Semiconductors

TDA18273 Driver User Guide

Rev. 0.3 — 24th September 2010

Document information

Info	Content
Keywords	TDA18273, Silicon Tuner, TV reception, Hybrid, Driver
Abstract	User guide of TDA18273 driver. Description of the public functions and how to use the driver.

founded by Philips

Revision history

Rev	Date	Author	Description
0.0	09th February 2010	C. CAZETTES	NXPFE/V4 Driver User Guide Template Creation.
0.1	27th June 2010	C. CAZETTES	NXPFE/V4 Driver User Guide Template Update.
0.2	27th August 2010	M.VANNIER	Update with latest TDA18273 driver changes + flow Charts programming
0.3	24th September 2010	D.LEGENDRE	Update flowcharts (HwInit and SetRf)

1. Introduction

This document contains a description of the public functions and types of TDA18273 driver.

2. Package contents

The driver package contains the following components:

-A library containing the driver

-A header file containing the public functions of the driver

-A folder containing the include files required for the driver

-A folder for the TDA18273 initial configuration.

3. Description of public functions of the driver

3.1 Overview

The driver can be fully controlled by the following functions:

```
tmbslTDA18273_Open
tmbslTDA18273_Close
tmbslTDA18273_HwInit
tmbslTDA18273_SetStandardMode
tmbslTDA18273_GetStandardMode
tmbslTDA18273_SetRF
tmbslTDA18273_GetRF
tmbslTDA18273_GetLockStatus
tmbslTDA18273_GetPowerLevel
tmbslTDA18273_SetPowerState
tmbslTDA18273_GetPowerState
tmbslTDA18273_SetLLPowerState
tmbslTDA18273_GetLLPowerState
tmbslTDA18273_SetFineRF
tmbslTDA18273_GetIF
tmbslTDA18273_GetCF_Offset
tmbslTDA18273_WaitIRQ
tmbslTDA18273_GetXtalCal_End
tmbslTDA18273_SetIRQWait
tmbslTDA18273_GetIRQWait
tmbslTDA18273_SetIRQWaitHwInit
tmbslTDA18273_GetIRQWaitHwInit
tmbslTDA18273_GetXtalCal_End
tmbslTDA18273_GetIRQ
tmbslTDA18273_Write
tmbslTDA18273_Read
tmbslTDA18273_GetSWVersion
tmbslTDA18273_GetSWSettingsVersion
tmbslTDA18273_CheckHWVersion
```

3.2 tmbslTDA18273_Open

3.2.1 Description

Opens the driver instance. No hardware access is performed in this function.

3.2.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **tmbslFrontEndDependency_t *psSrvFunc***: Structure containing the Hardware Access functions, the Time functions and the Debug functions.

3.2.3 Example

```
tmErrorCode_t               err = TM_OK;
tmbslFrontEndDependency_t   sSrvBslFunc;

sSrvBslFunc.sIo.Write             = UserWrittenI2C_Write;
sSrvBslFunc.sIo.Read              = UserWrittenI2C_Read;
sSrvBslFunc.sTime.Get             = Null;
sSrvBslFunc.sTime.Wait            = UserWritten_Wait;
sSrvBslFunc.sDebug.Print          = UserWritten_Print;
sSrvBslFunc.sMutex.Open           = Null;
sSrvBslFunc.sMutex.Close          = Null;
sSrvBslFunc.sMutex.Acquire        = Null;
sSrvBslFunc.sMutex.Release        = Null;
sSrvBslFunc.dwAdditionalDataSize  = 0;
sSrvBslFunc.pAdditionalData       = Null;

err = tmbslTDA18273_Open(0, &sSrvBslFunc);
```

3.2.4 Legacy compatibility

```
err = tmbslTDA18273Init(0, &sSrvBslFunc);
```

3.3 tmbslTDA18273_Close

3.3.1 Description

Closes the driver instance. Must be called before calling *tmbslTDA18273_Open* again if already initialized.

3.3.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

3.3.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_Close(0);
```

3.3.4 Legacy compatibility

```
err = tmbslTDA18273DeInit(0);
```

3.4 tmbslTDA18273_HwInit

3.4.1 Description

Initializes TDA18273 Hardware.

Must be called each time TDA18273 power supply is switched to ON.

3.4.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

3.4.3 Example

```
tmErrorCode_t   err = TM_OK;

  err = tmbslTDA18273_HwInit(0);
```

3.4.4 Legacy compatibility

```
  err = tmbslTDA18273Reset(0);
```

3.5 tmbslTDA18273_SetStandardMode

3.5.1 Description

Sets the TDA18273 standard mode preset

3.5.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **TDA18273StandardMode_t *standardMode***: Standard mode to be set. Possible values are:

```
tmTDA18273_DVBT_6MHz
tmTDA18273_DVBT_7MHz
tmTDA18273_DVBT_8MHz
tmTDA18273_QAM_6MHz
tmTDA18273_QAM_8MHz
tmTDA18273_ISDBT_6MHz
tmTDA18273_DMBT_8MHz
tmTDA18273_ANLG_MN
tmTDA18273_ANLG_B
tmTDA18273_ANLG_GH
tmTDA18273_ANLG_I
tmTDA18273_ANLG_DK
tmTDA18273_ANLG_L
tmTDA18273_ANLG_LL
tmTDA18273_FM_Radio
tmTDA18273_Scanning
tmTDA18273_ScanXpress
```

3.5.3 Example

```
tmErrorCode_t   err = TM_OK;

  err = tmbslTDA18273_SetStandardMode(0, TDA18273_DVBT_6MHz);
```

3.5.4 Legacy compatibility

```
  err = tmbslTDA18273SetStandardMode(0, tmTDA18273_DVBT_6MHz);
```

3.6 tmbslTDA18273_GetStandardMode

3.6.1 Description

Gets the TDA18273 standard mode preset

3.6.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **TDA18273StandardMode_t *pS*tandardMode***: Pointer to the object in which the standard mode preset must be stored. Possible return values are the same as *tmbslTDA18273_SetStandardMode*.

3.6.3 Example

```
tmErrorCode_t            err = TM_OK;
TDA18273StandardMode_t  standardMode = TDA18273_StandardMode_Unknown

  err = tmbslTDA18273_GetStandardMode(0, &standardMode);
```

3.6.4 Legacy compatibility

```
tmTDA18273StandardMode_t standardMode = tmTDA18273_StandardMode_Unknown

  err = tmbslTDA18273GetStandardMode(0, &standardMode);
```

3.7 tmbslTDA18273_SetRF

3.7.1 Description

Sets the tuner to a specified RF frequency.

3.7.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32 *uRF***: Frequency to set in Hertz.

3.7.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_SetRF(0, 770166000);
```

3.7.4 Legacy compatibility

```
err = tmbslTDA18273SetRF(0, 770166000);
```

3.8 tmbslTDA18273_GetRF

3.8.1 Description

Gets the last set RF frequency.

