BLP25RFE001 - User Guide

Programing User Guide

AMPLEON

Rev. 0.2 — 22 March 2018 **Application note**

Document information

Info	Content
Keywords	RF heating
Abstract	User guide of the BLP25RFE001 software driver. Description of the public functions and how to use the source code.

Revision history

Rev	Date	Description
1.0	27 Jan 2015	Creation
1.1	4 Feb 2015	Add flowcharts for initialization and devices synchronization
1.2	22 March 2018	Changed formatting

Contact information

For more information, please visit: http://www.ampleon.com

For sales office addresses, please visit: http://www.ampleon.com/sales

1. Introduction

This document contains a description of the public functions of the BLP25RFE001 software driver.

Volcano is the initial project nickname of the BLP25RFE001 product. In this document and the software driver, Volcano always refers to the BLP25RFE001 product.

2. Package contents

The driver package contains the following components:

- A header file containing the public functions of the driver
    - Volcano.h
- A header file containing the private functions and internal data types of the driver
    - Volcano_local.h
- A source file containing the code of all the functions
    - Volcano.c

3. Description of public functions of the driver

3.1 Overview

The driver can be fully controlled by the following functions:

```
Volcano_Open
Volcano_Close
Volcano_HwInit
Volcano_SetRF
Volcano_ResetSynchro
Volcano_SetSynchroMaster
Volcano_SetSynchroSlave
Volcano_SetPowerStateMode
Volcano_SetRFPhase
Volcano_SetRFAtten
Volcano_GetPowerStateMode
Volcano_GetAtrfPhase
Volcano_GetThermo
Volcano_Write
Volcano_Read
Volcano_ReadRegMap
```

3.2 Volcano_Open

3.2.1 Description

Initializes the driver instance. No hardware access is performed in this function. The function *Volcano_Open* function must be called before *Volcano_HwInit* (hardware initialization).

3.2.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number (for a single BLP25RFE001 device).

- **tmUnitSelect_t tUnitD**: Device unit number (used only for OM15000 or SANGO board, otherwise to be set to 0).

- **SysDependency_t *psSrvFunc**: Structure containing the hardware access functions and the time functions.

3.3 Volcano_Close

3.3.1 Description

De-initializes the driver instance. Must be called before calling *Volcano_Open* again if already initialized.

3.3.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.

3.4 Volcano_HwInit

3.4.1 Description

Initializes the hardware of the device matching the device unit number.

3.4.2 Parameters

- **tmUnitSelect_t *tUnit***: Device unit number.

- **Volcano__ModeMS_t *uModeMS***: Device role (Master or Slave).

3.5 Volcano_SetRF

3.5.1 Description

Tunes the device to the selected frequency in Hertz.

3.5.2 Parameters

- **tmUnitSelect_t *tUnit***: Device unit number.
- **UInt32 *uLO***: Local Oscillator frequency (in Hz).
- **Bool *bBlanking***: PA in power down during SetRF.

3.6 Volcano_ResetSynchro

3.6.1 Description

Activates the LO_CHAIN synchronization.

3.6.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.

3.7 Volcano_SetSynchroMaster

3.7.1 Description

Activates the LO_CHAIN synchronization for the master device.

3.7.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.

3.8 Volcano_SetSynchroSlave

3.8.1 Description

Activates the LO_CHAIN synchronization for the slave device.

3.8.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.

3.9 Volcano_SetPowerStateMode

3.9.1 Description

Manages the power down state according to the desired state mode.

3.9.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **Volcano_PowerStateMode_t PowerStateMode**: Desired power state mode.

3.10 Volcano_SetRFPhase

3.10.1 Description

Tunes the RF output signal to the selected phase.

3.10.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt16 Phase**: Phase value in degree.

3.11 Volcano_SetRFAtten

3.11.1 Description

Tunes the attenuation of the RF output signal.

3.11.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt16 Atten**: RF attenuation value in 1/256 dB steps.

3.12 Volcano_GetPowerStateMode

3.12.1 Description

Get the current power state mode.

