



## [tONaRT] key detection V2 SDK Documentation

Alexander Lerch

(c) 2006 by zplane.development

December 11, 2006

## Contents

<b>1</b>	<b>[tONaRT] key detection SDK Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	API Documentation . . . . .	1
1.2.1	Naming Conventions . . . . .	1
1.2.2	Required Functions . . . . .	2
1.2.3	Optional Functions . . . . .	2
1.2.4	C++ Usage Example . . . . .	3
1.2.5	Error Codes . . . . .	4
1.3	Background Information . . . . .	4
1.4	MSVC Workspace KeyDetect.dsw . . . . .	4
1.4.1	File Structure . . . . .	4
1.4.2	Project Structure . . . . .	5
1.5	Coding Style minimal overview . . . . .	5
1.6	Licensing Issues . . . . .	6
1.7	Support . . . . .	6
<b>2</b>	<b>[tONaRT] key detection Directory Hierarchy</b>	<b>6</b>
2.1	[tONaRT] key detection Directories . . . . .	6
<b>3</b>	<b>[tONaRT] key detection Data Structure Index</b>	<b>6</b>
3.1	[tONaRT] key detection Data Structures . . . . .	6
<b>4</b>	<b>[tONaRT] key detection File Index</b>	<b>6</b>
4.1	[tONaRT] key detection File List . . . . .	6
<b>5</b>	<b>[tONaRT] key detection Directory Documentation</b>	<b>7</b>
5.1	incl/ Directory Reference . . . . .	7
<b>6</b>	<b>[tONaRT] key detection Data Structure Documentation</b>	<b>7</b>
6.1	CtONaRT_If Class Reference . . . . .	7
6.1.1	Detailed Description . . . . .	7
6.1.2	Member Typedef Documentation . . . . .	8
6.1.3	Constructor & Destructor Documentation . . . . .	8
6.1.4	Member Function Documentation . . . . .	8

## 1 [tONaRT] key detection SDK Documentation 1

---

6.2	CtONaRT_If::Results_t_tag Struct Reference . . . . .	10
6.2.1	Detailed Description . . . . .	10
6.2.2	Field Documentation . . . . .	10
7	[tONaRT] key detection File Documentation . . . . .	11
7.1	docugen.txt File Reference . . . . .	11
7.1.1	Detailed Description . . . . .	11
7.2	tONaRT.h File Reference . . . . .	11
7.2.1	Detailed Description . . . . .	11
7.2.2	Define Documentation . . . . .	12

# 1 [tONaRT] key detection SDK Documentation

## 1.1 Introduction

[tONaRT] key detection is a Software Developers Kit (SDK) that enables the user to automatically determine the key and the concert pitch of any given audio input signal. The technology analyses the pitch content of the audio signal and estimates the key with a sophisticated probability model.

[tONaRT] key detection comes with a C++-style API.

The SDK is available for all common x86 and PPC operating systems. Apart from the delivered libraries, no additional libraries are needed. The SDK is delivered with the needed libraries, the corresponding header files, this documentation and a small example application.

The first part of this document explains the API of the SDK. Then, a small usage example explaining the basic usage of the SDK is given. After some additional notes, some automatically generated documentation follows.

## 1.2 API Documentation

To use the key detection SDK, the library itself (libtONaRT.lib) and the corresponding header file **tONaRT.h** are required. Further header files are not needed. Depending on the development system, several other libraries can be delivered for correct linking.

### 1.2.1 Naming Conventions

Before using the SDK, an **instance** has to be created. After successful creation, the other API functions can be called.

When talking about **frames**, the number of audio samples per channel is meant. For mono signals, the number of frames equals the number of samples. For stereo signals, the number of samples is twice the number of frames. So, in general, the number of

samples equals the number of frames times the number of channels. If the sample size is 16bit, one sample has a memory usage of 2 byte.

### 1.2.2 Required Functions

The following functions have to be called when using the KeyDetect library:

- **int CtONaRT\_If::CreateInstance (CtONaRT\_If \*&phInstanceHandle, int iSampleRate, int iNumberOfChannels, float fStartPitch = 0)**

Creates a new instance of the CtONaRT\_If class; the handle to the new instance is written to parameter phInstanceHandle. To initialize the instance, the input sample rate and the number of audio channels has to be handed over. Optionally, the standard pitch can initialized to something other than 440Hz.