3.8.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32* *puRF***: Pointer to the object in which the RF frequency in Hz must be stored.

3.8.3 Example

```
tmErrorCode_t   err = TM_OK;
Uint32          uRF = 0;

err = tmbslTDA18273_GetRF(0, &uRF);
```

3.9 tmbslTDA18273_GetLockStatus

3.9.1 Description

Gets the PLL lock status of the TDA18273.

3.9.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **tmbslFrontEndState_t **pLockStatus***: Pointer to the object in which the lock status must be stored. Available values are:

```
tmbslFrontEndStateUnknown
tmbslFrontEndStateLocked
tmbslFrontEndStateNotLocked
```

3.9.3 Example

```
tmErrorCode_t   err = TM_OK;
tmbslFrontEndState_t eLockStatus = tmbslFrontEndStateUnknown;

  err = tmbslTDA18273_GetLockStatus(0, &eLockStatus);
```

3.9.4 Legacy compatibility

```
  err = tmbslTDA18273GetLockStatus(0, &eLockStatus);
```

3.10 tmbslTDA18273_GetPowerLevel

3.10.1 Description

Gets the RF input Power Level of the TDA18273 in dBµV.

3.10.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt8 *pPowerLevel***: Pointer to the object in which the power level must be stored. Value returned is expressed in ½ step of dBµV.

3.10.3 Example

```
tmErrorCode_t   err = TM_OK;
UInt8          uValue = 0;
Decimal PowerLevel;

err = tmbslTDA18273_GetPowerLevel(0, &uValue);
PowerLevel = uValue * 0.5;
```

3.10.4 Legacy compatibility

```
err = tmbslTDA18273GetPowerLevel(0, &uPowerLevel);
```

3.11 tmbslTDA18273_SetPowerState

3.11.1 Description

Sets the power state of the TDA18273.

3.11.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **tmPowerState_t *powerState***: Power state that needs to be set. Possible enumeration values are:

```
tmPowerOn
tmPowerStandby
tmPowerSuspend
tmPowerOff
```

3.11.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_SetPowerState(0, tmPowerOn);
```

3.12 tmbslTDA18273_GetPowerState

3.12.1 Description

Gets the power state of the TDA18273.

3.12.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **tmPowerState_t* *pPowerState***: Pointer to the object in which the read power state must be stored. Possible return values are the same as *tmbslTDA18273_SetPowerState*.

3.12.3 Example

```
tmErrorCode_t   err = TM_OK;
tmPowerState_t  powerState;

err = tmbslTDA18273_GetPowerState(0, &powerState);
```

3.13 tmbslTDA18273_SetLLPowerState

3.13.1 Description

Sets the low-level power state of the TDA18273.

3.13.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **TDA18273PowerState_t *powerState***: Power state that needs to be set. Possible enumeration values are:

```
  TDA18273_PowerNormalMode
  TDA18273_PowerStandbyWithXtalOn
  TDA18273_PowerStandby

  Current mapping with tmPowerState_t parameter of
tmbslTDA18273_SetPowerState():
  tmPowerOn = TDA18273_PowerNormalMode
  tmPowerStandby = TDA18273_PowerStandbyWithXtalOn
  tmPowerSuspend = TDA18273_PowerStandbyWithXtalOn
  tmPowerOff = TDA18273_PowerStandbyWithXtalOn
```

3.13.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_SetLLPowerState(0, TDA18273_PowerNormalMode);
```

3.13.4 Legacy compatibility

```
err = tmbslTDA18273SetPowerState(0, tmTDA18273_PowerNormalMode);
```

3.14 tmbslTDA18273_GetLLPowerState

3.14.1 Description

Gets the low-level power state of the TDA18273.

3.14.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **TDA18273PowerState_t* *pPowerState***: Pointer to the object in which the read power state must be stored. Possible return values are the same as *tmbslTDA18273_SetLLPowerState*.

3.14.3 Example

```
tmErrorCode_t           err = TM_OK;
TDA18273PowerState_t  llPowerState;

err = tmbslTDA18273_GetLLPowerState(0, &llPowerState);
```

3.14.4 Legacy compatibility

```
tmTDA18273PowerState_t  powerState;
err = tmbslTDA18273GetPowerState(0, &powerState);
```

3.14.5 Legacy compatibility

```
err = tmbslTDA18273GetRF(0, &uRF);
```

3.15 tmbslTDA18273_SetFineRF

3.15.1 Description

Add or substract one step of 125 KHz to the current tuned RF.

3.15.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **Int8 *step*** : step +1 or -1 of 125 KHz.

3.15.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_SetFineRF(0, 1);
```

3.15.4 Legacy compatibility

```
err = tmbslTDA18273RFineTuning(0, 1);
```

3.16 tmbslTDA18273_GetIF

3.16.1 Description

Gets the IF Frequency set in TDA18273 driver.

3.16.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32* *puIF***: Pointer to the object in which the IF Frequency in Hz must be stored.

3.16.3 Example

```
tmErrorCode_t   err = TM_OK;
Uint32          uIF = 0;

err = tmbslTDA18273_GetIF(0, &uIF);
```

3.16.4 Legacy compatibility

```
err = tmbslTDA18273GetIF(0, &uIF);
```

3.17 tmbslTDA18273_Get_CF_Offset

3.17.1 Description

Gets the center frequency offset in TDA18273 driver. Mainly used in Analog standards with offset between center of the canal and picture video carrier.

3.17.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32* *puOffset***: Pointer to the object in which the CF Offset Frequency in Hz must be stored.

3.17.3 Example

```
tmErrorCode_t   err = TM_OK;
Uint32          uCF_Offset = 0;

err = tmbslTDA18273_Get_CF_Offset(0, & uCF_Offset);
```

3.17.4 Legacy compatibility

```
err = tmbslTDA18273Get_CF_Offset(0, & uCF_Offset);
```

3.18 tmbslTDA18273_WaitIRQ

3.18.1 Description

Waits IRQ to trigger and time out if not raised.

This action is automatically done in *tmbslTDA18273_HwInit*, *tmbslTDA18273_SetRF* and *tmbslTDA18273_GetPowerLevel* functions. No need to do it on top of that.

3.18.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32 *timeOut***: time-out in ms for IRQ to raise

- **UInt32 *waitStep***: time step to poll IRQ in ms.

3.18.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_WaitIRQ(0, 700, 50);
```

3.18.4 Legacy compatibility

```
err = tmbslTDA18273WaitIRQ(0, 700, 50);
```

3.19 tmbslTDA18273_GetXtalCal_End

3.19.1 Description

Gets XtalCal_End bit.

