefore, a **start command is necessary to launch the counting** (via the WAVECAT64CH_StartRateCounters function).
The rate is independent from the number of events read per second.
Both starting and reading the rate counters is forbidden during standard acquisition runs of waveforms, but it can permit performing specific runs.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ReadRateCounters(
WAVECAT64CH_ChannelType **channelType**,
int **twoChannelGroup**,
unsigned short ***hitCounterCh0**,
unsigned short ***hitCounterCh1**,
int ***timeCounterCh0**,
int ***timeCounterCh1**);

Arguments

Name	Description
channelType	Type of the channel to enable/disable: WAVECAT64CH_FRONT_CHANNEL or WAVECAT64CH_BACK_EXTRA_CHANNEL
twoChannelGroup	Number of the group of 2 channels for the given channelType in which the rate counter is located (system dependent)
***hitCounterCh0**	Pointer to the hit counter of channel0
***hitCounterCh1**	Pointer to the hit counter of channel1
***timeCounterCh0**	Pointer to the time counter of channel0
***timeCounterCh1**	Pointer to the time counter of channel1

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_EnableRateCounterVeto

Description
This function enables the gated veto of all the rate counters in the system. This veto occurs on a channel basis after any increment of the corresponding hit counter, in order to avoid counting potential parasitic hits. The veto gate length is defined by the WAVECAT64CH_SetRateCounterVetoLength function.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_EnableRateCounterVeto (void);

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_DisableRateCounterVeto

Description
This function disables the gated veto of all the rate counters in the system.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_DisableRateCounterVeto (void);

Return Value
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetRateCounterVetoLength

Description
This function sets the value in µs of the gate individually triggered channel by channel by any increment of the corresponding hit counter. Values shall range between 0.015 and 327.675 µs.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ SetRateCounterVetoLength (
float **vetoLength**);

Arguments

Name	Description
vetoLength	Veto duration in µs

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetRateCounterVetoLength

Description
This function returns the value in µs of the gate individually triggered channel by channel by any increment of the corresponding hit counter.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ GetRateCounterVetoLength (
float ***vetoLength**);

Arguments

Name	Description
***vetoLength**	Veto duration in µs

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.9 Time Base Parameters

WAVECAT64CH_SetSamplingFrequency

Description
This function sets the sampling frequency of the system.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_SetSamplingFrequency(
WAVECAT64CH_SamplingFrequencyType **samplingFrequency**);

Arguments

Name	Description
samplingFrequency	Sampling frequency. There are 8 possible values: - WAVECAT64CH_3_2GHZ, - WAVECAT64CH_2_13GHZ, - WAVECAT64CH_1_6GHZ, - WAVECAT64CH_1_28GHZ, - WAVECAT64CH_1_07GHZ, - WAVECAT64CH_800MHZ, - WAVECAT64CH_533MHZ, - WAVECAT64CH_400MHZ

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetSamplingFrequency

Description
This function returns the current value of the system's sampling frequency.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_GetSamplingFrequency(
WAVECAT64CH_SamplingFrequencyType ***samplingFrequency**);

Arguments

Name	Description
***samplingFrequency**	Sampling frequency returned by the function.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetRecordingDepth

Description
This function sets the recording depth (number of samples) of the system.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_SetRecordingDepth(
unsigned short **recordingDepth**);

Arguments

Name	Description
recordingDepth	The number of samples recorded ranging between 64 and 1024. This has to be a multiple of 16. Otherwise, the function will return the error code: WAVECAT64CH_InvalidParam

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetRecordingDepth

Description

This function returns the recording depth (number of samples) of the system.

Synopsis

WAVECAT64CH_ErrCode WAVECAT64CH_GetRecordingDepth(
unsigned short ***recordingDepth**);;

Arguments

Name	Description
***** **recordingDepth**	Recording depth returned by the function.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetReadoutLatency

Description

This function sets the readout latency of the system. Readout and ADC conversion can be performed either at 10 MHz (standard mode) or at 20 MHz (short mode). The short mode reduces the latency by a factor close to 2 but the noise then increases by a few %.

