Contents

VI10 SDK brief description

Platform specification

SDK is designed for MS VisualStudio 2015 (v14), MFC

SDK structure

Vibraimage1.sln	MS VisualStudio 2015 project file
run	Output binaries directory
VibraimageEx	Vibraimage10 Pro source code (example)
inc	SDK C++ headers
inc\AMCap.h	Audio/Video stream, decoding and camera connection classes
inc\default_path.h	GetDefaultPath() function declaration
inc\estring.h	Extended string functions
inc\inc.h	Increment tool class
inc\regdelete.h	Registry functions
inc\safe_delete.h	Memory tools
inc\VIEngine.h	SDK core functions/class
inc\ViEngineExSDK.h	SDK base functions/classes
inc\viengineexsdk_id.h	SDK constants
inc\viFrameVar.h	SDK constants for handling base parameters
inc\VIVar.h	SDK core types
inc\VIVarTag.h	SDK constants for handling core parameters
lib	SDK build libraries
lib\viamcap.lib	
lib\ViEngine.lib	
lib\ViEngineEx.lib	
lib\VISoundDll.lib	
share	SDK additional includes and sources
share\Cmdl.cpp	Command line parser (code)
share\Cmdl.h	Command line parser (header)
share\ColorProcress.cpp	Simple dialog-based item (code)
share\ColorProcress.h	Simple dialog-based item (header)
share\NewVarEvent.cpp	VI Event notifier base class (code)
share\NewVarEvent.h	VI Event notifier base class (header)
share\pugiconfig.hpp	PUGI XML library (header)
share\pugixml.cpp	PUGI XML library (code)
share\pugixml.hpp	PUGI XML library (header)
share\pugixpath.cpp	PUGI XML library (code)

VibraimageEx source code

This project contains full source code of Vibraimage10 Pro GUI application. You can use it as base or example for your Vibraimage-based project. Also you can modify this code to add new features and build new executable.

Configuration files

The GUI of Vibraimage10 Pro is defined by several XML-based files located at "\run\config\" directory.

You can modify, add or delete this files to make anew view of application.

Language configuration

All text strings are located in XML-based files "\run\lang\??\ vibraimage.xml ". You can edit this files to change text in Vibraimage10 Pro GUI application.

SDK base classes and interfaces

File: inc\ViEngineExSDK.h

Namespace: `CViEngineExNS`

Quick start

1. Implement main SDK class

```
CViEngineEx m_engine;
```

2. Implement your interfaces

```
class CViEngineExProcReg :      public CViEngineExRegInterface
{
public:
....
}

....
class CViEngineExProc :   public CViEngineExBaseInterface
{
public:
....
}

CViEngineExProc           m_engineInterface;
CViEngineExProcReg        m_engineReg;
```