The function returns 0 upon success.

- **int CtONaRT\_If::DestroyInstance (CtONaRT\_If \*&phInstanceHandle)**

Destroys the instance of CtONaRT\_If given in parameter phInstanceHandle; this parameter is set to zero after the destroy.

The function returns 0 upon success.

- **int CtONaRT\_If::Process (float \*pfInputBufferInterleaved, int iNumberOfFrames)**

Does the actual key analysis. The parameter pfInputBufferInterleaved contains the audio data interleaved in floating point format. The parameter iNumberOfFrames contains the number of frames (i.e. samples/number of channels) in the buffer. iNumberOfFrames must not exceed the value of 16384 do to internal memory handling issues.

The process interface is stream based, i.e. it processes one audio block after the other. Therefore, the process function is generally called while there is more audio data available (e.g. from file).

The function returns 0 upon success.

- **int CtONaRT\_If::GetOverallResults (\_stResults\_ \*pstResults)**

Signals the library that the Process function will not be called anymore and writes the results to parameter pstResults after successful processing. The memory pstResults points to has to be allocated by the user.

The function returns 0 upon success.

### 1.2.3 Optional Functions

- **int CtONaRT\_If::GetTempResults (\_stResults\_ \*pstResults)**

Like CtONaRT\_If::GetOverallResults, but returns the current results at this processing point, and does not signal stop of processing.

This function is experimental in the moment and the results may be unreliable.

The function returns 0 upon success.

### 1.2.4 C++ Usage Example

The complete code can be found in the example source file `KeyDetectTestCLMain.cpp`. The example application uses the open source library `libSndFile` for audio file format parsing.

In the first step, a pointer to the needed `CtONaRT_If` instance and a structure for the results have to be declared:

```
// declare instance pointer for key detection
CtONaRT_If      *pCKeyDetectInstance  = 0;

// declare structure for key detection results
CtONaRT_If::Results_t stResult;
```

A new instance can be created with the following code snippet:

```
// create a new instance
if (CtONaRT_If::CreateInstance (   pCKeyDetectInstance,
                                   sfInputInfo.samplerate,
                                   sfInputInfo.channels) != 0)

{
    fprintf(stdout, "Memory Allocation Failed!\n");
    sf_close (pFInputFile);
    return -1;
}
```

In the processing loop, a new block of audio data is read from the file and is handed to the process function for analysis.

```
// read _BLOCKSIZE frames from file
iNumSamplesRead = (int)(sf_readf_float (pFInputFile, afFloatData, _BLOCKSIZE));
// do key analysis
if (pCKeyDetectInstance->Process (afFloatData, iNumSamplesRead) != 0)
    break;
```

The temporary results until this processing block can be evaluated with

```
// get temporary results
//pCKeyDetectInstance->GetTempResults (stResults);
```

Please note that the use of this function is experimental and the results might not be reliable.

After the complete audio data has been handed over to the process function, the final results can be obtained:

```
// get overall results and write them to command line
pCKeyDetectInstance->GetOverallResults (&stResult);
if (!bWriteResult)
    CLShowProcessedTime (clock() - clStartTime);
if (!bWriteResult)
{
    sprintf (acUserOutput, "Pitch: %5.0fHz, Key: %s", stResult.fStandardPitchFrequency, stResult.sKey);
    //sprintf (acUserOutput, "Pitch: %5.0fHz, Key[0]: %s, Key[1]: %s, Key[2]: %s, Key[3]: %s", stResult.fStandardPitchFrequency, stResult.sKey[0], stResult.sKey[1], stResult.sKey[2], stResult.sKey[3]);
}
```

Finally, after successful processing, the instance of [tONaRT] key detection can be destroyed

```
// destroy the instance of tONaRT
CtONaRT_If::DestroyInstance (pCKeyDetectInstance);
```

The above code snippets demonstrated the basic functionality of the tONaRT library. The exact functionality of the functions is described above.