This action is automatically done in *tmbslTDA18273_HwInit* function. No need to do it on top of that.

3.19.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbXtalCalEnd**: XtalCal_End triggered

3.19.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bXtalCalEnd;

err = tmbslTDA18273_GetXtalCal_End(0,& bXtalCalEnd);
```

3.19.4 Legacy compatibility

```
err = tmbslTDA18273GetXtalCal_End(0,& bXtalCalEnd);
```

3.20 tmbslTDA18273_Set_IRQWait

3.20.1 Description

Set flag if all driver functions has to wait or not IRQ (exluding hardware init).

3.20.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbWait**: Flag IRQ Wait

3.20.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bWait = True;

err = tmbslTDA18273_Set_IRQWait(0, bWait);
```

3.20.4 Legacy compatibility

```
err = tmbslTDA18273Set_IRQWait(0, bWait);
```

3.21 tmbslTDA18273_Get_IRQWait

3.21.1 Description

Gives flag if all driver functions wait IRQ or not (exluding hardware init).

3.21.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbWait**: Flag IRQ Wait

3.21.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bWait;

err = tmbslTDA18273_Get_IRQWait(0,&bWait);
```

3.21.4 Legacy compatibility

```
err = tmbslTDA18273Get_IRQWait(0,&bWait);
```

3.22 tmbslTDA18273_SetIRQWaitHwInit

3.22.1 Description

Set flag if hardware init driver function has to wait or not IRQ.

3.22.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbWaitHwInit**: Flag IRQ Wait HwInit

3.22.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bWaitHwInit = True;

err = tmbslTDA18273_Set_IRQWaitHwInit(0, bWaitHwInit);
```

3.22.4 Legacy compatibility

```
err = tmbslTDA18273SetIRQWaitHwInit(0, bWaitHwInit);
```

3.23 tmbslTDA18273_GetIRQWaitHwInit

3.23.1 Description

Gives Flag information if hardware init driver function waits IRQ.

3.23.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbWait**: Flag IRQ Wait

3.23.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bWaitHwInit = True;

err = tmbslTDA18273_Get_IRQWaitHwInit(0,&bWaitHwInit);
```

3.23.4 Legacy compatibility

```
err = tmbslTDA18273GetIRQWaitHwInit(0,&bWaitHwInit);
```

3.24 tmbslTDA18273_GetIRQ

3.24.1 Description

Gives information if IRQ raised or not (flag IRQ Status).

3.24.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbIRQ**: Flag IRQ status

3.24.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bIRQ = True;

err = tmbslTDA18273_GetIRQ(0,&bIRQ);
```

3.24.4 Legacy compatibility

```
err = tmbslTDA18273GetIRQ (0,&bIRQ);
```

3.25 tmbslTDA18273_GetXtalCal_End

3.25.1 Description

Gives information if it is end of Cristal calibration (flag XtalCal_End).

3.25.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

-**Bool* pbXtalCal_End**: Flag End of Cristal calibration

3.25.3 Example

```
tmErrorCode_t   err = TM_OK;
Bool bIRQ = True;

err = tmbslTDA18273_GetXtalCal_End (0,&bXtalCal_End);
```

3.25.4 Legacy compatibility

```
err = tmbslTDA18273GetXtalCal_End (0,&bXtalCal_End);
```

3.26 tmbslTDA18273_ Write

3.26.1 Description

Write into part or entire register.

3.26.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **const TDA18273_BitField_t*** pBitField, /* I: Bitfield structure */.

- **UInt8** uData, /* I: Data to write */

- **eBusAccess_t** eBusAccess /* I: Access to bus */

The list of const TDA18273_BitField_t is defined in tmbslTDA18273_RegDef.c file.

eBusAccess_t is enumeration definin the type of access:

- Bus_RW Read & write in HardWare
- Bus_NoRead Read in Memory, Write in HardWare
- Bus_NoWrite Read in HardWare, Write in Memory
- Bus_None Read & write in memory

3.26.3 Example

Write a new value on AGC1_TOP symbol from register AGC1_byte_1 assuming entire register value was read before and available in memory.

```
tmErrorCode_t   err = TM_OK;
UInt8 Value = 0x03;

err = tmbslTDA18273_Write(0, gTDA18273_Reg_AGC1_byte_1__AGC1_TOP,
Value , Bus_NoRead );
```

3.26.4 Legacy compatibility

```
err = tmbslTDA18273Write(0, gTDA18273_Reg_AGC1_byte_1__AGC1_TOP,
Value , Bus_Noread );
```

3.27 tmbslTDA18273_ Read

3.27.1 Description

Read into part or entire register (from memory or Hardware).

3.27.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **const TDA18273_BitField_t*** pBitField, /* I: Bitfield structure */.

- **UInt8*** puData, /* o: Data read */

- **eBusAccess_t** eBusAccess /* I: Access to bus */

The list of const TDA18273_BitField_t is defined in tmbslTDA18273_RegDef.c file.

eBusAccess_t is enumeration definin the type of access:

- Bus_RW Read & write in HardWare
- Bus_NoRead Read in Memory, Write in HardWare
- Bus_NoWrite Read in HardWare, Write in Memory
- Bus_None Read & write in memory

3.27.3 Example

Write a new value on AGC1_TOP symbol from register AGC1_byte_1 assuming entire register value was read before and available in memory.

```
tmErrorCode_t   err = TM_OK;
UInt8 Value;

err = tmbslTDA18273_Read(0, gTDA18273_Reg_AGC1_byte_1__AGC1_TOP,
&Value , Bus_NoWrite );
```

3.27.4 Legacy compatibility

```
err = tmbslTDA18273Read(0, gTDA18273_Reg_AGC1_byte_1__AGC1_TOP,
&Value , Bus_NoWrite );
```

3.28 tmbslTDA18273_GetSWVersion

3.28.1 Description

Gets the version of the driver.