Synopsis

WAVECAT64CH_SetReadoutLatency(
WAVECAT64CH_ReadoutLatencyType **readoutLatency**);

Arguments

Name	Description
readoutLatency	Readout latency, to be chosen between WAVECAT64CH_STANDARD_READOUT and WAVECAT64CH_SHORT_READOUT

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetReadoutLatency

Description
This function returns the current value of the system's readout latency.

Synopsis
WAVECAT64CH_GetReadoutLatency(
WAVECAT64CH_ReadoutLatencyType ***readoutLatency**);

Arguments

Name	Description
***readoutLatency**	Readout latency returned by the function.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_SetExternalClockOption

Description
This function enables or disables the external clock option. Said external clock has to be sent to the corresponding front-panel input at a frequency of 200 MHz.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ SetExternalClockOption (
WAVECAT64CH_StateType **externalClockOptionState**);

Arguments

Name	Description
externalClockOptionState	State of the external clock option: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_GetExternalClockOption

Description
This function returns the current state (ON/OFF) of the external clock option (see just above).

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ GetExternalClockOption (
WAVECAT64CH_StateType ***externalClockOptionState**);

Arguments

Name	Description
***externalClockOptionState**	State of the external clock option: ON (WAVECAT64CH_STATE_ON) or OFF (WAVECAT64CH_STATE_OFF)

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

4.10 Acquisition

WAVECAT64CH_AllocateEventStructure

Description
This function allocates the necessary memory for the event structure. It has to be used before launching a run, right before WAVECAT64CH_StartRun. The size of the structure depends on the system.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_AllocateEventStructure (WAVECAT64CH_EventStruct ***eventStruct**);

Arguments

Name	Description
***eventStruct**	Event structure as described in chapter 3.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_StartRun

Description
This function has to be called at the beginning of a run before going into the acquisition loop. **Important! Once the acquisition is running, accessing the board to change a parameter is not allowed!** In that case, the associated library functions return the error code WAVECAT64CH_ErrorAcquisitionRunning.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_StartRun(void);

Arguments
No arguments

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_PrepareEvent

Description
This function must be used inside the acquisition loop to prepare the readout of every event. For instance, it automatically sends the software trigger to the system if the software trigger mode has been selected.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_PrepareEvent (void);

Arguments
No arguments

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_ReadEventBuffer

Description
Located inside the acquisition loop, this functions checks if event data was transmitted from the board. It returns:

- WAVECAT64CH_No_Event until the board triggers
- WAVECAT64CH_ReadoutError if an error occurred
- WAVECAT64CH_IncompleteEvent is event data is still incomplete
- WAVECAT64CH_Success once event data is fully acquired.

It thus has to be called as long as the return code becomes WAVECAT64CH_Success.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_ReadEventBuffer(void);

Arguments
No arguments

Return Values
0: Success; Negative numbers are error codes (see function description right above and Return Codes).

WAVECAT64CH_DecodeEvent

Description
This function decodes the current event (read by WAVECAT64CH_ReadEventBuffer) and fills up the event structure.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_DecodeEvent(
WAVECAT64CH_EventStruct ***eventStruct**);

Arguments

Name	Description
***eventStruct**	Event structure as described in chapter 3.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_StopRun

Description
This function must be called at the end of a run, once out of the acquisition loop.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_StopRun(void);

Arguments
No arguments

Return Values
0: Success; Negative numbers are error codes (see Return Codes).

WAVECAT64CH_FreeEventStructure

Description
This function frees the memory allocated for the event structure by WAVECAT64CH_AllocateEventStructure. It has to be used at the end of a run, right after WAVECAT64CH_StopRun.

Synopsis
WAVECAT64CH_ErrCode WAVECAT64CH_FreeEventStructure (
WAVECAT64CH_EventStruct ***eventStruct**);

Arguments

Name	Description
***eventStruct**	Event structure opened by WAVECAT64CH_AllocateEventStructure.

Return Values
0: Success; Negative numbers are error codes (see Return Codes).