3. Connect interfaces

```
m_engineInterface.Connect(&m_engine);
m_engineReg.Connect(&m_engine);
```

4. Start engine

```
m_engine.Start();
```

class CViEngineEx

```
class VIENGINEEX_API CViEngineEx
{
public:
      void *m_pApp; // Reference to internal core storage
public:
```

```
        CViEngineEx();      // default constructor
        ~CViEngineEx(); // default destructor
public:
        ////////////////////////////////////////////////////////////////////////
        // OpenDocument:
        //     Open specified source file
        // Parameters:
        //     path:
        //            1) AVI file path like "D:\test.avi"
        //            2) Video source like "*VIDEO:*menu*1", "*VIDEO:My Camera 1"
        //            3) LDQ file like "D:\Child1.ldq"
        // Return value: retruns 0
        ////////////////////////////////////////////////////////////////////////
        int          OpenDocument(LPCWSTR path);

        ////////////////////////////////////////////////////////////////////////
        // GetCurrentDocument:
        // Receive last opened video source
        // Parameters:
        //     path: pointer to string
        //     cch: the size of string
        // Returns "true" if succeded
        ////////////////////////////////////////////////////////////////////////
        bool   GetCurrentDocument(LPWSTR path, int cch);

        ////////////////////////////////////////////////////////////////////////
        // Start
        // Prform initialization and start engine
        // Parameters:
        //     hWndParent: handle to parent window
        // Returns: handle to engine window
        ////////////////////////////////////////////////////////////////////////
        HWND   Start(HWND hWndParent = 0);

        ////////////////////////////////////////////////////////////////////////
        // Stop:
        //     Stop engine anl clear resources.
        ////////////////////////////////////////////////////////////////////////
        void   Stop();
public:
        ////////////////////////////////////////////////////////////////////////
        // GetVersion:
        // Returns engine version. Currently 0
        ////////////////////////////////////////////////////////////////////////
        unsigned int GetVersion(int id);

        ////////////////////////////////////////////////////////////////////////
        // Return configuration folder. By default returns "config".
        // It means <Applications folder>\config
        ////////////////////////////////////////////////////////////////////////
        LPCWSTR GetConfigBase();

        ////////////////////////////////////////////////////////////////////////
        // GetVar:
        // Returns parameter with specified ID, subID ant vt
        ////////////////////////////////////////////////////////////////////////
        COleVariant GetVar(int id, int subID, VARTYPE vt, int nVar = 1);

        ////////////////////////////////////////////////////////////////////////
        // PutVar:
        // Set parameter value with specified ID, subID ant vt
        ////////////////////////////////////////////////////////////////////////
        bool   PutVar(int id, int subID, const COleVariant& v, int nVar = 1);

        ////////////////////////////////////////////////////////////////////////
```

```
        // UpdateColor:
        // Get default color for parameter's menu with specified ID, subID
        // Returns "true" if succeded
        ////////////////////////////////////////////////////////////////////////
        bool   UpdateColor(int id, int subID, int ref, COLORREF *pColor);


        ////////////////////////////////////////////////////////////////////////
        // OnMenu:
        //     Perform menu action with specified ID
        // Returns "true" if succeded
        ////////////////////////////////////////////////////////////////////////
        bool   OnMenu(int id);

        ////////////////////////////////////////////////////////////////////////
        // GetMenuState:
        //     Get menu item state and text with specified ID
        // Returns "true" if succeded
        ////////////////////////////////////////////////////////////////////////
        bool   GetMenuState(int id, BOOL& bEnabled,
                BOOL& bChecked, LPWSTR pText = 0);


        ////////////////////////////////////////////////////////////////////////
        // GetWindow:
        //     Returns: handle to main window
        ////////////////////////////////////////////////////////////////////////
        HWND   GetWindow();
public:

        ////////////////////////////////////////////////////////////////////////
        // Tag2Id:
        // Convert string sID to integer ID
        // Returns: ID of specified parameter
        // Example Tag2Id("VI_VAR_SIZE") returns 3
        ////////////////////////////////////////////////////////////////////////
        int Tag2Id(LPCWSTR sid);

public:

        ////////////////////////////////////////////////////////////////////////
        // CAMCapWnd:
        // Returns video capture interface
        ////////////////////////////////////////////////////////////////////////
        CAMCapWnd *   GetVideo();

        ////////////////////////////////////////////////////////////////////////
        // GetEngine:
        // Returns VIEngine core interface
        ////////////////////////////////////////////////////////////////////////
        CViEngine *   GetEngine();


        ////////////////////////////////////////////////////////////////////////
        // CViEngineExGraphInterface:
        // Returns graph window interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExGraphInterface*              GetGraph();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExHistInterface:
        // Returns histogram window interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExHistInterface*               GetHist();
```

```
        ////////////////////////////////////////////////////////////////////////
        // CViEngineExMeasureInterface:
        // Returns measurement (M mode) interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExMeasureInterface*            GetMeasure();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExLDWInterface:
        // Returns LD window (LD mode) interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExLDWInterface*                GetLDW();


        ////////////////////////////////////////////////////////////////////////
        // CViEngineExStatInterface:
        // Returns stat window interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExStatInterface*               GetStat();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExLangInterface:
        // Returns language translation interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExLangInterface*               GetLang();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExRegInterface:
        // Returns registry storage interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExRegInterface*                GetReg();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExAutoCaptureInterface:
        // Returns image acpture window interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExAutoCaptureInterface*  GetCapture();

        ////////////////////////////////////////////////////////////////////////
        // CViEngineExDatabaseInterface:
        // Returns database storage interface
        ////////////////////////////////////////////////////////////////////////
        CViEngineExDatabaseInterface*           GetDB();
};
```

SDK Core

Vibraimage10 SDK based on Vibraimage8 SDK core

Function

This SDK is designed to create applications using the technology Vibraimage. SDK is intended for use jointly with a programming language «C + +» among (in the) «MS Windows».

