Light is OSRAM

Tuner4TRONIC DLL 4

Please note:
All information in this guide has been prepared with great care. OSRAM, however, does not accept liability for possible errors, changes and/or omissions. Please check www.osram.com or contact your sales partner for an updated copy of this guide. This technical application guide is for information purposes only and aims to support you in tackling the challenges and taking full advantage of all opportunities the technology has to offer. Please note that this guide is based on own measurements, tests, specific parameters and assumptions. Individual applications may not be covered and need different handling. Responsibility and testing obligations remain with the luminaire manufacturer/OEM/application planner.

Table of content

1 Introduction

This document describes the API of Tuner4TRONIC DLL (T4T-DLL) version 4.x.

T4T-DLL provides the following features:

- Programming of OSRAM ECGs using luminaire production files (with extention **.osrtup**)
- Readback of OSRAM ECGs and saving data to a raw data file (**.osrtur**)
- Reading of driver information from an OSRAM ECG in order to identify it.
- Reading of luminaire GTIN from an OSRAM ECG
- Other features.

It is intended for customers whose luminaire production software is a **.NET application** so that the user references the T4T-DLL in his application and can then call the entry points of T4T-DLL.

Other types of production software such as LabView VIs, Java applications, python scripts may also work with T4T-DLL. However T4T-DLL was tested only with .NET applications. Please contact OSRAM Tuner4TRONIC support (T4Tsupport@osram.com) to support you integrating T4T-DLL into your luminaire production software.

T4T-CMD is based on T4T-DLL. It´s an example of a .NET application that uses T4T-DLL.

T4T-DLL supports the following programming interfaces:

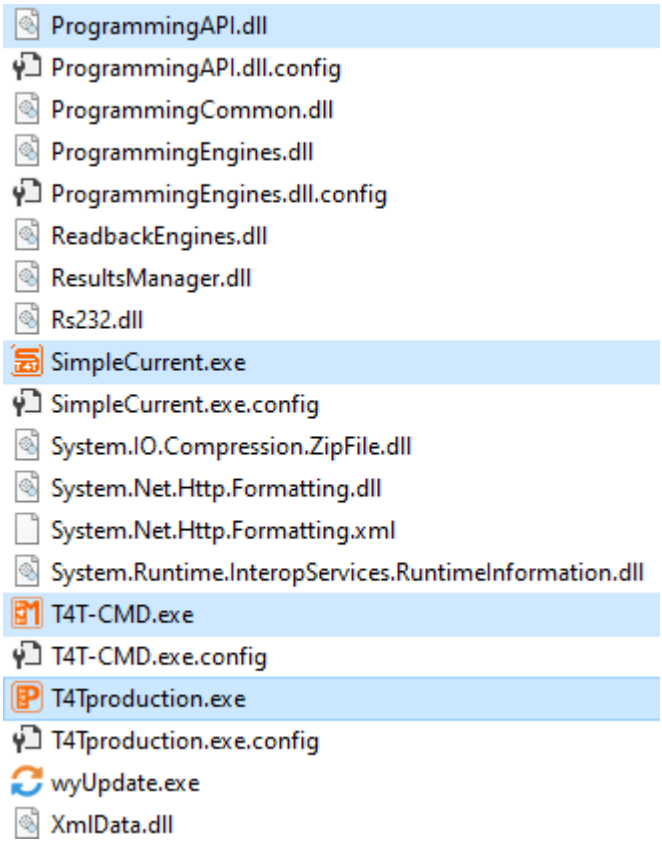
- OSRAM DALI Magic
- FEIG NFC reader ISC.PRH101
- FEIG NFC reader CPR30
- FEIG NFC reader ISC.MR102
- FEIG NFC reader ISC.LR1002
- FEIG ECCO Smart HF.BLE
- OSRAM OT Programmer (COM Box)

2 Installation

As installation requirement the **.Net Framework 4.6** must be installed on the PC running the application using the T4T-DLL.

Since version 4.x T4T-DLL is directly included in the installation of Tuner4TRONIC Production, the binary **ProgrammingAPI.dll** can be found in the installation directory of T4T-P 4.x (**C:\Program Files (x86)\Tuner4TRONIC 4**). Please check https://www.osram.com/ds/tools/tuner4tronic.jsp to download the Tuner4TRONIC Production 4 installer.exe.

The screenshot below shows the executables included in the installation directory.



3 API Specification

The DLL file is **ProgrammingAPI.dll**. Add a reference to that file in the application that shall use T4T-DLL.

The namespace that is exposed is **ProgrammingAPI**. It contains several classes. The most important one is the **ProgrammingAPI**-Class which provides the methods for loading a production file, programming an ECG, reading back an ECG as well as other methods.

Please find below the generated software documentation. An example of a .NET command line application using the T4T-DLL is provided at the end of the document.

ProgrammingAPI

Generated by Doxygen 1.9.2

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 ProgrammingAPI Namespace Reference

Classes

- class ApplicationConstants

  This class provides the return codes of ProgrammingAPI
- class DriverInfo

  Describes the driver information which consists of its GTIN, FW and HW major versions and its serial number provided as strings.
- class Helper

  This class provides the programming interface type
- struct LuminaireConfigurationInfo

  Describes the Luminaire Configuration Information.
- class ProgrammingAPI

  This class provides the methods for loading a production file, programming an ECG, reading back an ECG as well as other methods.
- class ProgrammingStatusEventArgs

  Represents the data provided by Programming Status Event which can be either a programming status, a reading status or an error code.
- class ReadbackFileAvailableEventArgs

  Represents the data provided by Readback file available-event
- class ValueEventArgs

  Represents the progress value of ECG programming or ECG readback in percentage as `int` and string

Functions

- delegate void WriteCompleteCallback (ApplicationConstants.ReturnCode programmingStatus)

  Callback method that shall be triggered once the programming is completed.