3.28.2 Parameters

- **ptmSWVersion_t *pSWVersion***: Pointer to the structure in which the version must be stored.

3.28.3 Example

```
tmErrorCode_t   err = TM_OK;
tmSWVersion_t   swVersion;

err = tmbslTDA18273_GetSWVersion(0, &swVersion);
```

3.28.4 Legacy compatibility

```
err = tmbslTDA18273GetSWVersion(0, &swVersion);
```

3.29 tmbslTDA18273_GetSWSettingsVersion

3.29.1 Description

Gets the version of the driver settings.

3.29.2 Parameters

- **ptmSWSettingsVersion_t *pSWSettingsVersion***: Pointer to the structure in which the settings version must be stored.

3.29.3 Example

```
tmErrorCode_t           err = TM_OK;
tmSWSettingsVersion_t   swSettingsVersion;

err = tmbslTDA18273_GetSWSettingsVersion(0, &swSettingsVersion);
```

3.29.4 Legacy compatibility

```
err = tmbslTDA18273GetSWSettingsVersion(0, &swSettingsVersion);
```

3.30 tmbslTDA18273_CheckHWVersion

3.30.1 Description

Checks the version of TDA18273 Hardware. Returns TM_OK if ES2 detected (major version 1, minor version 1).

3.30.2 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

3.30.3 Example

```
tmErrorCode_t   err = TM_OK;

err = tmbslTDA18273_CheckHWVersion(0);
```

3.30.4 Legacy compatibility

```
err = tmbslTDA18273CheckHWVersion(0);
```

4. Description of the structures

4.1 tmbslFrontEndDependency_t.

This structure contains the following objects:

- **tmbslFrontEndIoFunc_t *sIo***: In/out functions;
- **tmbslFrontEndTimeFunc_t *sTime***: Timer functions:
- **tmbslFrontEndDebugFunc_t *sDebug***: Debug functions:
- **tmbslFrontEndMutexFunc_t *sMutex***: Mutex functions:
- **UInt32 *dwAdditionalDataSize***: Must be set to zero (0).
- **pVoid *pAdditionalData***: Must be set to *Null* (0) (Not used).

4.2 tmbslFrontEndIoFunc_t

This structure contains pointers to the following functions:

- **tmErrorCode_t (**Read*)**: Pointer to the user-written read function (Mandatory).
- **tmErrorCode_t (**Write*)**: Pointer to the user-written write function (Mandatory).

4.3 tmbslFrontEndTimeFunc_t

This structure contains pointers to the following functions:

- **tmErrorCode_t (**Get*)**: Must be set to *Null* (0) (Not used).
- **tmErrorCode_t (**Wait*)**: Pointer to the user-written wait function (Mandatory).

4.4 tmbslFrontEndDebugFunc

This structure contains pointers to the following functions:

- **void (**Print*)**: Pointer to the user-written print function or *Null* if debug not used.

4.5 tmbslFrontEndMutexFunc

This structure contains pointers to the following functions:

- **void (*Open)**: Pointer to the user-written Open function or *Null* if mutex not used.
- **void (*Close)**: Pointer to the user-written Close function or *Null* if mutex not used.
- **void (**Acquire*)**: Pointer to the user-written Acquire function or *Null* if mutex not used.
- **void (**Release*)**: Pointer to the user-written Release function or *Null* if mutex not used.

4.6 tmSWVersion_t

This structure contains the following objects:

- **UInt32 *compatibilityNr***: Interface compatibility number.
- **UInt32 *majorVersionNr***: Interface major version number.
- **UInt32 *minorVersionNr***: Interface minor version number.
- **UInt32 *buildVersionNr***: Interface build version number.

4.7 tmSWSettingsVersion_t

This structure contains the following objects:

- **UInt32 *customerNr***: Settings customer number.
- **UInt32 *projectNr***: Settings project number.
- **UInt32 *majorVersionNr***: Settings major version number.
- **UInt32 *minorVersionNr***: Settings minor version number.

5. Description of the user-written functions

5.1 UserWritten_Read

5.1.1 Prototype

tmErrorCode_t *UserWritten_Read*(tmUnitSelect_t *tUnit*, UInt32 *AddrSize*, UInt8* *pAddr*, UInt32 *ReadLen*, UInt8* *pData***)**

5.1.2 Description

This function will be called by the driver to read registers on the TDA18273.

It returns an error code (*TM_OK* (0) if no error).

5.1.3 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.
- **UInt32 *AddrSize***: Size of the address in bytes.
- **UInt8* *pAddr***: Pointer to the address of the first register to be read.
- **UInt32 *ReadLen***: Number of registers which must be read.
- **UInt8* *pData***: Table where the read values must be stored.

5.1.4 Example

```
tmErrorCode_t err;
UInt8 uAddress = 0x0A;
UInt8 pData[3] = { 0x01, 0xBC, 0xE0};
err = UserWritten_Read(0, 1, &uAddress, 3, pData);
```

In this case the *UserWrittenRead* function must write 0x01 in the register 0xA, 0xBC in the register 0x0B and 0xE0 in the register 0x0C, on the chip identified by the number 0;

5.2 UserWritten_Write

5.2.1 Prototype

tmErrorCode_t *UserWritten_Write* (tmUnitSelect_t *tUnit*, UInt32 *AddrSize*, UInt8* *pAddr*, UInt32 *WriteLen*, UInt8* *pData***)**

5.2.2 Description

This function will be called by the driver to write registers on the TDA18273.

It returns an error code (0 for no error).

5.2.3 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.
- **UInt32 *AddrSize***: Size of the address in bytes.
- **UInt8* *pAddr***: Pointer to the address of the first register to be written.
- **UInt32 *WriteLen***: Number of registers which must be written.
- **UInt8* *pData***: Table where the values to write are stored.