Mechanism of the interaction

All operations with Vibraimage module are carried out through the instance VIEngine. Video / audio data is supplied from the user program to the module in the form of uncompressed frames. Video / audio capture produced by means of the user program.

Pre-starting procedure (Preparation for work)

It is necessary to set the path to the Windows registry using the interaction function `CVIEngine::SetRegKey, CVIEngine::SetCallbackOnNewVar, CVIEngine::SetCallbackOnImg8` before the module starting .

*Note: There is no need to make it if CViEngineEx class is used.

Starting and stopping of the module

Starting and stopping are made through functions `CVIEngine::Start, CVIEngine::Stop`

*Note: There is no need to make it if CViEngineEx class is used.

Events

Event handling mechanism is realized through the use of «CALLBACK» functions.

Functions

CVIEngine(int nThread=-1);

The object constructor CVIEngine.

Parameters:

int nThread

Determines the flows number that will be broken into the main processing. Valid values [1-8], -1 means that the number of threads is determined automatically depending on the number of processors in the system.

~CVIEngine(void);

Object destructor CVIEngine.

bool AddImage(void p, int w, int h, int bpp,double t);*

Add function another source video frame for processing. Invoked whenever the video frame from the source enters to the program.

Parameters:

*void *p* A pointer to the memory block of size *w*h*bpp/8* containing w * h dots

int w Image width. The value must be a multiple of 4

int h Image height.

int bpp The number of bits per pixel. Valid values are [24,8]

double t Frame arrival time in seconds.

Returns *true*, if the operation was successful.

bool AddSound(WAVEFORMATEX pfmt, double dblT, void* pBuffer, int lSize);*

A function of adding another audio source frame for processing. Invoked whenever audio frame from a source comes to the program.

Parameters:

WAVEFORMATEX pfmt* The format of the audio data. The recommended format: *PCM 44100 16bit mono*

Double dblT	Frame arrival time in seconds.
*void *pBuffer*	A pointer to the memory block containing *lSize* data bytes.
lSize	Data block size.

Returns *true*, if the operation was successful.

void Sync(void);

Function expects the moment of completion of a cycle of the main processing of a frame.

bool SetResultPtr(int id,void ptr, int w, int h);*

*bool GetResultPtr(int id,void ** ptr, int* pw, int* ph);*

Functions of installation and obtaining information about result image sizes and location.

Parameters:

int id	Image identifier (enum VI_MODE_RESULT_VARS)
*void *ptr*	A pointer to the memory block w*h*32bit
int w,int p	Image size. Image size must be identical to the size of the input frame.

Returns *true*, if the operation was successful.

DWORD GetResultVer(void);

The function returns counter value of processed images.

void LockResult(bool bLock, void* pLock);*

Function to create the temporary blocking of the image results.

During lock action image data won't be changed by the module.

Parameters:

bool bLock	True - to start lock, false - to stop lock
*void *pLock*	Object of lock for completion

Example:

```
void *lock = m_engine.LockResult(true,0); // start lock

// saving/modification data

m_engine.LockResult(false,lock); //stop lock
```

bool GetSrcLine(int x, int y, float px=0, float* py=0);*

Function passes the "slices" of the current image with the given coordinates.
To invoke this function parameter VI_MODE_RESULT must correspond to a single image.

Parameters:

int x,int y Coordinates of the "slices" point.

*float *px,float* py* A pointer to the memory block, where the horizontal and vertical "slices" will be saved. The size of the blocks must correspond to the size of the frame.

Returns *true*, if the operation was successful.

bool GetSumHist(int id, float px=0, float* py=0);*

The function passes averaged horizontal and vertical point values vibraimage.

Parameters:

int id Image identifier (enum VI_MODE_RESULT_VARS)

*float *px,float* py* A pointer to the memory block, where the horizontal and vertical blocks will be saved. The size of the blocks must correspond to the size of the frame.

Returns *true*, if the operation was successful.

*void SetRegKey(void *strW);*

The function sets the path in the registry Windows, on which the module will read and write its settings.