3.12.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **Volcano_PowerStateMode_t *pPowerStateMode**: Pointer to the current power state mode.

3.13 Volcano_GetAtrfPhase

3.13.1 Description

Launches and get back ATRF phase in steps of 1.4°.

3.13.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt32 *pATRF_Phase**: Pointer to the ATRF phase in steps of 1.4°.

3.14 Volcano_GetAtrfPhase

3.14.1 Description

Returns the IC temperature in degrees Celsius.

3.14.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt8 * pThermo**: Pointer to the IC temperature in degrees Celsius.

3.15 Volcano_Write

3.15.1 Description

Writes in the device hardware registers.

3.15.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- const **Volcano_BitField_t *pBitField**: Pointer to the device register field.
- **UInt8 uData**: Data to write.

3.16 Volcano_Read

3.16.1 Description

Reads in the device hardware registers.

3.16.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- const **Volcano_BitField_t *pBitField**: Pointer to the device register field.
- **UInt8 *puData**: Data to read.

3.17 Volcano_ReadRegMap

3.17.1 Description

Updates the cached register map from the device hardware.

3.17.2 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt8 uAddress**: Data address to read.
- **UInt32 uReadLen**: Number of data to read (in bytes).

4. Description of the structures

4.1 SysDependency_t

This structure contains the following objects:

- **IoFunc_t *sIo***: In/out functions

- **TimeFunc_t *sTime***: Timer functions

4.2 IoFunc_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).

- **tmErrorCode_t (**WriteRead*)**: Pointer to the user-written write-then-read function (mandatory).

4.3 TimeFunc_t

This structure contains pointers to the following functions:

- **tmErrorCode_t (**Wait*)**: Pointer to the user-written wait function (mandatory).

4.4 Volcano_BitField_t

This structure contains pointers to the following fields:

- **UInt8 Address**: Register address.

- **UInt8 PositionInBits**: Bit field position.

- **UInt8 WidthInBits**: Number of bits in the bit field.

- **UInt8 Attributes**: Not used.

5. Description of the user written functions

The prototypes can be found in the file *Volcano.h*, in the structure type definition of *SysDependency_t*.

5.1 Read

5.1.1 Prototype

tmErrorCode_t Read (tmUnitSelect_t tUnit,UInt32 ReadBitsLen, UInt8* pData)

5.1.2 Description

This function is currently not used and can remain a dummy function for the user. The Read function is performed through the WriteRead function.

5.2 Write

5.2.1 Prototype

tmErrorCode_t Write (tmUnitSelect_t tUnit,UInt32 WriteBitsLen, UInt8* pData)

5.2.2 Description

This function will be called by the driver to write registers on the BLP25RFE001 device.

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

The pData array must contain:

- 2 bits of operand code (00b for write)
- 6 bits of address
- 8 bits of data

Note: the Write function does not need to support the burst mode as it is not used in the BLP25RFE001 driver.

5.2.3 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt32 WriteBitsLen**: Number of bits which must be written.
- **UInt8* pData**: Table containing operand code, register address and value to write.

5.2.4 Example

```
tmErrorCode_t err;
UInt8 pData[2] = {0x01, 0xBC};
err = Write(0, 16, pData);
```

In this case the Write function must write 0xBC in the register 0x01, on the device identified by the number 0.

5.3 WriteRead

5.3.1 Prototype

tmErrorCode_t WriteRead (tmUnitSelect_t tUnit, UInt32 WriteBitsLen, UInt8* pDataWrite, UInt32 ReadBitsLen, UInt8* pDataRead)

5.3.2 Description

Basically, this function is only used to read registers from the device. This function will be called by the driver to write registers (to send the read operand code and the register address), then read them, on the BLP25RFE001 device. The burst mode is internally

supported to optimize the read of consecutive bytes, in sending only the first register address.

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

The pDataWrite array must contain:

- 2 bits of operand code (01b for read)
- 6 bits of address
- 8 bits of data (for write operand code only)

The pDataRead array must contain:

- 8 (or more) bits of data

Note: the WriteRead function must support the burst mode for reading as it is implemented in the BLP25RFE001 driver.