5.2.4 Example

```
tmErrorCode_t err;
UInt8 uAddress = 0x0A;
UInt8 pData[3] = { 0x01, 0xBC, 0xE0};
err = UserWritten_Write(0, 1, &uAddress, 3, pData);
```

In this case the *UserWritten_Write* function must write 0x01 in the register 0xA, 0xBC in the register 0x0B and 0xE0 in the register 0x0C, on the chip identified by the number 0;

5.3 UserWritten_Wait

5.3.1 Prototype

tmErrorCode_t *UserWritten_Wait*(tmUnitSelect_t *tUnit*, UInt32 *tms***)**

5.3.2 Description

This function will be called by the driver to wait for a certain time.

It returns an error code (0 for no error).

5.3.3 Parameters

- **tmUnitSelect_t *tUnit***: Concerned unit.

- **UInt32 *tms***: Time to wait in ms.

5.3.4 Example

```
tmErrorCode_t err;
err = UserWritten_Wait(0, 10);
```

In this case the system must wait 10ms for the chip identified by the number 0;

5.4 UserWritten_Print

5.4.1 Prototype

void *UserWritten_Print*(UInt32 *level*, const char* *format*, ...**)**

5.4.2 Description

This function is used for debug purposes. It will be called by the driver to give some information to the system.

5.4.3 Parameters

- **UInt32 *level***: Severity level of the information given.

- **const char* *format***: Format control like in C printf function.

- **...**: Optional variable argument list like in C printf function.

5.4.4 Example

```
tmErrorCode_t err;
UInt32 uIF;
err = tmbslTDA18273_GetIF(0, &uIF);

UserWritten_Print(0, "TDA18273 IF: %d Hz", uIF);
```

6. How to use the TDA18273 driver

6.1 Preprocessor definitions to compile the driver

If you need Mutex protection in driver to protect driver instance from concurrent accesses the following preprocessor definition must be set to compile the driver:

```
-D_TVFE_IMPLEMENT_MUTEX
```

To enable trace debug messages in driver, the following preprocessor definition must be set:

```
-D_TVFE_DEBUG_TRACE
```

For debugging purpose, you can activate the following preprocessor definition to use “C” bit-fields for register map:

```
-D_TMBSL_TDA18273_ REGMAP_BITFIELD_DEFINED
```

When preprocessor TMBSL_TDA18273_ REGMAP_BITFIELD_DEFINED and in case of a Big-Endian host processor, the following preprocessor definition must be set to compile the driver:

```
-D_TARGET_PLATFORM_MSB_FIRST
```

6.2 Source files directories

The following source files must be compiled:

```
.\tmbslTDA18273\src\tmbslTDA18273.c
.\tmbslTDA18273\src\tmbslTDA18273_Instance.c
.\tmbslTDA18273\src\tmbslTDA18273_RegDef.c
```

6.3 Header files directories

The following include directories must be added to the compiling environment:

```
.\inc
.\tmbslTDA18273\cfg
.\tmbslTDA18273\inc
```

6.4 Header files

The following provided header files must be included:

```
#include "tmNxTypes.h"
#include "tmNxCompId.h"
#include "tmFrontEnd.h"
#include "tmbslFrontEndTypes.h"

#include "tmbslTDA18273.h"
```

6.5 Initialization of the TDA18273

6.5.1 Description

The driver is set-up using the function *tmbslTDA18273_Open*. The *tmbslTDA18273_Open* function accepts as parameter a structure containing pointers to the functions used for communication, time and debug. The definitions of these functions are given in the structures *tmbslFrontEndIoFunc_t*, *tmbslFrontEndTimeFunc_t* and *tmbslFrontEndDebugFunc* in the file *tmbslFrontEndtypes.h* provided with the driver.

Only the *UserWritten_Read*, *UserWritten_Write* and *UserWritten_Wait* functions are mandatory

Once the driver is properly set-up with *tmbslTDA18273_Open*, TDA18273 hardware should be initialized with *tmbslTDA18273_HwInit*.

6.5.2 Example

*Initializes TDA18273 (TunerUnit=**0**):*

```
tmbslFrontEndDependency_t   sSrvTunerFunc;
UInt32                      TunerUnit = 0;

sSrvTunerFunc.sIo.Write             = UserWritten_I2CWrite;
sSrvTunerFunc.sIo.Read              = UserWritten_I2CRead;
sSrvTunerFunc.sTime.Get             = Null;
sSrvTunerFunc.sTime.Wait            = UserWritten_Wait;
sSrvTunerFunc.sDebug.Print          = UserWritten_Print;
sSrvTunerFunc.sMutex.Open           = Null;
sSrvTunerFunc.sMutex.Close          = Null;
sSrvTunerFunc.sMutex.Acquire        = Null;
sSrvTunerFunc.sMutex.Release        = Null;
sSrvTunerFunc.dwAdditionalDataSize  = 0;
sSrvTunerFunc.pAdditionalData       = Null;

/* Open TDA18273 driver instance */
err = tmbslTDA18273_Open(TunerUnit, &sSrvTunerFunc);
if(err == TM_OK)
{
  /* TDA18273 Hardware initialization */
  err = tmbslTDA18273_HwInit(TunerUnit);
}
```

6.6 How to tune the TDA18273 to a frequency

6.6.1 Description

Once the TDA18273 is initialized, a standard mode preset must be set through *tmbslTDA18273_SetStandardMode* function.

Next step is to tune TDA18273 to a desired frequency with *tmbslTDA18273_SetRF*.

Note: *tmbslTDA18273_SetStandardMode* should be called at least for the first tuning and each time the standard is changed.

6.6.2 Example

*Tunes TDA18273 (TunerUnit=**0**) to 770.166 MHz in DVBT 8MHz standard mode:*

```
UInt32                      TunerUnit = 0;
tmTDA18273StandardMode_t    TDA18273StdMode = tmTDA18273_DVBT_8MHz;
UInt32                      uRF = 0;
tmbslFrontEndState_t        PLLLock = tmbslFrontEndStateUnknown;

 /* TDA18273 standard mode if changed */
err = tmbslTDA18273_SetStandardMode(TunerUnit, TDA18273StdMode);

if(err == TM_OK)
{
  /* TDA18273 RF frequency 770.166 MHz */
  uRF = 770166000;
  err = tmbslTDA18273_SetRF(TunerUnit, uRF);
}

if(err == TM_OK)
{
  /* Get TDA18273 PLL Lock status */
  err = tmbslTDA18273_GetLockStatus(TunerUnit, &PLLLock);
}
```