Example:

```
m_engine.SetRegKey(L"SOFTWARE\\ELSYS\\Vibraimage70_SDK_Sample\\engine");
```

void Start(void);

Module start.

void Stop(void);

Module stop.

void Pause(bool bSet);

Pause flag activation. Equivalent *PutI1(VI_FILTER_PAUSE,bSet?1:0);*

void Reset(void);

Reset flag activation. Equivalent *PutI1(VI_VAR_RESET,1);*

void GetI(int id,LONG &v1, LONG& v2);

void GetI(int id,int &v1, int& v2);

void GetF(int id,float &v1, float& v2);

int& GetI1(int id);

int& GetI2(int id);

float& GetF1(int id);

float& GetF2(int id);

VI_VAR& GetVar(int id);

Functions return a value with the specified identifier *id (enum VI_VAR_ENGINE).*

void PutVar(int id,const VI_VAR& v);

void PutI1(int id,int v);

void PutI2(int id,int v);

void PutF1(int id,float v);

void PutF2(int id,float v);

void PutI(int id,int v1,int v2);

void PutF(int id,float v1,float v2);

Functions set a value with the specified identifier *id (enum VI_VAR_ENGINE).* When these functions are called event *CallbackOnNewVar* occurs.

LPCWSTR GetStr(int id);

The function returns value of the string parameter with the specified identifier *id (enum VI_VAR_ENGINE).*

void PutStr(int id, LPCWSTR str);

The function returns value of the string parameter with the specified identifier *id (enum VI_VAR_ENGINE).* When these functions are called event *CallbackOnNewVar* occurs.

void SetCallbackOnNewVar(void * pFn, void * pData);

The function sets the event handler *CallbackOnNewVar*, which occurs with any change in the module parameter.

Parameters:

*void * pFn*	A pointer to the function of the type of
*[static void CallbackOnNewVar(void *pUserData,int id,int subID);]*	
*void *pData*	User data
*void *pUserData*	User data

int id Parameters identifier

int subID It is not used in this version

*void SetCallbackOnImg8(void * pFn, void * pData);*

The function sets the event handler CallbackOnImg8, which occurs when receiving b / w images during a call *AddImage*.

Parameters:

*void * pFn* A pointer to the function of the type of

*[static void CallbackOnVideo8(void *pUserData,BYTE* i8,int w,int h,double t);]*

*void *pData* User data

bool SeqIsOk(void);

The function returns the state of the protection module - key presence.

bool SeqIsDemo(void);

The function returns a flag «Demo» of the module protection.

bool SeqIsLimit(void);

The function returns a flag « Limit» of the module protection.

bool SeqIsInit(void);

The function returns the state of the protection module - readiness to work.

LPCWSTR SeqSerial(void);

The function returns the serial number of the key.

LPCWSTR SeqOwner(void);

The function returns the text identifier of the key.

LPCWSTR SeqLimit(void);

The function returns a constraint row of the key.

LPCWSTR Tag(int id);

The function returns the text identifier of the parameter from *(enum VI_VAR_ENGINE).*

int GetID(LPCWSTR tag);

The function returns the number identifier of the parameter from *(enum VI_VAR_ENGINE).*

*float GetAVG(int id,int *pCnt=0);*

The function returns the average value.

Parameters:

int id Identifier (enum *VI_VAR_ENGINE*)

*int *pCnt* The pointer according to which the number of averagings will be saved

void PutAVG(int id,int cnt);

The function sets the number of averagings.

Parameters:

int id	Identifier (enum *VI_VAR_ENGINE*)
int cnt	The number of averagings

void PutAVG(int id,int cnt,float v);

The function sets the number of averages and start value.

Parameters:

int id	Identifier (enum *VI_VAR_ENGINE*)
int cnt	The number of averagings
float v	Start value

*bool GetStat(int id,float *cMin, float *cMax, float *bMin, float *bMax,float *rate);*

The function returns the status value for the «LD» mode parameter.

Parameters:

int id	Identifier (enum *VI_VAR_ENGINE*)
*float *cMin*	The current value of the parameter minimum.
*float *cMax*	The current value of the parameter maximum.
*float *bMin*	The basic value of the parameter minimum.
*float *bMax*	The basic value of the parameter maximum.
*float *rate*	The result of the checks.