5.3.3 Parameters

- **tmUnitSelect_t tUnit**: Device unit number.
- **UInt32 WriteBitsLen**: Number of bits which must be written.
- **UInt8* pDataWrite**: Table containing operand code, register address and value to write.
- **UInt32 ReadBitsLen**: Number of bits which must be read (can be more than one byte).
- **UInt8* pDataRead**: Table containing operand code, register address and read value.

5.3.4 Example

```
tmErrorCode_t err;
UInt8 pDataWrite[1] = {0x41};
UInt8 pDataRead[1] = {0xFF};
err = WriteRead(0, 8, pDataWrite, 8, pDataRead);
```

In this case the WriteRead function must write 0x41 to the SPI controller to read the value at the register address 0x01, on the device identified by the number 0. An 8-bit value is stored afterwards in replacing the 0xFF value.

5.4 Wait

5.4.1 Prototype

tmErrorCode_t Wait(UInt32 tms**)**

5.4.2 Description

This function will be called by the driver to wait for a given time in milliseconds.

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

5.4.3 Parameters

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

5.4.4 Example

```
tmErrorCode_t err;
err = Wait(10);
```

In this case, the system must wait for 10ms.

6. How to use the BLP25RFE001 driver

6.1 Preprocessor definitions to compile the driver

Depending on the platform, the following define may added in the compilation line:

```
BOARD_NXP_OM15000
BOARD_NXP_OM15004C2
BOARD_NXP_SANGO4RF
```

None of them is used for a standard configuration.

6.2 Source files directories

The following source file must be compiled:

```
.\Volcano_driver\Volcano.c
```

6.3 Header files directories

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

```
.\inc
.\Volcano_driver\
```

6.4 Header files

The following provided header files must be included:

```
#include "tmNxTypes.h"
#include "tmCompId.h"
#include "tmbslFrontEndTypes.h"
#include "tmFrontEnd.h"
#include "tmUnitParams.h"
#include "tmbslFrontEndCfgItem.h"

#include "Volcano.h"
```

6.5 Customization of the BLP25RFE001

6.5.1 Introduction

The BLP25RFE001 can be customized with the *Volcano_Local.h* file.

It is located in:

```
.\Volcano_driver\Volcano_Local.h
```

In the *Volcano.c* file, an array of objects *VolcanoObject_t* is initialized in the function *Volcano_Open* before the *Volcano_HwInit*. The array name is *gVolcanoInstance[]* and allows to customize one or more devices.

The driver allows up to 4 devices to be controlled. Limiting the driver to less than 4 devices can be achieved in decreasing the define value Volcano_UNITS (at 4 by default). Allowing more than 4 devices would require to extend the size of the array *gVolcanoInstance[]*.

6.5.2 VolcanoObject_t type

This structure matches the instance of each device. It is filled at driver initialization with *Volcano_Open*.

6.5.3 Type definition

6.5.3.1 Power State

Initializes PowerStateMode field:

- Stand-by mode : Volcano_standby
- Generator without daisy chaining : Volcano_generator_no_daisy
- Generator with daisy chaining : Volcano_generator_daisy
- Slave with daisy chaining : Volcano_slave_daisy
- Slave without daisy chaining : Volcano_slave_no_daisy
- Generator PPA without daisy chaining : Volcano_generator_no_daisy_PPA
- Generator PPA with daisy chaining : Volcano_generator_daisy_PPA
- Slave PPA with daisy chaining : Volcano_slave_daisy_PPA
- Slave PPA without daisy chaining : Volcano_slave_no_daisy_PPA

6.5.3.2 Device Role

Used to configure the device role in the function *Volcano_HwInit*:

- Master : Volcano_Master
- Slave : Volcano_Slave

6.6 Initialization of the BLP25RFE001

6.6.1 Description

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

Only the Read, Write and Wait functions are used.

Once the driver is properly set-up with *Volcano_Open*, the BLP25RFE001 hardware can be initialized with *Volcano_HwInit*.