3.1.1 Function Documentation

3.1.1.1 WriteCompleteCallback()

```
delegate void ProgrammingAPI.WriteCompleteCallback (
            ApplicationConstants.ReturnCode programmingStatus )
```

Callback method that shall be triggered once the programming is completed.

Parameters

programmingStatus	The ReturnCode is either "Success" or one of the programming related messages

Chapter 4

Class Documentation

4.1 ProgrammingAPI.ApplicationConstants Class Reference

This class provides the return codes of ProgrammingAPI

Public Types

- enum ReturnCode : int {
  Success = 0 , GeneralApplicationError = 1 , TrialPeriodExpired = 2 , ParamFile_NotReadable = 101 ,
  ParamFile_Invalid = 102 , ParamFile_OldVersion = 103 , ParamFile_IsReadOnly = 104 , ParamFile_UpgradeFailed
  = 105 ,
  PI_NoneFound = 200 , PI_FWVersionUnsupported = 201 , PI_WrongTypeFound = 202 , PI_TooManyFound
  = 203 ,
  PI_Overloaded = 204 , PI_CommunicationError = 205 , PI_PowerOnProgrammingInterfaceIsFailed = 206 ,
  NoECGConnected = 300 ,
  MoreThanOneECGConnected = 301 , TooManyECGsConnected = 302 , WrongECGTypeOrVersionConnected
  = 303 , ECGProtected = 304 ,
  InvalidPassword = 305 , InvalidPasswordTimelocked = 306 , InconsistentPasswords = 307 , ECGCommunicationFailure
  = 308 ,
  DaliShotcutOccured = 309 , LuminaireWrongConnection = 310 , VerificationFailed = 311 , ProgrammingAborted
  = 312 ,
  LuminaireConfigurationNotMatching = 313 , NotSupportedForMultiPF = 400 , ParameterNotSupported = 401
  , ParameterClipped = 402 ,
  DeviceDetectionStarted = 1000 , DALIAdressingStarted = 1001 , PreprocessingStarted = 1002 ,
  ProgrammingStarted = 1003 ,
  PostProcessingStarted = 1004 , VerificationStarted = 1005 , ModeSettingStarted = 1006 , ProgrammingFailed
  = 1007 ,
  ProgrammingBatchSizeReached = 1008 , ReadFailed = 1009 , ReadingCanceledByUser = 1011 ,
  ReadingWaitingForOneEcg = 1012 ,
  ReadingInProgress = 1013 , ReadingDone = 1014 , NFCECGPowerIsON = 314 , ConnectedLuminaireCountExceedsBatchSize
  = 315 ,
  ECGIsProgrammedAlready = 316 , MapLuminaireConfigurationFailed = 317 , ProgrammingFailedDueToMisMatchInRFPasswor
  = 318 , NoMatchingDDFilesFoundWithGTIN = 319 ,
  NoMatchingDDFilesFoundWithFWVersion = 320 , NoMatchingDDFilesFoundWithHWVersion = 321 ,
  NoMatchingDDFilesFoundWithFWHWVersion = 322 , ErrorInDeviceDataBase = 323 ,
  IllegalHexStringValue = 324 , PasswordsNotMatchingEcgs = 325 , DifferentEcgsInTheBox = 326 }

  Return Codes of ProgrammingAPI

4.1.1 Detailed Description

This class provides the return codes of ProgrammingAPI

4.1.2 Member Enumeration Documentation

4.1.2.1 ReturnCode

```
enum ProgrammingAPI.ApplicationConstants.ReturnCode :  int
```

Return Codes of ProgrammingAPI

Enumerator

Success	The success
GeneralApplicationError	An unspecified application error has occurred
TrialPeriodExpired	Indicates that the beta trial version of the DLL has expired. No further usage of this version is possible.
ParamFile_NotReadable	The Luminaire Production File doesn't exist at given path or the file cannot be opened.
ParamFile_Invalid	Invalid Luminaire Production File. This can be a wrong extension, wrong (future) version, or no production file at all.
ParamFile_OldVersion	A production file of an old T4T version was provided.
ParamFile_IsReadOnly	Returns if the file is read only.
ParamFile_UpgradeFailed	Returns if the file upgrade is failed.
PI_NoneFound	No Programming Interface was found.
PI_FWVersionUnsupported	The FW version of the Programming Interface is not supported (e.g. DALI Magic with FW 2.18).
PI_WrongTypeFound	The found PI does not match the Luminaire configuration.
PI_TooManyFound	There is more than one PI of the required type connected. Hence the T4T-DLL cannot decide which one to use.
PI_Overloaded	Too many ECG connected to Dali Magic or a DALI shortcut was detected.
PI_CommunicationError	A communication error with the PI occurred. Maybe the PI was disconnected to the USB port.
PI_PowerOnProgrammingInterfaceIsFailed	Power on programming interface is failed.
NoECGConnected	No response was received. Either No ECG is connected or ECGs are not powered.
MoreThanOneECGConnected	Single ECG configuration was selected but there is more than one ECG connected.
TooManyECGsConnected	More ECGs than allowed/required for the luminaire configuration are connected.
WrongECGTypeOrVersionConnected	One or more ECGs don't match the type or FW/HW version included in the Production File. Check the IC/NAED code of the connected ECGs.