Returns *true*, if the operation was successful.

bool GetStatCalc(int id);

The function returns *true*, f the parameter involved in the LD calculation.

float GetStatRate1(int id);

float GetStatRate2(int id);

Functions return current values of the multipliers for the *id* parameter in calculating LD.

float GetStatLevel(int id);

The function returns threshold of exceeding of the current parameters over the basic for the *id* parameter in calculating LD.

float GetStatRate1def(int id);

float GetStatRate2def(int id);

Functions return multipliers values by default for the *id* parameter in calculating LD.

float GetStatLevelDef(int id);

The function returns threshold of exceeding by default for the *id* parameter in calculating LD.

int GetStatHist(int id, int pHist256=0,float * pFPS=0);*

The function returns a histogram with the specified id.

Parameters:

int id Identifier (enum *VI_VAR_ENGINE*)

Valid values:

[VI_VAR_HIST_N_A0…VI_VAR_HIST_F_B2],(VI_STAT_START…VI_STAT_END)

*int *pHist256* Data block of 256 elements in size for reception of the histogram

*float *pFps* Frame frequency for which the histogram is calculated.

It returns the current frame counter.

int GetStatHistFFT(int id, float pHist256=0,float * pFPS=0);*

The function returns a histogram with the specified id. For valid id - this is an analogue function GetStatHist, but histogram is transmitted as float

Parameters:

int id Identifier (enum *VI_VAR_ENGINE*)

Valid values: (VI_STAT_START…VI_STAT_END)

*float *pHist256* Data block of 256 elements in size for histogram reception

*float *pFps* The frame frequency for wich a histogram was calculated.

Return current frame counter.

void PutStatCalc(int id, bool bCalc);

The function sets participation flag the *id* parameter in calculating LD.

void PutStatRate1(int id, float v);

void PutStatRate2(int id, float v);

Functions sets current values of the multipliers for the *id* parameter in calculating LD.

void PutStatLevel(int id, float v);

The function sets threshold of exceeding of the current parameters over the basic for the *id* parameter in calculating LD.

void Lock(int bLock);

The function returns the state of the lock flag. If the flag is set - current calculations are interrupted and new are ignored.

bool IsLocked();

The function returns the state of the lock flag.

void RegSave(void);

The function makes saving the current settings in the register. The way in the register is set by the function SetRegKey.

RGBQUAD GetPal(void);*

The function returns a pointer to 256 elements of the current palette.

void PutPal(RGBQUAD pal);*

The function sets new 256 colors of a palette.

int Tag2Id(LPCWSTR tag);

An analogue function GetID.

Basic constants

VI_VAR_SIZE

The current frame size.

Used fields: i1,i2

VI_VAR_K

The parameter «K» value

Used fields: f1

VI_VAR_TH

The parameter «L» value

Used fields: f1

VI_VAR_N0

The current parameter «N» value for the calculation block on N

Used fields: i1

To change the value to use VI_VAR_N0_RQST

VI_VAR_N1

The current parameter «N» value for the calculation block on 10

Used fields: i1

To change the value to use VI_VAR_N1_RQST

VI_VAR_N2

The current parameter «N» value for the calculation block on 2

Used fields: i1

To change the value to use VI_VAR_N2_RQST

VI_VAR_N0_RQST

Interface for setting VI_VAR_N0

The works mechanism: the user sets value and module changes its configuration in case of arrival of a new frame.

VI_VAR_N1_RQST

Interface for setting VI_VAR_N1

The works mechanism: the user sets value and module changes its configuration in case of arrival of a new frame.

VI_VAR_N2_RQST

Interface for setting VI_VAR_N2

The works mechanism: the user sets value and module changes its configuration in case of arrival of a new frame.

VI_MODE_RESULT

Flags value constructing images results. Image is constructed, if the appropriate flag is set.

Used fields: i1

Values:

VI_RESULT_SRC_A	Original BW image with aura for AM
VI_RESULT_VI0_A	Vibraimage N AM
VI_RESULT_VI1_A	Vibraimage 10 AM
VI_RESULT_VI2_A	Vibraimage 2 AM
VI_RESULT_DELTA_A	Interframe difference AM
VI_RESULT_SRC_B	Original BW image with aura for F
VI_RESULT_VI0_B	Vibraimage N F
VI_RESULT_VI1_B	Vibraimage 10 F
VI_RESULT_VI2_B	Vibraimage 2 F
VI_RESULT_DELTA_B	Interframe difference F
VI_RESULT_SRC_0	Original BW image

VI_MODE_AURA
Flags aura overlay image value. Aura is constructed, if the appropriate flag is set.