6.6.2 Example

Initialize the BLP25RFE001 like a master device (tUnit=0):

```
tmErrorCode_t err = TM_OK;
SysDependency_t   sSrvSysFunc;    /* setup parameters */

/* Low layer structure to link with user written functions */
sSrvSysFunc.sIo.Write             = UserWrittenSPIWrite;
sSrvSysFunc.sIo.WriteRead         = UserWrittenSPIWriteRead;
sSrvSysFunc.sTime.Wait            = UserWrittenWait;

/* Volcano driver initialization */
err = Volcano_Open(0, 0, &sSrvSysFunc); // master unit 0

/* Hardware init of the master device */
if (err == TM_OK)
    err = Volcano_HwInit(0, Volcano_Master);
```

6.7 How to tune the BLP25RFE001 to a frequency

6.7.1 Description

Once the BLP25RFE001 is initialized, the power state should be appropriate prior to program the device to a selected frequency.

6.7.2 Example

Tune a master BLP25RFE001 (tUnit=0) to 2400 MHz:

```
  /* set proper power state */
  if (err == TM_OK)
      err = Volcano_SetPowerStateMode(0,
Volcano_generator_no_daisy);

  /* set a frequency on the master device */
  err = Volcano_SetRF(0, 2400000, False); // in kHz
```

6.8 Sample code to drive two BLP25RFE001

6.8.1 Description

This is the complete sample code to drive two devices, one master and one slave. The frequency selected on the master will also apply to the slave.

6.8.2 Example

```
  tmErrorCode_t err = TM_OK;
  SysDependency_t   sSrvSysFunc;    /* setup parameters */

  /* Low layer structure to link with user written functions */
  sSrvSysFunc.sIo.Write             = UserWrittenSPIWrite;
  sSrvSysFunc.sIo.WriteRead         = UserWrittenSPIWriteRead;
  sSrvSysFunc.sTime.Wait            = UserWrittenWait;

  /* Driver initialization for master and slave */
  err = Volcano_Open(0, 0, &sSrvSysFunc); // master unit 0

  if (err == TM_OK)
      err = Volcano_Open(1, 1, &sSrvSysFunc); // slave unit 1

  /* Hardware init of the master device */
  if (err == TM_OK)
      err = Volcano_HwInit(0, Volcano_Master);

  /* set proper power state */
  if (err == TM_OK)
      err = Volcano_SetPowerStateMode(0,
Volcano_generator_daisy);
```

```
if (err == TM_OK)
    err = Volcano_SetPowerStateMode(1, Volcano_slave_no_daisy);

/* set a frequency on the master device */
err = Volcano_SetRF(0, 2400000, False); // in kHz

/* launch synchro reset all devices master and slave(s) */
if (err == TM_OK)
    err = Volcano_ResetSynchro(0);
if (err == TM_OK)
    err = Volcano_ResetSynchro(1);

/* launch Synchro on slave(s) device(s) */
if (err == TM_OK)
    err = Volcano_SetSynchroSlave(1);

/* launch synchro on master device */
if (err == TM_OK)
    err = Volcano_SetSynchroMaster(0);
```