Enumerator

ECGProtected	One Or more ECG is write-protected with an OEM Code but no OEM code was provided in the production file.
InvalidPassword	The ECG is locked with a different OEM Code than provided in the production file.
InvalidPasswordTimelocked	The ECG was locked with different password and due to using a wrong password is now time-locked. After first such event external SW needs to wait 5s before next programming attempt can be made.
InconsistentPasswords	The system level or user level password in the device do not match the provided unlock password.
ECGCommunicationFailure	The ECG provided no or an unexpected response.
DaliShotcutOccured	Programming was terminated due to a DALI shortcut.
LuminaireWrongConnection	The 2DIM device wiring connection is wrong
VerificationFailed	Verification of programming failed.
ProgrammingAborted	The programming was aborted by method "abort".
LuminaireConfigurationNotMatching	Single ECG or No ECG connected
NotSupportedForMultiPF	This function call is not supported when the loaded production file contains more than one ECG
ParameterNotSupported	The selected parameter is not supported by the ECG.
ParameterClipped	The value that was used to program the parameter that was out of range and clipped to the allowed range.
DeviceDetectionStarted	The programming procedure was started and the first step (device detection procedure) will be started.
DALIAdressingStarted	The DALI Addressing Procedure was started.
PreprocessingStarted	The DALI Addressing Procedure was started.
ProgrammingStarted	The ECG configuration was checked found ok and programming will be started.
PostProcessingStarted	The DALI Addressing Procedure was started.
VerificationStarted	The programming was completed and verification will be started.
ModeSettingStarted	The programming and verification was completed successfully and Mode setting will be started
ProgrammingFailed	The programming procedure was terminated unsuccessful.
ProgrammingBatchSizeReached	The Max Batch size in workflow is reached
ReadFailed	The read operation failed.
ReadingCanceledByUser	ECG reading process was canceled by user.
ReadingWaitingForOneEcg	ECG reading started, waiting for ECG detection.
ReadingInProgress	ECG reading is in progress.
ReadingDone	ECG reading completed.
NFCECGPowerIsON	The ecg power is on
ConnectedLuminaireCountExceedsBatchSize	The connected luminaire count exceeds luminaire batch size
ECGIsProgrammedAlready	The ecg is programmed already
MapLuminaireConfigurationFailed	The Map configuration is failed when programming ECG.

Enumerator

ProgrammingFailedDueToMisMatchInRFPassword	The Programming failed due to mismatch in RF password.
NoMatchingDDFilesFoundWithGTIN	No matching ECG found with the specified GTIN.
NoMatchingDDFilesFoundWithFWVersion	Found device with valid GTIN but invalid Firmware version.
NoMatchingDDFilesFoundWithHWVersion	Found device with valid GTIN but invalid Hardware version.
NoMatchingDDFilesFoundWithFWHWVersion	Found device with valid GTIN but invalid Firmware version and Hardware version.
ErrorInDeviceDataBase	No matching ECG details of the connected ECG found in devices.xml.
IllegalHexStringValue	The input does not match the Hex string values
PasswordsNotMatchingEcgs	The ECGs in the box have different unlock passwords
DifferentEcgsInTheBox	The ECGs in the box have different type, FW rev or HW rev

4.2 ProgrammingAPI.DriverInfo Class Reference

Describes the driver information which consists of its GTIN, FW and HW major versions and its serial number provided as strings.

Public Member Functions

- DriverInfo (string gtin, string fw, string hw, string sn)

  Constructor
- override string ToString ()

  Returns a string containing all the driver information

Public Attributes

- string Gtin

  GTIN as string
- string FWVersion

  FW major version as string
- string HWVersion

  HW major version as string
- string SerialNumber

  Serial number as string

4.2.1 Detailed Description

Describes the driver information which consists of its GTIN, FW and HW major versions and its serial number provided as strings.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 DriverInfo()

```
ProgrammingAPI.DriverInfo.DriverInfo (
            string gtin,
            string fw,
            string hw,
            string sn )
```

Constructor

Parameters

gtin	GTIN as string
fw	FW major version as string
hw	HW major version as string
sn	Serial number as string

4.2.3 Member Function Documentation

4.2.3.1 ToString()

```
override string ProgrammingAPI.DriverInfo.ToString ( )
```

Returns a string containing all the driver information

Returns

String containing all the driver information

4.2.4 Member Data Documentation

4.2.4.1 FWVersion

```
string ProgrammingAPI.DriverInfo.FWVersion
```

FW major version as string

Driver´s FW major version

4.2.4.2 Gtin

`string ProgrammingAPI.DriverInfo.Gtin`

GTIN as string

Driver´s GTIN

4.2.4.3 HWVersion

`string ProgrammingAPI.DriverInfo.HWVersion`

HW major version as string

Driver´s HW major version

4.2.4.4 SerialNumber

`string ProgrammingAPI.DriverInfo.SerialNumber`

Serial number as string

Driver´s serial number

4.3 ProgrammingAPI.Helper Class Reference

This class provides the programming interface type

Public Types

- enum InterfaceType { None = 0 , DALI = 1 , OSRSER = 2 , NFC = 4 , **ListGenerator** = 100 }
  Programming interface type

4.3.1 Detailed Description

This class provides the programming interface type

4.3.2 Member Enumeration Documentation

4.3.2.1 InterfaceType

`enum ProgrammingAPI.Helper.InterfaceType`

Programming interface type

Enumerator

None	Unspecified
DALI	DALI programming interface
OSRSER	OT Programmer
NFC	NFC reader

4.4 ProgrammingAPI.LuminaireConfigurationInfo Struct Reference

Describes the Luminaire Configuration Information.

Properties

- string GTIN `[getset]`

  GTIN as string
- int ModelID `[getset]`

  Model ID as `int`. Only OT drivers (with OT Programmer interface) have a Model ID
- string BasicCode `[getset]`

  Basic Code as string
- int NAEDCode `[getset]`

  NAED as `int`
- int Multiplicity `[getset]`

  Driver´s multiplicity in the luminaire

4.4.1 Detailed Description

Describes the Luminaire Configuration Information.