6.7 Sample code to initialize and tune the TDA18273 to a frequency

```
#include "tmNxTypes.h"
#include "tmNxCompId.h"
#include "tmFrontEnd.h"
#include "tmbslFrontEndTypes.h"

#include "tmbslTDA18273.h"


int main(void)
{
  tmErrorCode_t               err = TM_OK;
  tmbslFrontEndDependency_t   sSrvTunerFunc;
  UInt32                      TunerUnit = 0;
  tmTDA18273StandardMode_t    TDA18273StdMode = tmTDA18273_DVBT_8MHz;
  UInt32                      uRF = 0;
  tmbslFrontEndState_t        PLLLock = tmbslFrontEndStateUnknown;


  sSrvTunerFunc.sIo.Write             = UserWritten_I2CWrite;
  sSrvTunerFunc.sIo.Read              = UserWritten_I2CRead;
  sSrvTunerFunc.sTime.Get             = Null;
  sSrvTunerFunc.sTime.Wait            = UserWritten_Wait;
  sSrvTunerFunc.sDebug.Print          = UserWritten_Print;
  sSrvTunerFunc.sMutex.Open           = Null;
  sSrvTunerFunc.sMutex.Close          = Null;
  sSrvTunerFunc.sMutex.Acquire        = Null;
  sSrvTunerFunc.sMutex.Release        = Null;
  sSrvTunerFunc.dwAdditionalDataSize  = 0;
  sSrvTunerFunc.pAdditionalData       = Null;

  /* Open TDA18273 driver instance */
  err = tmbslTDA18273_Open(TunerUnit, &sSrvTunerFunc);
  if(err == TM_OK)
  {
    /* TDA18273 Hardware initialization */
    err = tmbslTDA18273_HwInit(TunerUnit);

    if(err == TM_OK)
    {
      /* TDA18273 standard mode if changed */
       err = tmbslTDA18273_SetStandardMode(TunerUnit,
                                           TDA18273StdMode);
    }

    if(err == TM_OK)
    {
      /* TDA18273 RF frequency 770.166 MHz */
      uRF = 770166000;
      err = tmbslTDA18273_SetRF(TunerUnit, uRF);
    }


    if(err == TM_OK)
```

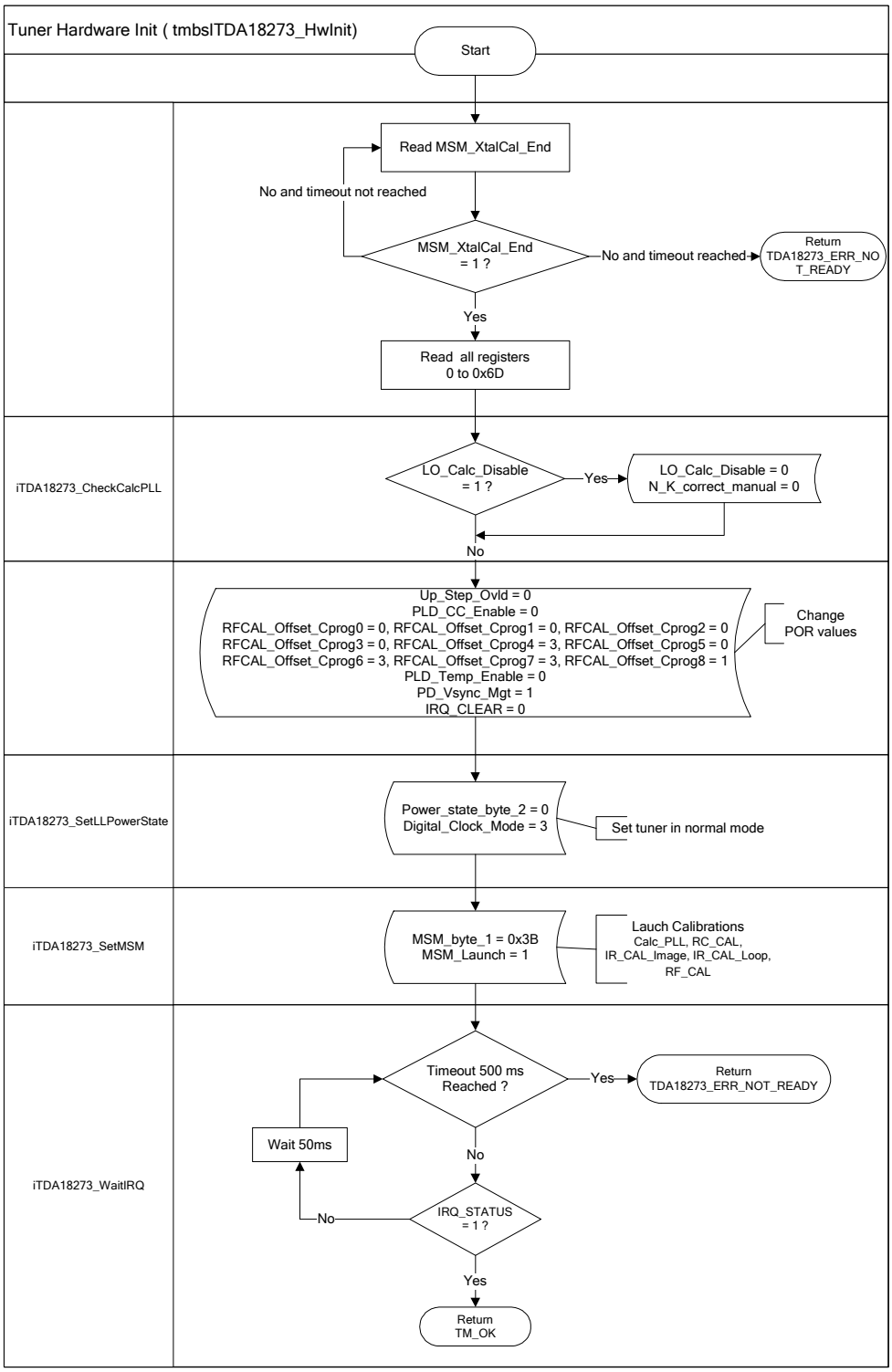
```
    {
      /* Get TDA18273 PLL Lock status */
      err = tmbslTDA18273_GetLockStatus(TunerUnit, &PLLLock);
    }


    if(err == TM_OK)
    {
      /* Put TDA18273 in standby mode: */
      err = tmbslTDA18273_SetPowerState(TunerUnit,
                                        tmPowerStandby);
    }


    /* Close TDA18273 driver instance */
    err = tmbslTDA18273_Close(TunerUnit);
  }

  return err;
}
```