### 1.2.5 Error Codes

where are the errors defined etc.. Below is a short description of possible error codes:

- 0: no error occurred
- 1000001: memory allocation failed
- 5000004: invalid sample frequency
- 5000005: invalid number of channels
- 5000003: invalid function args
- 9999999: an unknown error occurred

## 1.3 Background Information

While [tONaRT] key detection is robust across different musical genres etc., the algorithm expects the input signal to be homogenous, i.e. to contain only one key without (too much) modulations. The standard pitch detection has an accuracy of app. 1-2Hz at 440Hz.

## 1.4 MSVC Workspace KeyDetect.dsw

### 1.4.1 File Structure

**1.4.1.1 Documentation** This documentation and all other documentation can be found in the directory **./doc**.

**1.4.1.2 Project Files** The MS VisualC++-Workspace (.dsw) and all Projectfiles (.dsp) can be found in the directory **./build** and its subfolders, where the subfolders names correspond to the project names.

**1.4.1.3 Source Files** All source files are in the directory **./src** and its subfolders, where the subfolder names equally correspond to the project names.

**1.4.1.4 Include Files** If include files are project intern, they are in the source directory `./src` of the project itself. If include files are to be included by other projects they can be found in `./src/include`. If a SDK-like library/DLL is built for which a header is needed, such a header can be found in `./inc`.

**1.4.1.5 Resource Files** The resource files can be found in the subdirectory `/res` of the corresponding build-directory.

**1.4.1.6 Library Files** The directory `./lib` is for used and built libraries.

**1.4.1.7 Binary Files** The final executable as well as the distributable Dynamic Link Libraries can be found in the directory `./bin`. In debug-builds, the binary files are in the subfolder `/Debug`.

**1.4.1.8 Temporary Files** The directory `./tmp` is for all temporary files while building the projects. In debug-builds, the temporary files can be found in the subfolder `/Debug`.

## 1.4.2 Project Structure

The project structure is as following:

- **KeyDetectTestCL**: small demonstration source code that links and uses the [tONaRT] key detection. The project output is an executable binary (EXE).

For audio file parsing and I/O, the open source library `libSndFile` is used.

## 1.5 Coding Style minimal overview

Variable names have a preceding letter indicating their types:

unsigned:	u
pointer:	p
array:	a
class:	C
bool:	b
char:	c
short (int16):	s
int (int32):	i
__int64:	l
float (float32):	f
double (float64):	d
char:	c

For example, a pointer to a buffer of unsigned ints will be named `puiBufferName`.

## 1.6 Licensing Issues

The example application uses the open source library **libSndFile**, which is under the license LGPL, so it must not be linked statically and it has to be stated that it is used in this application. For further information about the LGPL license, visit [www.Gnu.org](http://www.Gnu.org).

## 1.7 Support

Support for the source code is - within the limits of the agreement - available from

[zplane.development](mailto:zplane.development)

katzbachstr. 21

D-10965 berlin

germany

fon: +49.30.854 09 15.0

fax: +49.30.854 09 15.5

@: [info@zplane.de](mailto:info@zplane.de)

## 2 [tONaRT] key detection Directory Hierarchy

### 2.1 [tONaRT] key detection Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

**incl**

**7**

## 3 [tONaRT] key detection Data Structure Index

### 3.1 [tONaRT] key detection Data Structures

Here are the data structures with brief descriptions:

**CtONaRT\_If**

**7**

**CtONaRT\_If::Results\_t\_tag** (Structure that holds the results )

**10**

## 4 [tONaRT] key detection File Index

### 4.1 [tONaRT] key detection File List

Here is a list of all files with brief descriptions:

**tONaRT.h** (Interface of the **CtONaRT\_If** class )

11

## 5 [tONaRT] key detection Directory Documentation

### 5.1 incl/ Directory Reference

#### Files

- file **tONaRT.h**  
*interface of the **CtONaRT\_If** class.*

## 6 [tONaRT] key detection Data Structure Documentation

### 6.1 CtONaRT\_If Class Reference

```
#include <tONaRT.h>
```

#### 6.1.1 Detailed Description

##### CLASS

This class provides the interface for the [tONaRT] key detection.