4.4.2 Property Documentation

4.4.2.1 BasicCode

```
string ProgrammingAPI.LuminaireConfigurationInfo.BasicCode [get], [set]
```

Basic Code as string

Driver´s Basic Code

4.4.2.2 GTIN

```
string ProgrammingAPI.LuminaireConfigurationInfo.GTIN [get], [set]
```

GTIN as string

Driver´s GTIN

4.4.2.3 ModelID

```
int ProgrammingAPI.LuminaireConfigurationInfo.ModelID  [get], [set]
```

Model ID as `int`. Only OT drivers (with OT Programmer interface) have a Model ID

Driver´s Model ID

4.4.2.4 Multiplicity

```
int ProgrammingAPI.LuminaireConfigurationInfo.Multiplicity  [get], [set]
```

Driver´s multiplicity in the luminaire

Driver´s multiplicity

4.4.2.5 NAEDCode

```
int ProgrammingAPI.LuminaireConfigurationInfo.NAEDCode  [get], [set]
```

NAED as `int`

Driver´s NAED

4.5 ProgrammingAPI.ProgrammingAPI Class Reference

This class provides the methods for loading a production file, programming an ECG, reading back an ECG as well as other methods.

Public Types

- enum UpdateCheckEnum { Timeout , UpdateAvailable , NoUpdates , Error }

  Possible results when checking for updates.

Static Public Member Functions

- static UpdateCheckEnum CheckForUpdates ()

  Checks if updates for the T4T-Suite are available on the update server.
- static List< string > GetInterfaceList ()

  Gets the list of connected programming interfaces.
- static void SetLumGTIN (long value)

  Overwrites the Luminaire GTIN in the currently loaded production file with the value passed
- static void SetLumIdentificationNumber (ulong value)

  Overwrites the Luminaire identification number in the currently loaded production file with the value passed
- static void SetLumContentFormatVersion (int value)

  Overwrites the Luminaire Content Format Version in the currently loaded production file with the value passed
- static void SetLumVendorSpecificContent (string value)

  Overwrites the Luminaire Vendor Specific Content in the currently loaded production file with the value passed
- static void SetCloudEnable (bool isEnable)

  Enables or disables cloud services.
- static void SetUpdateResultsEnable (bool isEnable)

  Enables or disables updating of programming results in the production file.
- static ApplicationConstants.ReturnCode LoadProductionFile (string fileName, Helper.InterfaceType piType=Helper.InterfaceType.None, string piName="")

  Loads the production file then checks if a supported (or the user-specified) programming interface is connected. If cloud services are enabled, programming data of the production file is uploaded to the cloud for the current session.
- static void Program (WriteCompleteCallback callback, int boxProgrammingEcgNumber=0, int antennaPowerSelectedIndex=0)

  Programs the data of the loaded production file. Uses the programming settings like Reworks/First Programming, OEMCode and Verify as stored in the currently loaded production file.
- static void Program (ushort oemcode, bool verifyProgramming, bool enableSingleProgramming, bool disableFamilyProgramming, WriteCompleteCallback callback, int boxProgrammingEcgNumber=0, int antennaPowerSelectedIndex=0)

  Programs the data of the loaded production file. Overwrites the programming settings stored in the currently loaded production file with the parameters that are passed, if permitted by the production file.
- static void Program (string masterKey, bool verifyProgramming, bool enableSingleProgramming, bool disableFamilyProgramming, WriteCompleteCallback callback, int boxProgrammingEcgNumber=0, int antennaPowerSelectedIndex=0)

  Programs the data of the loaded production file. Overwrites the programming settings stored in the currently loaded production file with the parameters that are passed, if permitted by the production file.
- static ApplicationConstants.ReturnCode EcgReadback (Helper.InterfaceType piType=Helper.InterfaceType.None, string piName="")

  Performs a complete readback of a connected ECG. On successful completion of ECG readback a ReadbackFileAvailable event is triggered.
- static ApplicationConstants.ReturnCode ReadDriverInfo (out DriverInfo driverInfo, Helper.InterfaceType piType=Helper.InterfaceType.None, string piName="")

  Reads the driver information of a connected ECG.
- static ApplicationConstants.ReturnCode ReadLuminaireGtin (out string luminaireGtin, Helper.InterfaceType piType=Helper.InterfaceType.None, string piName="")

  Reads the luminaire GTIN of a connected ECG.
- static void **Abort** ()

  Aborts programming and reading back of an ECG. Also the ProgrammingStatusChanged-event is triggered with code ProgrammingAborted
- static bool IsWorking ()

  Indicates whether an ECG programming process or ECG readback process is ongoing.
- static void EnableLabelPrinting (bool isEnable)

  Enables or disables label printing for ECG programming.

- static void SetSpoolingFolderPath (string spoolingFolderName)

  Sets the spooling folder path for label printing.
- static void SetLabelDefinitionFile (string labelDefFile)

  Sets the label definition file for label printing.

Properties

- static string Version `[get]`

  Gets the T4T-DLL version. For example "4.2.1.0".
- static string Luminaire_Name `[get]`

  Gets the luminaire name stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded
- static string Luminaire_OrderCode `[get]`

  Gets the the luminaire order code stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded.
- static string Luminaire_Image `[get]`

  Gets the path to the decoded luminaire image included in the currently loaded production file. Returns an empty string if no or unusable production file is loaded or production file includes no luminaire image.
- static string Luminaire_Description `[get]`

  Gets the luminaire description text stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded.
- static string Luminaire_CustomerProject `[get]`

  Gets the "customer/project" text stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded
- static IEnumerable< LuminaireConfigurationInfo >? Luminaire_Configuration `[get]`

  Gets the collection of luminaire configurations (GTIN, ModelID, BasicCode, NAEDCode and Multiplicity) in the loaded production file.

Events

- static EventHandler< ProgrammingStatusEventArgs > **ProgrammingStatusChanged**

  This event is triggered when there is a change in programming status or in reading status or if an error occurs during programming or reading.
- static EventHandler< ValueEventArgs > **ProgressChanged**

  This event is triggered when there is a change in the programming or reading progress. The returned value represents the completed percentage between 1 and 100%
- static EventHandler< ReadbackFileAvailableEventArgs > **ReadbackFileAvailable**

  This event is triggered when a raw data file (.osrtur) is available after ECG read back was completed.

4.5.1 Detailed Description

This class provides the methods for loading a production file, programming an ECG, reading back an ECG as well as other methods.

