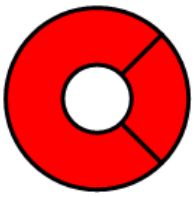
General Embedded C Library Manual

General Embedded C Libraries

An open source software library



Document status: ***Draft*** | *Preliminary* | *Release*

Github link	https://github.com/GeneralEmbeddedCLibraries
Developers	Žiga Miklošič
E-mail	ziga.miklosic@gmail.com

Revision History

Revision	Author	Date	Changes/Remarks
V1.0	*Miklošič Ž.*	*16.10.2021*	*Initial document*

Contents

1 About

Scope of that document is to describe usage and internal architecture/concept used in more complex modules that are part of larger General Embedded C Library package. Additional document will give general guidance how the modules shall be used.

2 Workflow

2.1 Module structure

Each module has identical directory structure as shown from picture bellow:

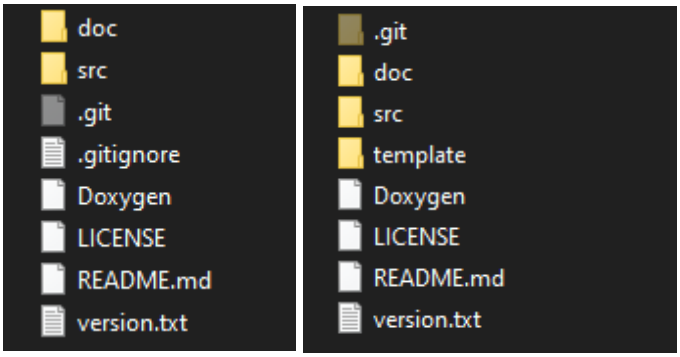
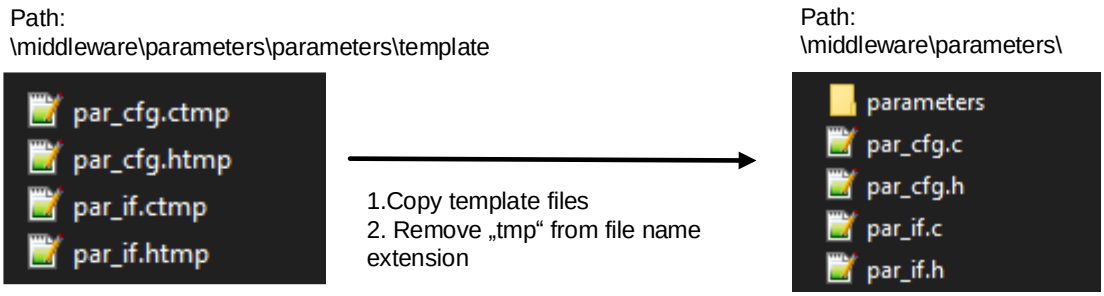


Figure 1 Top directory of each module. Simple module (left) and complex module (right)

More complex modules has also **template** folder where source code template files are prepared for user. When module has template folder its content must be copied and paste to one directory above the module directory. After coping it is mandatory to remove “tmp” string from file name extension.

E.g.:



Template files are used for module configuration for user application needs or to adapt module to specific platform needs.

2.2 Module usage

Each module usage is described in its repository main page.

2.3 Module dependencies

Each module dependencies are described in its repository main page.

2.4 Module API

Each module has Doxygen generated API description inside module **doc** directory.

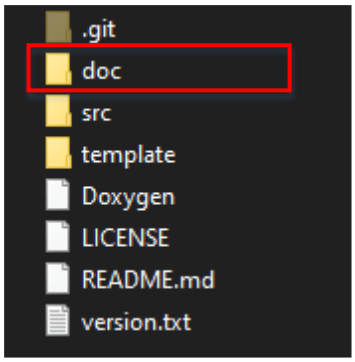


Figure 2 Module Doxygen documentation location

2.5 Git submodule

Each module shall be added to main project as git submodule.