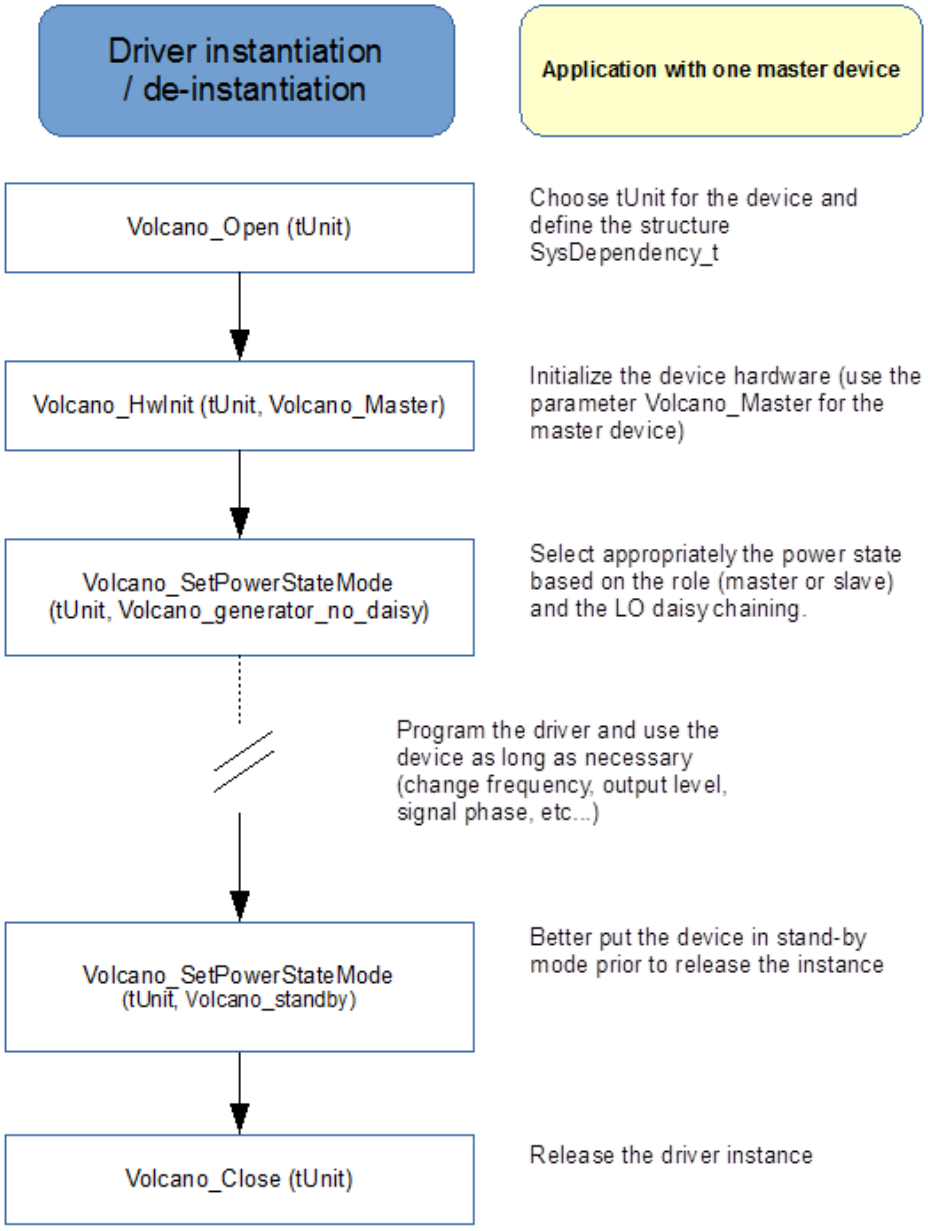
6.9 Return values of the functions

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* provided with the driver.