4.5.2 Member Enumeration Documentation

4.5.2.1 UpdateCheckEnum

`enum ProgrammingAPI.ProgrammingAPI.UpdateCheckEnum`

Possible results when checking for updates.

Enumerator

Timeout	Timeout reached while attempting to contact the update server.
UpdateAvailable	A newer version is available on the update server.
NoUpdates	We have already the latest version.
Error	Error

4.5.3 Member Function Documentation

4.5.3.1 CheckForUpdates()

```
static UpdateCheckEnum ProgrammingAPI.ProgrammingAPI.CheckForUpdates ( )  [static]
```

Checks if updates for the T4T-Suite are available on the update server.

Returns

A code of type UpdateCheckEnum

4.5.3.2 EcgReadback()

```
static ApplicationConstants.ReturnCode ProgrammingAPI.ProgrammingAPI.EcgReadback (
            Helper.InterfaceType piType = Helper.InterfaceType.None,
            string piName = "" )  [static]
```

Performs a complete readback of a connected ECG. On successful completion of ECG readback a ReadbackFileAvailable event is triggered.

Parameters

piType	Type of the programming interface
piName	Name of the programming interface as string. For NFC it´s the NFC reader ID as Hex string. For DALI it´s the name of the DALI magic. for OT Programmer it´s the COM port number

Returns

Code indicating whether the readback operation was successfully started.

No need to load a production file prior to calling this method.

This method starts internally its own thread so the user does not need to call it on a different thread. The method returns when the ECG readback is started (and ECG reading continues in the internal thread).

4.5.3.3 EnableLabelPrinting()

```
static void ProgrammingAPI.ProgrammingAPI.EnableLabelPrinting (
            bool isEnable ) [static]
```

Enables or disables label printing for ECG programming.

Parameters

isEnable	If `true` label printing is enabled, if `false` label printing is disabled.

Default is label printing disabled. If needed, this method shall be called before calling Program()

4.5.3.4 GetInterfaceList()

```
static List< string > ProgrammingAPI.ProgrammingAPI.GetInterfaceList ( ) [static]
```

Gets the list of connected programming interfaces.

Returns

List of the connected programming interfaces as strings

This method will detect only DALI and NFC programming interfaces. It won´t detect OT programmers.

4.5.3.5 IsWorking()

```
static bool ProgrammingAPI.ProgrammingAPI.IsWorking ( ) [static]
```

Indicates whether an ECG programming process or ECG readback process is ongoing.

Returns

boolean if `true` then ECG programming or ECG readback process is ongoing, if `false` no operation is ongoing.

4.5.3.6 LoadProductionFile()

```
static ApplicationConstants.ReturnCode ProgrammingAPI.ProgrammingAPI.LoadProductionFile (
            string fileName,
            Helper.InterfaceType piType = Helper.InterfaceType.None,
            string piName = "" ) [static]
```

Loads the production file then checks if a supported (or the user-specified) programming interface is connected. If cloud services are enabled, programming data of the production file is uploaded to the cloud for the current session.

Parameters

fileName	Name of the production file (with extention .osrtup)
piType	Type of the programming interface
piName	Name of the programming interface as string. For NFC it´s the NFC reader ID as Hex string. For DALI it´s the name of the DALI magic. for OT Programmer it´s the COM port number

Returns

Code indicating whether the production file was loaded successfully and a valid programming interface is connected.

This method must be called before calling Program()

4.5.3.7 Program() [1/3]

```
static void ProgrammingAPI.ProgrammingAPI.Program (
            string masterKey,
            bool verifyProgramming,
            bool enableSingleProgramming,
            bool disableFamilyProgramming,
            WriteCompleteCallback callback,
            int boxProgrammingEcgNumber = 0,
            int antennaPowerSelectedIndex = 0 )  [static]
```

Programs the data of the loaded production file. Overwrites the programming settings stored in the currently loaded production file with the parameters that are passed, if permitted by the production file.

Parameters

masterKey	The master key. It shall be in the format of 4 bytes hex value.
verifyProgramming	Indicates if programming shall be followed by verification (true) or not (false).
enableSingleProgramming	If set to `true` enables single programming
disableFamilyProgramming	Option to disable family programming.
callback	Callback method that shall be triggered once the programming is completed
boxProgrammingEcgNumber	Expected number of ECGs in the box (only for NFC box programming, optional)
antennaPowerSelectedIndex	NFC antenna power (only for LR1002 reader, optional)

User must load a production file via LoadProductionFile() before calling this method

This method starts internally its own thread so the user does not need to call it on a different thread. The method returns before the programming is completed. It returns when the connected ECG is detected (and ECG programming continues in the internal thread) or if no ECG is connected when the ECG detection timeout is reached.

The ECG detection timeout is set internally to 5 s for DALI- and OT Programmer and to 20 s for NFC programming interfaces. If ECG detection timeout is reached, a ProgrammingStatusChanged-event is triggered with code NoECGConnected

4.5.3.8 Program() [2/3]

```
static void ProgrammingAPI.ProgrammingAPI.Program (
            ushort oemcode,
```

```
            bool verifyProgramming,
            bool enableSingleProgramming,
            bool disableFamilyProgramming,
            WriteCompleteCallback callback,
            int boxProgrammingEcgNumber = 0,
            int antennaPowerSelectedIndex = 0 )  [static]
```

Programs the data of the loaded production file. Overwrites the programming settings stored in the currently loaded production file with the parameters that are passed, if permitted by the production file.

Parameters

oemcode	Four digits unlock code for programming of the protected features. This is only applied if Rework is true.
verifyProgramming	Indicates if programming shall be followed by verification (true) or not (false).
enableSingleProgramming	If set to `true` enables single programming
disableFamilyProgramming	Option to disable family programming.
callback	Callback method that shall be triggered once the programming is completed
boxProgrammingEcgNumber	Expected number of ECGs in the box (only for NFC box programming, optional)
antennaPowerSelectedIndex	NFC antenna power (only for LR1002 reader, optional)

User must load a production file via LoadProductionFile() before calling this method

This method starts internally its own thread so the user does not need to call it on a different thread. The method returns before the programming is completed. It returns when the connected ECG is detected (and ECG programming continues in the internal thread) or if no ECG is connected when the ECG detection timeout is reached.