The key detection is stream based and processes a block of data at each time. After successful processing, the results can be obtained with a defined structure.

Definition at line 100 of file tONaRT.h.

#### Public Types

- typedef **CtONaRT\_If::Results\_t\_tag Results\_t**  
*structure that holds the results*

#### Public Member Functions

- **CtONaRT\_If** ()
- virtual **~CtONaRT\_If** ()
- virtual DLLAPI int **Process** (float \*pfInputBufferInterleaved, int iNumberOfFrames)=0
- virtual DLLAPI int **GetTempResults** (**Results\_t** \*pstResults)=0
- virtual DLLAPI int **GetOverallResults** (**Results\_t** \*pstResults)=0

## Static Public Member Functions

- static DLLAPI int **CreateInstance** (CtONaRT\_If \*&pCInstancePointer, int iSampleRate, int iNumberOfChannels, float fStartPitch=0)
- static DLLAPI int **DestroyInstance** (CtONaRT\_If \*&pCInstancePointer)

## Data Structures

- struct **Results\_t\_tag**  
*structure that holds the results*

## 6.1.2 Member Typedef Documentation

### 6.1.2.1 typedef struct **CtONaRT\_If::Results\_t\_tag** **CtONaRT\_If::Results\_t**

structure that holds the results

## 6.1.3 Constructor & Destructor Documentation

### 6.1.3.1 CtONaRT\_If::CtONaRT\_If () [inline]

Definition at line 113 of file tONaRT.h.

### 6.1.3.2 virtual CtONaRT\_If::~CtONaRT\_If () [inline, virtual]

Definition at line 114 of file tONaRT.h.

## 6.1.4 Member Function Documentation

### 6.1.4.1 static DLLAPI int CtONaRT\_If::CreateInstance (**CtONaRT\_If** \*& *pCInstancePointer*, int *iSampleRate*, int *iNumberOfChannels*, float *fStartPitch* = 0) [static]

creates a new instance of [tONaRT] key detection

#### Parameters:

- pCInstancePointer* : handle to the new instance
- iSampleRate* : sample rate of audio signal to be analyzed (in Hz)
- iNumberOfChannels* : number of channels of audio signal to be analyzed
- fStartPitch* : initialization frequency of concert/standard pitch (in Hz, optional)

#### Returns:

static int : 0 when no error

**6.1.4.2 static DLLAPI int CtONaRT\_If::DestroyInstance (CtONaRT\_If \*& p-InstancePointer) [static]**

destroys an instance of [tONaRT] key detection

**Parameters:**

*pCInstancePointer* : handle to the instance to be destroyed

**Returns:**

static int : 0 when no error

**6.1.4.3 virtual DLLAPI int CtONaRT\_If::GetOverallResults (Results\_t \* pst-Results) [pure virtual]**

signals the library that no more input data is expected and writes the overall results

**Parameters:**

*\*pstResults* : pointer to structure for results

**Returns:**

virtual int : 0 when no error

**6.1.4.4 virtual DLLAPI int CtONaRT\_If::GetTempResults (Results\_t \* pst-Results) [pure virtual]**

writes the results until this point of the analysis (may be unreliable!)

**Parameters:**

*\*pstResults* : pointer to structure for results

**Returns:**

virtual int : 0 when no error

**6.1.4.5 virtual DLLAPI int CtONaRT\_If::Process (float \* pfInputBuffer-Interleaved, int iNumberOfFrames) [pure virtual]**

does the key analysis in blocks

**Parameters:**

*\*pfInputBufferInterleaved* : pointer to interleaved audio data in floating point format

*iNumberOfFrames* : number of frames per block (a frame equals one sample for mono signals and two samples for stereo signals), must not be higher than 16384

**Returns:**

virtual int : 0 when no error

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

- [tONaRT.h](#)

## 6.2 CtONaRT\_If::Results\_t\_tag Struct Reference

```
#include <tONaRT.h>
```

### 6.2.1 Detailed Description

structure that holds the results

Definition at line 105 of file tONaRT.h.