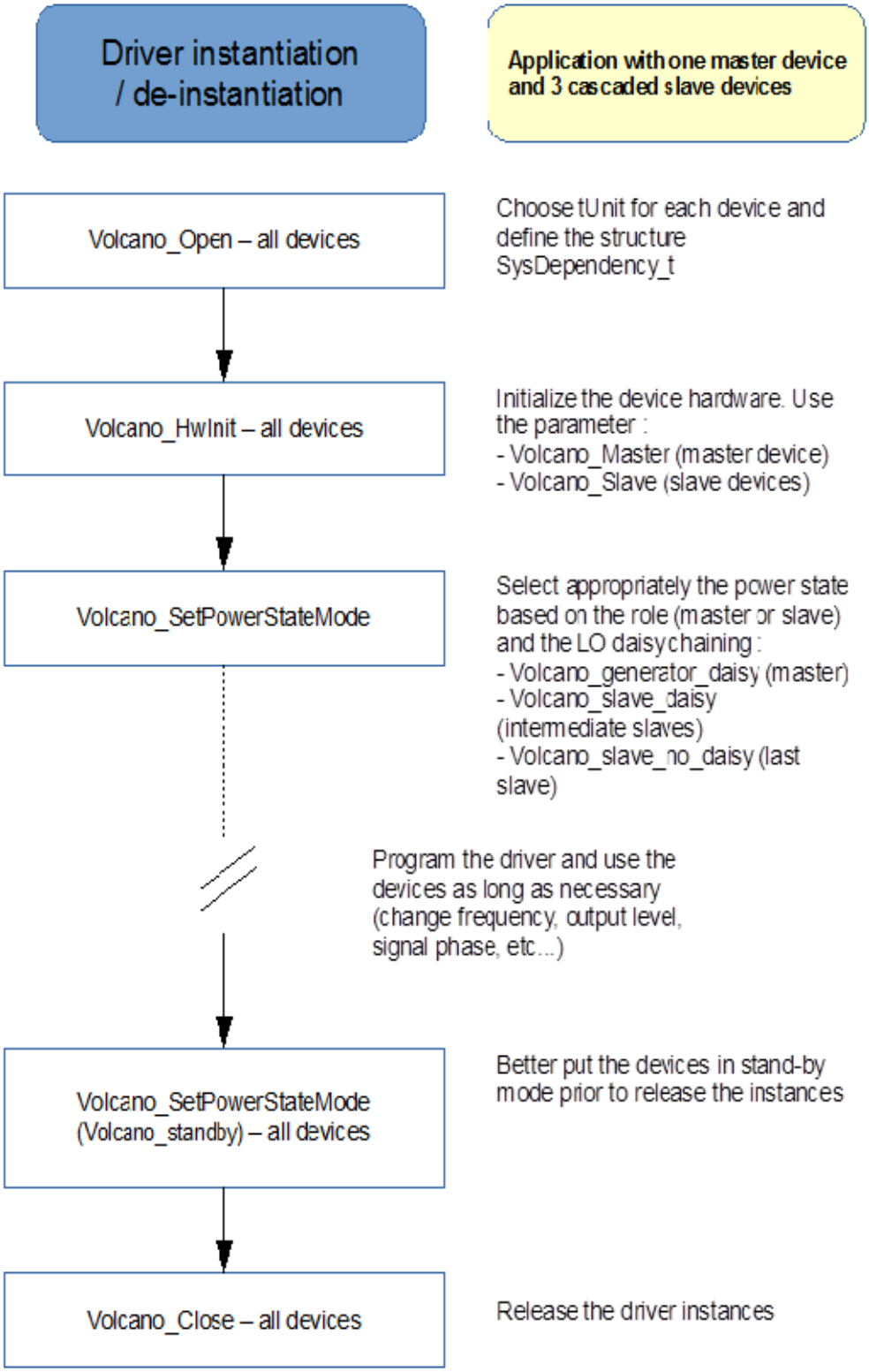
6.10 Uses cases and flowcharts

6.10.1 Software/hardware initialization

The following flowchart is given for a single master device (lighting application).