The ECG detection timeout is set internally to 5 s for DALI- and OT Programmer and to 20 s for NFC programming interfaces. If ECG detection timeout is reached, a ProgrammingStatusChanged-event is triggered with code NoECGConnected

4.5.3.9 Program() [3/3]

```
static void ProgrammingAPI.ProgrammingAPI.Program (
            WriteCompleteCallback callback,
            int boxProgrammingEcgNumber = 0,
            int antennaPowerSelectedIndex = 0 )  [static]
```

Programs the data of the loaded production file. Uses the programming settings like Reworks/First Programming, OEMCode and Verify as stored in the currently loaded production file.

Parameters

callback	Callback method that shall be triggered once the programming is completed
boxProgrammingEcgNumber	Expected number of ECGs in the box (only for NFC box programming, optional)
antennaPowerSelectedIndex	NFC antenna power (only for LR1002 reader, optional)

User must load a production file via LoadProductionFile() before calling this method

This method starts internally its own thread so the user does not need to call it on a different thread. The method returns before the programming is completed. It returns when the connected ECG is detected (and ECG programming continues in the internal thread) or if no ECG is connected when the ECG detection timeout is reached.

The ECG detection timeout is set internally to 5 s for DALI- and OT Programmer and to 20 s for NFC programming interfaces. If ECG detection timeout is reached, a ProgrammingStatusChanged-event is triggered with code NoECGConnected

4.5.3.10 ReadDriverInfo()

```
static ApplicationConstants.ReturnCode ProgrammingAPI.ProgrammingAPI.ReadDriverInfo (
            out DriverInfo driverInfo,
            Helper.InterfaceType piType = Helper.InterfaceType.None,
            string piName = "" )  [static]
```

Reads the driver information of a connected ECG.

Parameters

driverInfo	Where the driver information of the connected ECG shall be saved.
piType	Type of the programming interface
piName	Name of the programming interface as string. For NFC it´s the NFC reader ID as Hex string. For DALI it´s the name of the DALI magic. for OT Programmer it´s the COM port number

Returns

Code indicating whether the driver information was read successfully.

This method supports only DALI and NFC ECGs. OT drivers are not supported. No need to load a production file prior to calling this method.

4.5.3.11 ReadLuminaireGtin()

```
static ApplicationConstants.ReturnCode ProgrammingAPI.ProgrammingAPI.ReadLuminaireGtin (
            out string luminaireGtin,
            Helper.InterfaceType piType = Helper.InterfaceType.None,
            string piName = "" )  [static]
```

Reads the luminaire GTIN of a connected ECG.

Parameters

luminaireGtin	String where the luminaire GTIN of the connected ECG shall be saved.
piType	Type of the programming interface
piName	Name of the programming interface as string. For NFC it´s the NFC reader ID as Hex string. For DALI it´s the name of the DALI magic. for OT Programmer it´s the COM port number

Returns

Code indicating whether the luminaire GTIN was read successfully.

This method supports only DALI and NFC ECGs. OT drivers are not supported. No need to load a production file prior to calling this method.

4.5.3.12 SetCloudEnable()

```
static void ProgrammingAPI.ProgrammingAPI.SetCloudEnable (
            bool isEnable ) [static]
```

Enables or disables cloud services.

Parameters

isEnable	If `true` cloud services are enabled, if `false` cloud services are disabled.

Default is cloud services enabled. If needed, this method shall be called before calling LoadProductionFile()

4.5.3.13 SetLabelDefinitionFile()

```
static void ProgrammingAPI.ProgrammingAPI.SetLabelDefinitionFile (
            string labelDefFile ) [static]
```

Sets the label definition file for label printing.

Parameters

labelDefFile	Path to label definition file

If needed, this method shall be called before calling Program()

4.5.3.14 SetLumContentFormatVersion()

```
static void ProgrammingAPI.ProgrammingAPI.SetLumContentFormatVersion (
            int value ) [static]
```

Overwrites the Luminaire Content Format Version in the currently loaded production file with the value passed

Parameters

value	New value of Luminaire Content Format Version as `int`

If needed, this method shall be called after calling LoadProductionFile() and prior to calling Program()

4.5.3.15 SetLumGTIN()

```
static void ProgrammingAPI.ProgrammingAPI.SetLumGTIN (
            long value ) [static]
```

Overwrites the Luminaire GTIN in the currently loaded production file with the value passed

Parameters

value	New value of the Luminaire GTIN as `long`

If needed, this method shall be called after calling LoadProductionFile() and prior to calling Program()

4.5.3.16 SetLumIdentificationNumber()

```
static void ProgrammingAPI.ProgrammingAPI.SetLumIdentificationNumber (
            ulong value )  [static]
```

Overwrites the Luminaire identification number in the currently loaded production file with the value passed

Parameters

value	New value of Luminaire identification number as `ulong`

If needed, this method shall be called after calling LoadProductionFile() and prior to calling Program()

4.5.3.17 SetLumVendorSpecificContent()

```
static void ProgrammingAPI.ProgrammingAPI.SetLumVendorSpecificContent (
            string value )  [static]
```

Overwrites the Luminaire Vendor Specific Content in the currently loaded production file with the value passed

Parameters

value	New value of Luminaire vendor specific content as `string`

If needed, this method shall be called after calling LoadProductionFile() and prior to calling Program()

4.5.3.18 SetSpoolingFolderPath()

```
static void ProgrammingAPI.ProgrammingAPI.SetSpoolingFolderPath (
            string spoolingFolderName )  [static]
```

Sets the spooling folder path for label printing.