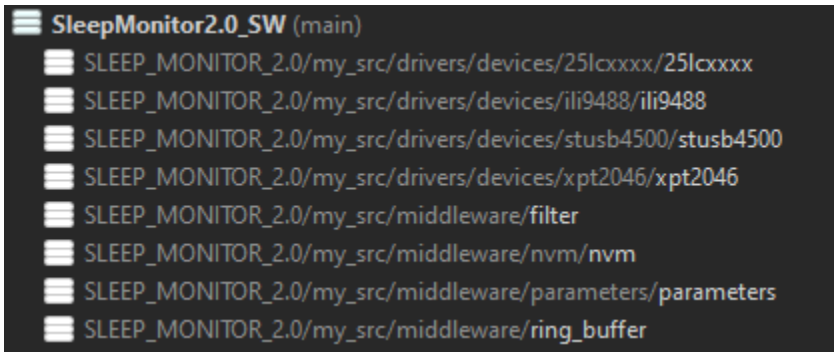


Figure 3 Using submodules with SmartGit git client tool

3 Parameters

3.1 Features

- Parameter configuration via single table
- Supported up to 8 data types: u8, i8, u16, i16, u32, i32, f32, string
- Each parameter has unique ID
- Access type configurable: Read only or Write-Read
- Storing & loading parameters from/to NVM on the fly
- Support of single parameter store (if NVM layer supports it)

Link to github: https://github.com/GeneralEmbeddedCLibraries/parameters

3.2 Exceptions

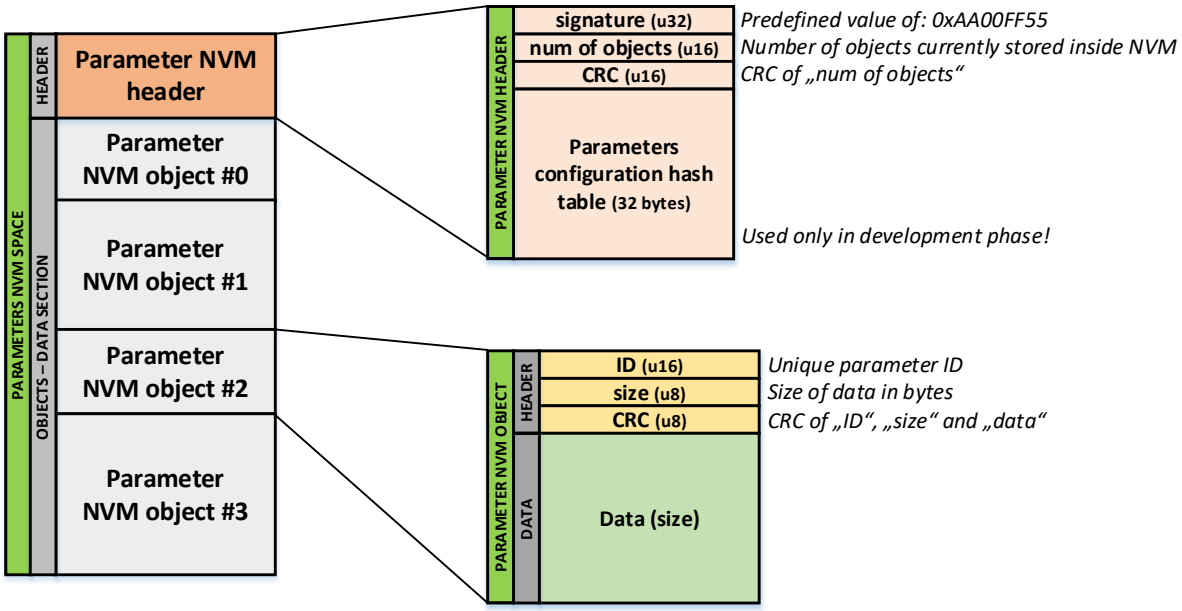
3.2.1 Loading parameters from NVM

In case parameter NVM header is corrupted all parameters will be set to default and old parameters stored inside NVM will be rewritten with new parameters. If that scenario occurs parameters stored inside NVM will not be compatible with old software.

In case single parameter NVM object is corrupted then complete parameters are cleared and completely rewritten with current definition of parameter table. If that scenario occurs parameters stored inside NVM will not be compatible with old software.