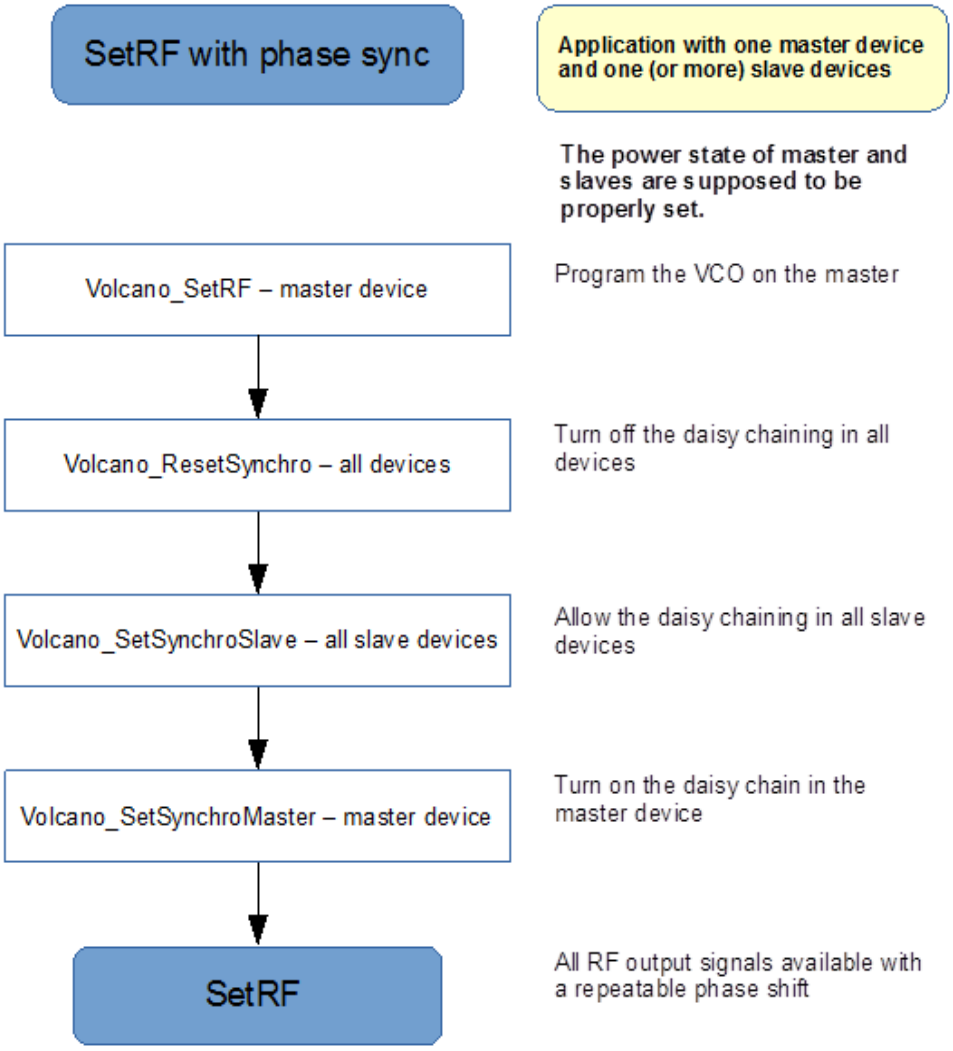


The following flowchart is given for an application with one master device and 3 cascaded slaves (cooking application).



6.10.2 Devices synchronization

In an application with one master and slaves, the master controls the signal frequency for all devices. As long as the phase of each device must be controlled (so to be identical between successive programming), a specific sequence must be followed.



7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. Ampleon does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, Ampleon does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. Ampleon takes no responsibility for the content in this document if provided by an information source outside of Ampleon.

In no event shall Ampleon be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, Ampleon' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of Ampleon.

Right to make changes — Ampleon reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — Ampleon products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an Ampleon product can reasonably be expected to result in personal injury, death or severe property or environmental damage. Ampleon and its suppliers accept no liability for inclusion and/or use of Ampleon products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

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

Customers re responsible for the design and operation of their applications and product using Ampleon products, and Ampleon accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the Ampleon product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Ampleon does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using Ampleon products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). Ampleon does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Terms and conditions of commercial sale — Ampleon products are sold subject to the general terms and conditions of commercial sale, as published at http://www.ampleon.com/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. Ampleon hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of Ampleon products by customer.

7.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

Any reference or use of any 'NXP' trademark in this document or in or on the surface of Ampleon products does not result in any claim, liability or entitlement vis-à-vis the owner of this trademark. Ampleon is no longer part of the NXP group of companies and any reference to or use of the 'NXP' trademarks will be replaced by reference to or use of Ampleon's own trademarks.

8. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, visit: http://www.ampleon.com
For sales office addresses, please visit: http://www.ampleon.com/sales

Date of release: 22 March 2018
Document identifier: BLP25RFE001-02

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, visit: http://www.ampleon.com
For sales office addresses, please visit: http://www.ampleon.com/sales

Date of release: 22 March 2018
Document identifier: BLP25RFE001-02