Parameters

spoolingFolderName	Spooling folder path

If needed, this method shall be called before calling Program()

4.5.3.19 SetUpdateResultsEnable()

```
static void ProgrammingAPI.ProgrammingAPI.SetUpdateResultsEnable (
            bool isEnable ) [static]
```

Enables or disables updating of programming results in the production file.

Parameters

isEnable	If `true` updating of programming results enabled, if `false` it is disabled.

Default is updating of programming results to production file enabled. If needed, this method shall be called before calling LoadProductionFile()

4.5.4 Property Documentation

4.5.4.1 Luminaire_Configuration

```
IEnumerable<LuminaireConfigurationInfo>? ProgrammingAPI.ProgrammingAPI.Luminaire_Configuration
[static], [get]
```

Gets the collection of luminaire configurations (GTIN, ModelID, BasicCode, NAEDCode and Multiplicity) in the loaded production file.

The collection of luminaire configurations.

4.5.4.2 Luminaire_CustomerProject

```
string ProgrammingAPI.ProgrammingAPI.Luminaire_CustomerProject [static], [get]
```

Gets the "customer/project" text stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded

The luminaire customer project.

4.5.4.3 Luminaire_Description

```
string ProgrammingAPI.ProgrammingAPI.Luminaire_Description [static], [get]
```

Gets the luminaire description text stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded.

The luminaire description.

4.5.4.4 Luminaire_Image

```
string ProgrammingAPI.ProgrammingAPI.Luminaire_Image  [static], [get]
```

Gets the path to the decoded luminaire image included in the currently loaded production file. Returns an empty string if no or unusable production file is loaded or production file includes no luminaire image.

The luminaire image.

4.5.4.5 Luminaire_Name

```
string ProgrammingAPI.ProgrammingAPI.Luminaire_Name  [static], [get]
```

Gets the luminaire name stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded

The name of the luminaire.

4.5.4.6 Luminaire_OrderCode

```
string ProgrammingAPI.ProgrammingAPI.Luminaire_OrderCode  [static], [get]
```

Gets the the luminaire order code stored in the currently loaded production file. Returns an empty string if no or unusable production file is loaded.

The luminaire order code.

4.5.4.7 Version

```
string ProgrammingAPI.ProgrammingAPI.Version  [static], [get]
```

Gets the T4T-DLL version. For example "4.2.1.0".

The T4T-DLL version.

4.6 ProgrammingAPI.ProgrammingStatusEventArgs Class Reference

Represents the data provided by Programming Status Event which can be either a programming status, a reading status or an error code.

Inherits EventArgs.

Public Member Functions

- ProgrammingStatusEventArgs (ApplicationConstants.ReturnCode returnCode)

  Constructor

Properties

- ApplicationConstants.ReturnCode ReturnCode [get]

  Gets the return code.

4.6.1 Detailed Description

Represents the data provided by Programming Status Event which can be either a programming status, a reading status or an error code.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 ProgrammingStatusEventArgs()

```
ProgrammingAPI.ProgrammingStatusEventArgs.ProgrammingStatusEventArgs (
            ApplicationConstants.ReturnCode returnCode )
```

Constructor

Parameters

returnCode	The return code

4.6.3 Property Documentation

4.6.3.1 ReturnCode

```
ApplicationConstants.ReturnCode ProgrammingAPI.ProgrammingStatusEventArgs.ReturnCode [get]
```

Gets the return code.

The return code which is either a programming status, a reading status or an error code.

4.7 ProgrammingAPI.ReadbackFileAvailableEventArgs Class Reference

Represents the data provided by Readback file available-event

Inherits EventArgs.

Public Member Functions

- ReadbackFileAvailableEventArgs (byte[ ] osrturFileByteArray, string gtin, string fw, string hw)

  Constructor

Properties

- byte[ ] FileAsByteArray `[getset]`

  The generated raw data file (.osrtur) as byte array
- string Gtin `[getset]`

  GTIN of connected ECG as string
- string Fw `[getset]`

  FW major version of connected ECG as string
- string Hw `[getset]`

  HW major version of connected ECG as string

4.7.1 Detailed Description

Represents the data provided by Readback file available-event

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ReadbackFileAvailableEventArgs()

```
ProgrammingAPI.ReadbackFileAvailableEventArgs.ReadbackFileAvailableEventArgs (
            byte[] osrturFileByteArray,
            string gtin,
            string fw,
            string hw )
```

Constructor

Parameters

osrturFileByteArray	Generated raw data file (.osrtur) as byte array
gtin	GTIN of connected ECG as string
fw	FW major version of connected ECG as string
hw	HW major version of connected ECG as string

4.7.3 Property Documentation

4.7.3.1 FileAsByteArray

```
byte [] ProgrammingAPI.ReadbackFileAvailableEventArgs.FileAsByteArray  [get], [set]
```

The generated raw data file (.osrtur) as byte array

The generated raw data file (.osrtur) as byte array

4.7.3.2 Fw

```
string ProgrammingAPI.ReadbackFileAvailableEventArgs.Fw  [get], [set]
```

FW major version of connected ECG as string

FW major version of connected ECG as string

4.7.3.3 Gtin

```
string ProgrammingAPI.ReadbackFileAvailableEventArgs.Gtin  [get], [set]
```

GTIN of connected ECG as string

GTIN of connected ECG as string

4.7.3.4 Hw

```
string ProgrammingAPI.ReadbackFileAvailableEventArgs.Hw  [get], [set]
```

HW major version of connected ECG as string

HW major version of connected ECG as string

4.8 ProgrammingAPI.ValueEventArgs Class Reference

Represents the progress value of ECG programming or ECG readback in percentage as `int` and string

Inherits EventArgs.