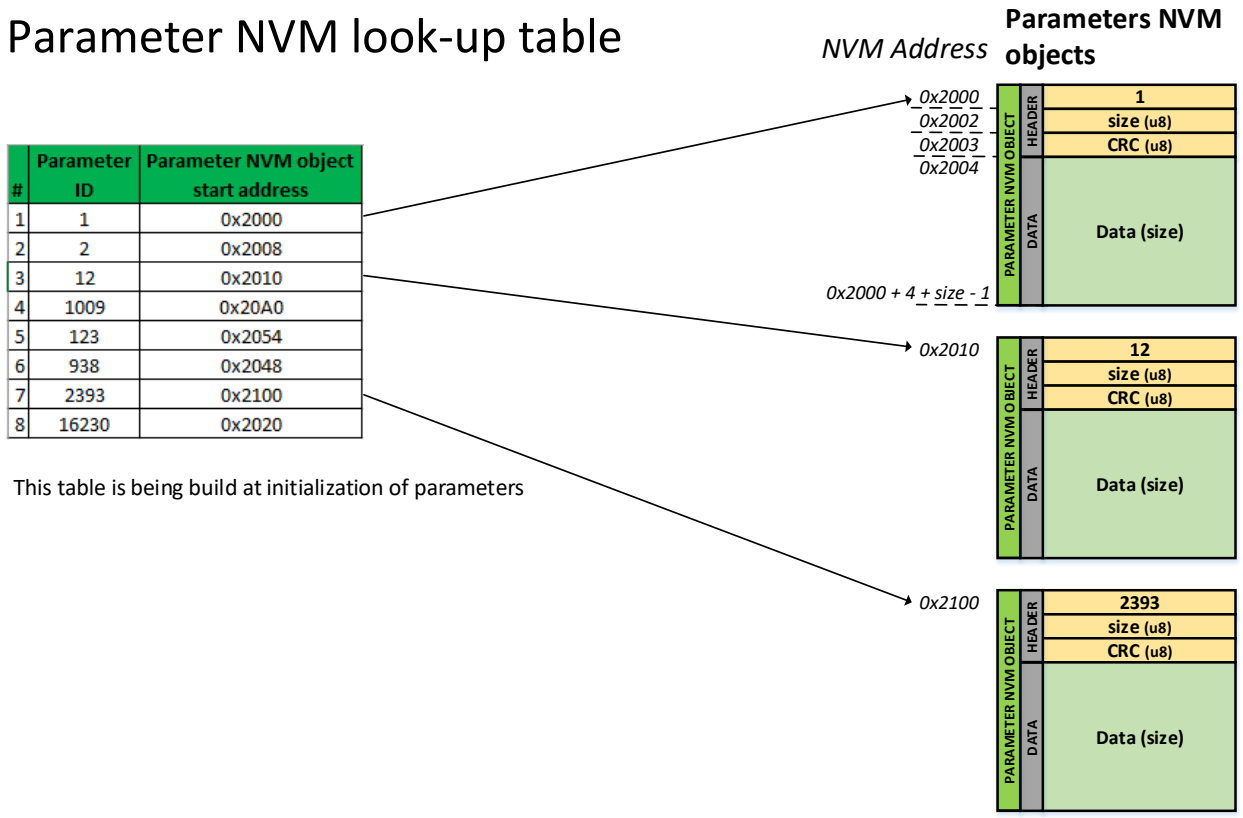
3.4 Persistent parameters

3.4.1 NVM structure



3.5.1 NVM look-up table

Purpose of NVM look-up table is linkage between parameter ID and its start location in NVM space. When storing single parameter to NVM this information is mandatory to know.