6.8 Return values of the functions

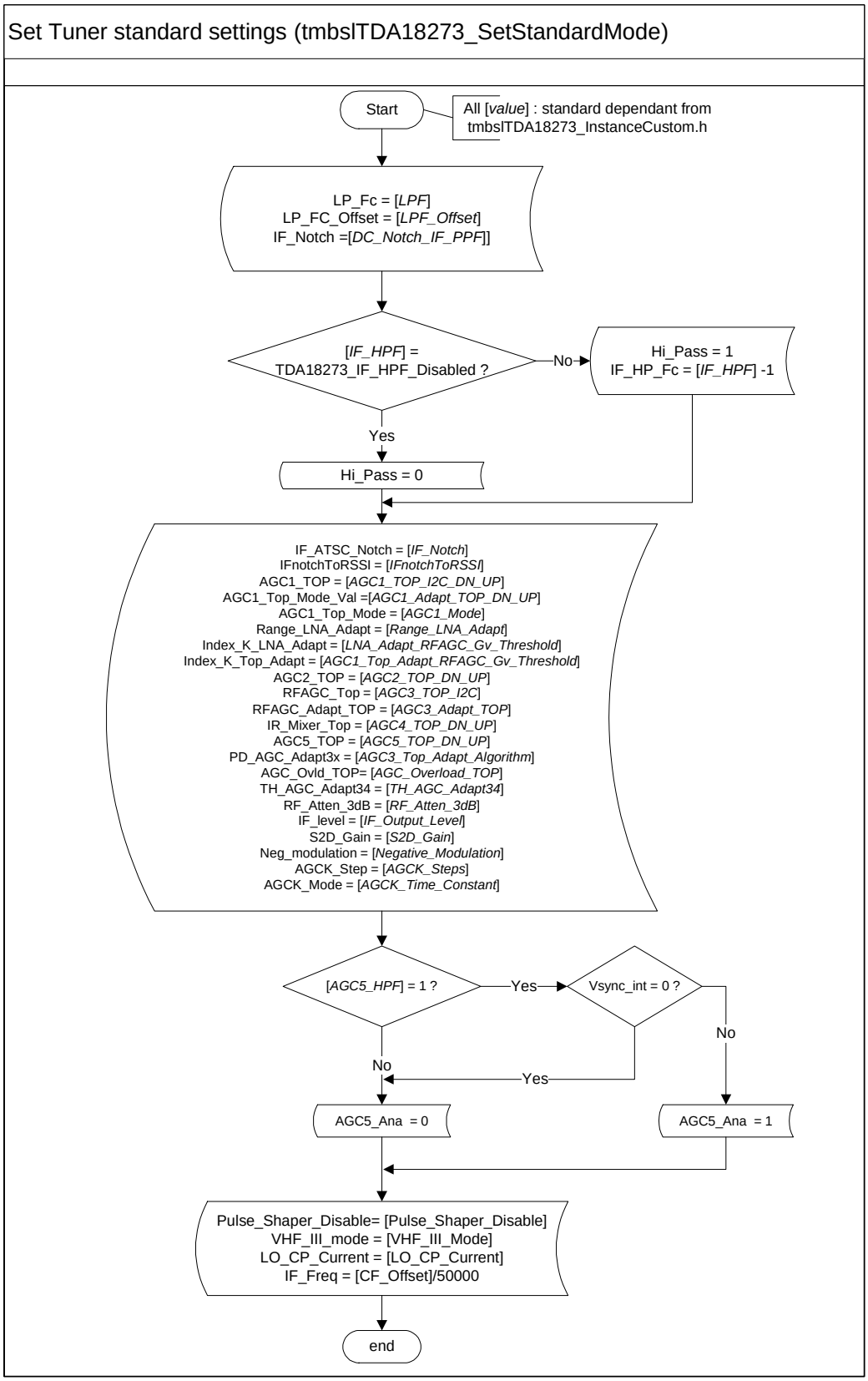
Every function of the driver returns an error message. If the function has been successfully executed the return error message is *TM_OK*. If the function was not successfully executed, the error message corresponding to the occurred error is returned. The error messages are listed in the files *tmCompId.h* and *tmbslTDA18273.h* provided with the driver.