#### Data Fields

- char **acKeyName** [1024]  
*string containing the name of the detected key (e.g. "A Maj")*
- int **iKeyIdx**  
*index of detected key, possible values are in the range from 0...24; indices 0..11 mark the keys "A Maj" to "G# Maj", indices 12..23 mark the keys "a min" to "g# min"*
- float **fKeyDetectionProbability**  
*probability of this detection (experimental!)*
- float **fStandardPitchFrequency**  
*frequency of concert pitch in Hz*

### 6.2.2 Field Documentation

#### 6.2.2.1 char CtONaRT\_If::Results\_t\_tag::acKeyName[1024]

string containing the name of the detected key (e.g. "A Maj")

Definition at line 107 of file tONaRT.h.

#### 6.2.2.2 float CtONaRT\_If::Results\_t\_tag::fKeyDetectionProbability

probability of this detection (experimental!)

Definition at line 109 of file tONaRT.h.

### 6.2.2.3 float CtONaRT\_If::Results\_t\_tag::fStandardPitchFrequency

frequency of concert pitch in Hz

Definition at line 110 of file tONaRT.h.

### 6.2.2.4 int CtONaRT\_If::Results\_t\_tag::iKeyIdX

index of detected key, possible values are in the range from 0...24; indices 0..11 mark the keys "A Maj" to "G# Maj", indices 12..23 mark the keys "a min" to "g# min"

Definition at line 108 of file tONaRT.h.

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

- [tONaRT.h](#)

## 7 [tONaRT] key detection File Documentation

### 7.1 docugen.txt File Reference

#### 7.1.1 Detailed Description

source documentation main file

Definition in file [docugen.txt](#).

### 7.2 tONaRT.h File Reference

#### 7.2.1 Detailed Description

interface of the [CtONaRT\\_If](#) class.

:

Definition in file [tONaRT.h](#).

#### Data Structures

- class [CtONaRT\\_If](#)
- struct [CtONaRT\\_If::Results\\_t\\_tag](#)  
*structure that holds the results*

#### Defines

- #define [\\_\\_tONaRT\\_IF\\_HEADER\\_INCLUDED\\_\\_](#)
- #define [DLLAPI](#)

**7.2.2 Define Documentation**

**7.2.2.1 #define \_\_tONaRT\_IF\_HEADER\_INCLUDED\_\_**

Definition at line 78 of file tONaRT.h.

**7.2.2.2 #define DLLAPI**

Definition at line 85 of file tONaRT.h.

## Index

~CtONaRT\_If  
    CtONaRT\_If, 8  
\_\_tONaRT\_IF\_HEADER\_-  
    INCLUDED\_  
    tONaRT.h, 12

acKeyName  
    CtONaRT\_If::Results\_t\_tag, 10

CreateInstance  
    CtONaRT\_If, 8  
CtONaRT\_If, 7  
    CtONaRT\_If, 8  
CtONaRT\_If  
    ~CtONaRT\_If, 8  
    CreateInstance, 8  
    CtONaRT\_If, 8  
    DestroyInstance, 8  
    GetOverallResults, 9  
    GetTempResults, 9  
    Process, 9  
    Results\_t, 8  
CtONaRT\_If::Results\_t\_tag, 10  
CtONaRT\_If::Results\_t\_tag  
    acKeyName, 10  
    fKeyDetectionProbability, 10  
    fStandardPitchFrequency, 10  
    iKeyId, 11

DestroyInstance  
    CtONaRT\_If, 8

DLLAPI  
    tONaRT.h, 12

docugen.txt, 11

fKeyDetectionProbability  
    CtONaRT\_If::Results\_t\_tag, 10  
fStandardPitchFrequency  
    CtONaRT\_If::Results\_t\_tag, 10

GetOverallResults  
    CtONaRT\_If, 9  
GetTempResults  
    CtONaRT\_If, 9

iKeyId  
    CtONaRT\_If::Results\_t\_tag, 11

incl/ Directory Reference, 7

Process  
    CtONaRT\_If, 9

Results\_t  
    CtONaRT\_If, 8

tONaRT.h, 11  
tONaRT.h  
    \_\_tONaRT\_IF\_HEADER\_-  
        INCLUDED\_, 12  
    DLLAPI, 12