3.6.1 Loading persistent parameters from NVM at start-up

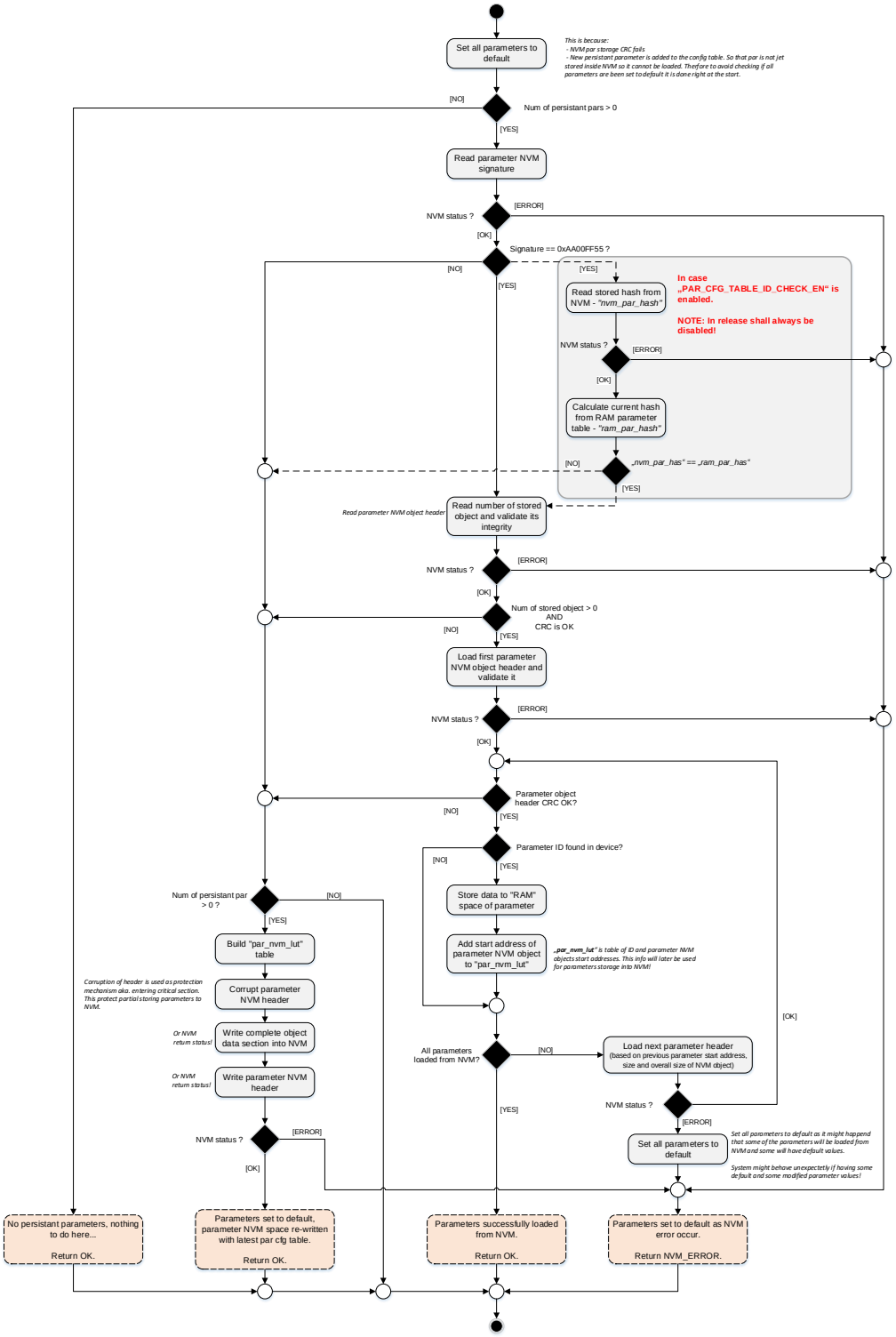


Figure 4 Loading parameter at start-up

3.6.2 Storing all persistent parameter to NVM