Used fields: i1

The values correspond VI_MODE_RESULT.

VI_FILTER_DELTA_LO
Value «Extended filter» for basic processing

Used fields: f1

VI_FILTER_DELTA_LOF
Value «Extended filter» for fast processing

Used fields: f1

VI_FILTER_DELTA_STRETCH
Useage flag «Stretch filter»

Used fields: i1

VI_FILTER_DELTA_MSCONT
Useage flag «Max speed filter»

Used fields: i1

VI_FILTER_SP
Useage flag «Single points filter»

Used fields: i1

VI_FILTER_AM
Value «Am scale filter»

Used fields: f1

VI_FILTER_CT
Value «Space filter»

Used fields: f1

VI_FILTER_MOTION
Useage flag «Motion detector»

Used fields: i1

VI_FILTER_MOTION_10X
Useage flag «Motion detector» for 10 frame

Used fields: i1

VI_FILTER_MOTION_LEVEL
Value «level» for «Motion detector»

Used fields: f1

VI_FILTER_MOTION_LEVEL2
Value «level2» for «Motion detector»

Used fields: f1

VI_FILTER_MOTION_SET
Activity flag «Motion detector»

Used fields: i1

VI_FILTER_MOTION_AUTO_RESET
Activity flag auto reset for «Motion detector»

Used fields: i1

VI_FILTER_NSKIP
Value «skip frames» for «Motion detector»

Used fields: i1

VI_VAR_NFRAME
Counter value of the processed frames

Used fields: i1

VI_VAR_NFRAME_IN
Counter value of the received frames

Used fields: i1

VI_VAR_FPS_PERIOD
Duration of a period time frame for which picture frequency is calculated in seconds

Used fields: f1

VI_VAR_FPS_BUFFER_SIZE
The size of the internal queue of frames for processing

Used fields: i1

VI_VAR_FPSIN
Input frame frequency

Used fields: f1

VI_VAR_FPSOUTF
Frame frequency of fast processing

Used fields: f1

VI_VAR_FPSOUTR
Frame frequency of basic processing

Used fields: f1

VI_VAR_FPSMAXF
Limitation frame frequency of fast processing

Used fields: f1

VI_VAR_FPSMAXR
Limitation frame frequency of basic processing

Used fields: f1

VI_FILTER_FPS2IN
Filters value «downrate»

Used fields: i1

VI_FILTER_PAUSE
Suspension of work flag

Used fields: i1

VI_VAR_AUDIO_TH
Value «Audio Threshold»

Used fields: f1

VI_VAR_RESET
Reset vibraimage flag

Used fields: i1

VI_MODE_WBG
Flags of use of a white background for vibroimages. Values correspond

VI_MODE_RESULT

Used fields: i1

VI_MODE_COLOR
Useage flag of the color original image instead of BW.

Used fields: i1

VI_VIRTUAL_POS_ENABLE
Useage flag of the virtual cursor

Used fields: i1

VI_VIRTUAL_POS_INTEGR_LEVEL
Threshold value of the virtual cursor

Used fields: f1

VI_VIRTUAL_POS_POS
Absolute coordinates of the virtual cursor

Used fields: i1,i2

VI_VIRTUAL_POS_POS_LOCAL
Relative coordinates of the virtual cursor

Used fields: i1,i2

VI_VIRTUAL_POS_SIZE
The size of the virtual cursor movement

Used fields: i1,i2

VI_FILTER_DISABLE_A
Flag of switch-off of calculation for AM

Used fields: i1

VI_FILTER_DISABLE_B
Flag of switch-off of calculation for F

Used fields: i1

VI_FILTER_DISABLE_2X
Flag of switch-off of fast calculation

Used fields: i1

VI_FILTER_DISABLE_VI0
Flag of switch-off of calculation for N frames

Used fields: i1

VI_FILTER_DISABLE_VI1
Flag of switch-off of calculation for 10 frames

Used fields: i1

VI_FILTER_DISABLE_VI2
Flag of switch-off of calculation for 2 frames

Used fields: i1

VI_FILTER_DISABLE_FFT_UNUSED
Flag of switch-off of the calculation frequency processing of Fourier unused parameters