7. ANNEXES

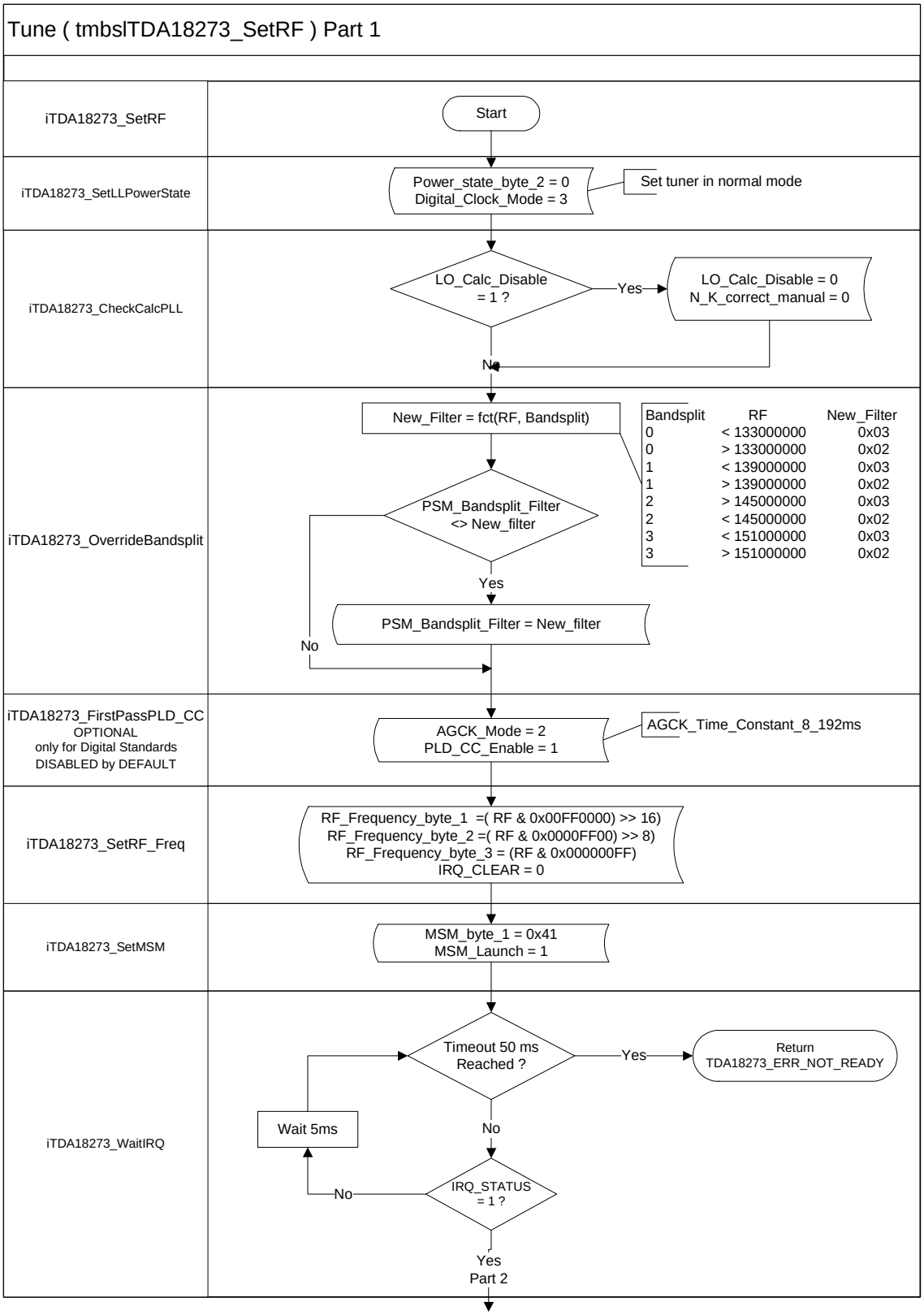
7.1 TDA18273 Hardware init sequence

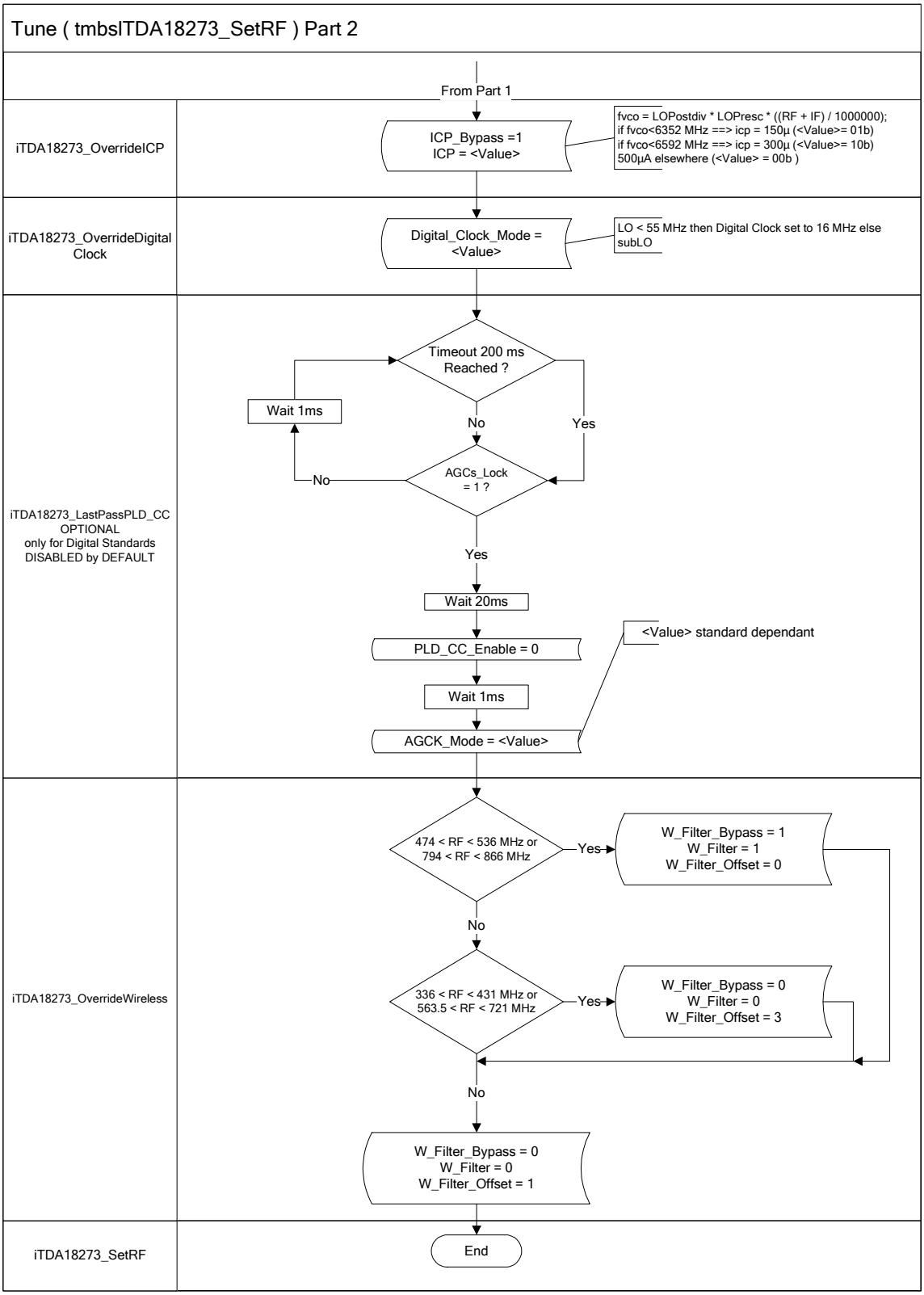


7.2 TDA18273 TV standards settings sequence



7.3 TDA18273 RF tune sequence





8. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. NXP Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify NXP Semiconductors for any damages resulting from such application.

Right to make changes — NXP Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). NXP Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

9. Licenses

10. Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — <patent owner>

11. Trademarks

<trademark name> — is a trademark of <owner>

<registered trademark name> — is a registered trademark of <owner>

12. Contents

Date of release:20/01/2010
Document number:UM1001

Published in Caen, France

Date of release:20/01/2010
Document number:UM1001

Published in Caen, France

Date of release:20/01/2010
Document number:UM1001

Published in Caen, France