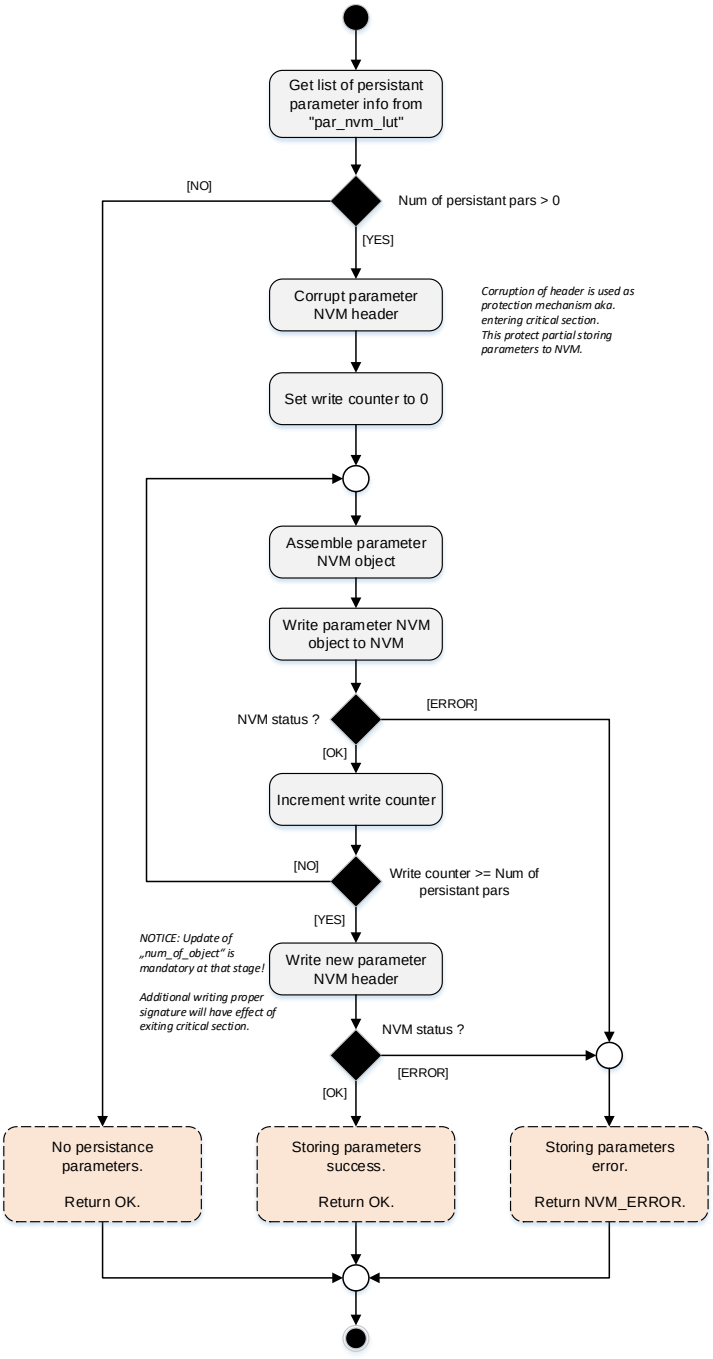


Figure 5 Storing persistent parameter to NVM during runtime procedure

3.6.3 Storing single parameter

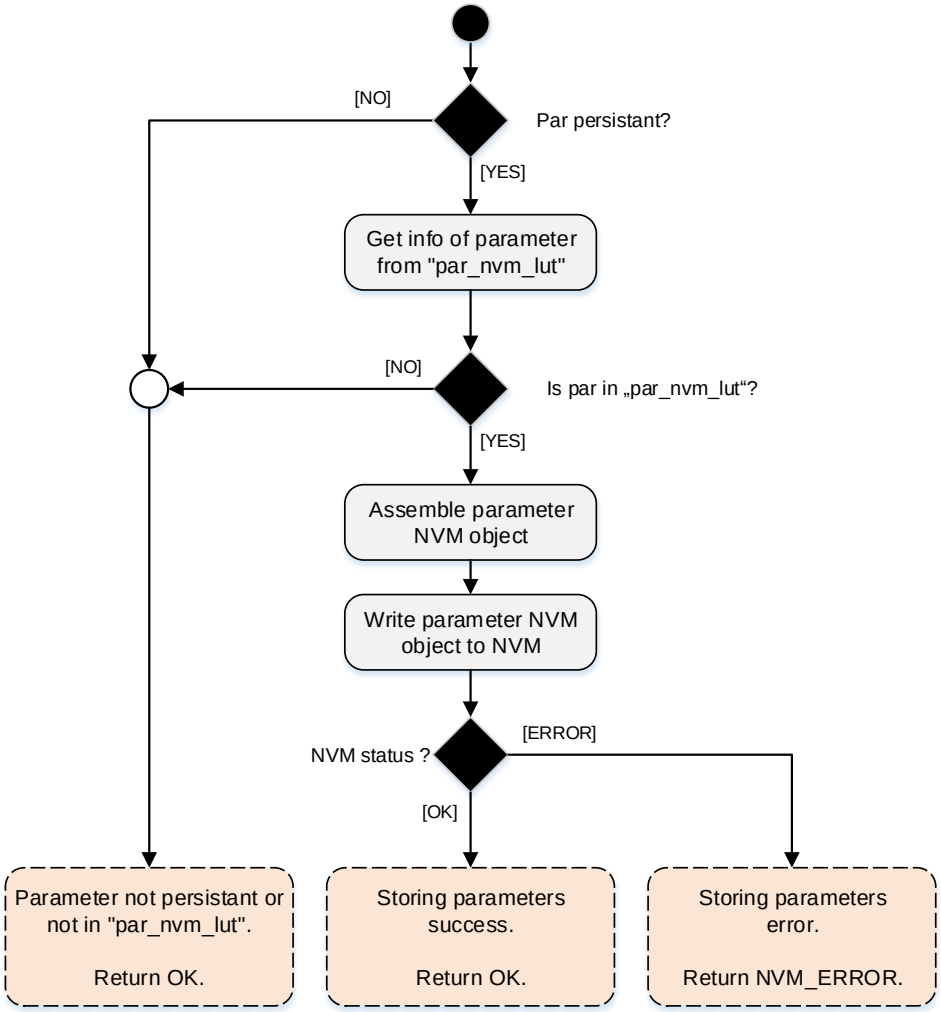


Figure 6 Storing single parameter to NVM