Used fields: i1

VI_VAR_STATE_VAR
Value of the hazard degree of the object

Used fields: f1

VI_VAR_STATE_CRITICAL
Value of the critical level of danger of the object

Used fields: f1

VI_VAR_STATE_FLAG_A
Resettable hazard flag

Used fields: i1

VI_VAR_STATE_FLAG_P
Not resettable hazard flag

Used fields: i1

VI_VAR_LD_PERIOD
Value «LD Stat period (s)»

VI_VAR_LD_ENABLE
Useage flag LD

Used fields: i1

VI_VAR_LD_LTH
Value «LD Lie treshold »

Used fields: f1

VI_VAR_LD_MODE
Flag of the LD mode

Used fields: i1

Values: *VI_MODE_LD_AUDIO, VI_MODE_LD_MANUAL, VI_MODE_LD_AUTO*

VI_VAR_LD_STARTED
Activity flag LD

Used fields: i1

VI_VAR_HIST_N_A0
Histogram identifierH AM N. It is used in *GetStatHist*

VI_VAR_HIST_N_B0
Histogram identifierH F N. It is used in *GetStatHist*

VI_VAR_HIST_N_A1
Histogram identifierH AM 10. It is used in *GetStatHist*

VI_VAR_HIST_N_B1
Histogram identifierH F 10. It is used in *GetStatHist*

VI_VAR_HIST_N_A2
Histogram identifierH AM 2. It is used in *GetStatHist*

VI_VAR_HIST_N_B2
Histogram identifierH F 2. It is used in *GetStatHist*

VI_VAR_HIST_C_A0
Histogram identifierC AM N. It is used in *GetStatHist*

VI_VAR_HIST_C_B0
Histogram identifierC F N. It is used in *GetStatHist*

VI_VAR_HIST_C_A1
Histogram identifierC AM 10. It is used in *GetStatHist*

VI_VAR_HIST_C_B1
Histogram identifierC F 10. It is used in *GetStatHist*

VI_VAR_HIST_C_A2
Histogram identifierC AM 2. It is used in *GetStatHist*

VI_VAR_HIST_C_B2
Histogram identifierC F 2. It is used in *GetStatHist*

VI_VAR_HIST_F_A0
Histogram identifierS AM N. It is used in *GetStatHist*

VI_VAR_HIST_F_B0
Histogram identifierS F N. It is used in *GetStatHist*

VI_VAR_HIST_F_A1
Histogram identifierS AM 10. It is used in *GetStatHist*

VI_VAR_HIST_F_B1
Histogram identifierS F 10. It is used in *GetStatHist*

VI_VAR_HIST_F_A2
Histogram identifierS AM 2. It is used in *GetStatHist*

VI_VAR_HIST_F_B2
Histogram identifierS F 2. It is used in *GetStatHist*

VI_VAR_HIST_FFT_READY
Event-indicator of readiness of FFT processing

VI_VAR_HIST_CX_AVG_A0
The center of the image AM N

Used fields: i1,i2,f1,f2

VI_VAR_HIST_CX_AVG_A1
The center of the image AM 10

Used fields: i1,i2,f1,f2

VI_VAR_HIST_CX_AVG_A2
The center of the image AM 2

Used fields: i1,i2,f1,f2

VI_VAR_HIST_CX_AVG_B0
The center of the image F N

Used fields: i1,i2,f1,f2

VI_VAR_HIST_CX_AVG_B1
The center of the image F 10

Used fields: i1,i2,f1,f2

VI_VAR_HIST_CX_AVG_B2
The center of the image F 2

Used fields: i1,i2,f1,f2

VI_VAR_STAT_INTEGR2A
The integrated intensity of the vibraimage for 2 AM

Used fields: f1

VI_VAR_STAT_INTEGR1A
The integrated intensity of the vibraimage for AM 10

Used fields: f1

VI_VAR_STAT_INTEGR0A
The integrated intensity of the vibraimage for AM N

Used fields: f1

VI_VAR_STAT_RES_*
Vibraimage Parameters

Used fields: f1

VI_VAR_STAT_RES_FN01*
User functions

Used fields: f1,str

VI_VAR_AUDIO_LEVEL
Sound level

Used fields: f1

VI_VAR_AUDIO_LEVEL25

Sound level with frequency 25 count./sec

Used fields: f1

VI_VAR_STAT_CFG_SIN

Period VI_VAR_STAT_RES_SIN

Used fields: f1