Public Member Functions

- ValueEventArgs (int value)
  Constructor
- override string ToString ()
  Returns the progress value as string

Properties

- int Value [getset]

  Progress value as `int`

4.8.1 Detailed Description

Represents the progress value of ECG programming or ECG readback in percentage as `int` and string

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ValueEventArgs()

```
ProgrammingAPI.ValueEventArgs.ValueEventArgs (
            int value )
```

Constructor

Parameters

value	Progress value as `int`

4.8.3 Member Function Documentation

4.8.3.1 ToString()

```
override string ProgrammingAPI.ValueEventArgs.ToString ( )
```

Returns the progress value as string

Returns

Progress value as string

4.8.4 Property Documentation

4.8.4.1 Value

```
int ProgrammingAPI.ValueEventArgs.Value [get], [set]
```

Progress value as `int`

Progress value as `int`

Chapter 5

Application Example

Below is an example for a .NET command line application using the T4T-DLL.

```
using System;
using System.IO;
using System.Linq;
using System.Threading;
using ProgrammingAPI;

namespace ExampleAppProgrammingAPI
{
    class Program
    {
        static AutoResetEvent programmingCompleteEvent = new AutoResetEvent(false);

        static void Main(string[] args)
        {
            if (args.Length > 0)
            {
                string command = args[0];
                string fileName = string.Empty;

                if (command.ToUpper() == Commands.StartProgramming)
                {
                    if (args != null && args.Count() > 1)
                    {
                        fileName = args[1];
                    }
                    else
                    {
                        Console.WriteLine("Production File Missing!!!");
                        return;
                    }

                    WriteToDevice(fileName);
                }
                else if (command.ToUpper() == Commands.StartReading)
                {
                    ReadFromDevice();
                }
                else
                {
                    Console.WriteLine("Invalid Command!!!");
                    return;
                }

                // wait for programming or readback completion
                while (ProgrammingAPI.ProgrammingAPI.IsWorking())
                {
                    Thread.Sleep(50);
```

```
            bool keyPressed = false;

            try
            {
                keyPressed = Console.KeyAvailable;
            }
            catch { }

            if (keyPressed)
            {
                ProgrammingAPI.ProgrammingAPI.Abort();

                Thread.Sleep(100);
            }
        }
    }
    else
    {
        Console.WriteLine("No Command Entered!!!");
    }
}

private static void WriteToDevice(string filename)
{
    ProgrammingAPI.ProgrammingAPI.LoadProductionFile(filename);
    System.Threading.Thread.Sleep(100);

    // Unsubscribe first
    ProgrammingAPI.ProgrammingAPI.ProgressChanged -= programmingManager_ProgressChanged;
    ProgrammingAPI.ProgrammingAPI.ProgrammingStatusChanged -=
    programmingManager_ProgrammingStatusChanged;

    // Subscribe to notifications
    ProgrammingAPI.ProgrammingAPI.ProgressChanged += programmingManager_ProgressChanged;
    ProgrammingAPI.ProgrammingAPI.ProgrammingStatusChanged +=
    programmingManager_ProgrammingStatusChanged;

    programmingCompleteEvent.Reset();

    ProgrammingAPI.ProgrammingAPI.Program(WriteCompleted);
}

private static void ReadFromDevice()
{
    // Unsubscribe first
    ProgrammingAPI.ProgrammingAPI.ProgressChanged -= programmingManager_ProgressChanged;
    ProgrammingAPI.ProgrammingAPI.ProgrammingStatusChanged -=
    programmingManager_ProgrammingStatusChanged;
    ProgrammingAPI.ProgrammingAPI.ReadbackFileAvailable -= OnReadbackFileAvailable;

    // Subscribe to notifications
    ProgrammingAPI.ProgrammingAPI.ProgressChanged += programmingManager_ProgressChanged;
    ProgrammingAPI.ProgrammingAPI.ProgrammingStatusChanged +=
    programmingManager_ProgrammingStatusChanged;
    ProgrammingAPI.ProgrammingAPI.ReadbackFileAvailable += OnReadbackFileAvailable;

    ProgrammingAPI.ApplicationConstants.ReturnCode ret = ProgrammingAPI.ProgrammingAPI.EcgReadback();
}

/// Call back method to be triggered on programming completion
private static void WriteCompleted(ProgrammingAPI.ApplicationConstants.ReturnCode programmingStatus)
{
    if (programmingStatus == ProgrammingAPI.ApplicationConstants.ReturnCode.Success)
    {
        int Code = (int)ProgrammingAPI.ApplicationConstants.ReturnCode.Success;
        Console.WriteLine("### {0} --- code: {1}", programmingStatus, Code);
    }
    else
    {
        int Code = (int)ProgrammingAPI.ApplicationConstants.ReturnCode.Success;
        Console.WriteLine("### ERROR --- message: {0} --- code: {1}", programmingStatus, Code);
    }
    programmingCompleteEvent.Set();
```

```
        }

        static void programmingManager_ProgressChanged(object sender, ValueEventArgs e)
        {
            Console.WriteLine("### Progress:" + e.ToString());
            Console.ResetColor();
        }

        static void programmingManager_ProgrammingStatusChanged(object sender, ProgrammingStatusEventArgs e)
        {
            Console.WriteLine("### Programming Status: " + e.ReturnCode.ToString());
            Console.ResetColor();
        }

        static void OnReadbackFileAvailable(object sender, ReadbackFileAvailableEventArgs e)
        {
            if (e.FileAsByteArray != null)
            {
                File.WriteAllBytes("DeviceReadback.osrtur", e.FileAsByteArray);
            }
        }
    }

    public class Commands
    {
        public const string StartProgramming = "T4T";
        public const string StartReading = "READBACK";
        public const string Version = "-VERSION";
        public const string VerboseFullTag = "-VERBOSE";
        public const string VerboseShortTag = "-V";
    }
}
```

Index

OSRAM GmbH

Headquarters Germany:

Marcel-Breuer-Strasse 6
80807 Munich, Germany
Phone +49 89 6213-0
Fax +49 89 6213-2020
www.osram.com

Tuner4TRONIC support: T4Tsupport@osram.com