LwWDG

Tilen MAJERLE

Apr 12, 2024

CONTENTS

Welcome to the documentation for version v1.1.2.

LwWDG is lightweight watchdog library, primarily targeting operating systems, to watch multiple threads and reset system if one of them fails.

Download library Getting started Open Github Donate

CHAPTER

ONE

FEATURES

- Written in C (C11)
- Easy to use - very little platform dependency
- Written for operating systems in mind

CHAPTER

TWO

REQUIREMENTS

- C compiler
- Few *kB* of non-volatile memory

CHAPTER

THREE

CONTRIBUTE

Fresh contributions are always welcome. Simple instructions to proceed:

1. Fork Github repository
2. Respect C style & coding rules used by the library
3. Create a pull request to `develop` branch with new features or bug fixes

Alternatively you may:

1. Report a bug
2. Ask for a feature request

CHAPTER

FOUR

LICENSE

```
MIT License

Copyright (c) 2024 Tilen MAJERLE

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 Getting started

Getting started may be the most challenging part of every new library. This guide is describing how to start with the library quickly and effectively

5.1.1 Download library

Library is primarily hosted on Github.

You can get it by:

- Downloading latest release from releases area on Github
- Cloning `main` branch for latest stable version
- Cloning `develop` branch for latest development

Download from releases

All releases are available on Github releases area.

Clone from Github

First-time clone

This is used when you do not have yet local copy on your machine.

- Make sure `git` is installed.
- Open console and navigate to path in the system to clone repository to. Use command `cd your_path`
- Clone repository with one of available options below
  - Run `git clone --recurse-submodules https://github.com/MaJerle/lwwdg` command to clone entire repository, including submodules
  - Run `git clone --recurse-submodules --branch develop https://github.com/MaJerle/lwwdg` to clone *development* branch, including submodules
  - Run `git clone --recurse-submodules --branch main https://github.com/MaJerle/lwwdg` to clone *latest stable* branch, including submodules
- Navigate to `examples` directory and run favourite example

Update cloned to latest version

- Open console and navigate to path in the system where your repository is located. Use command `cd your_path`
- Run `git pull origin main` command to get latest changes on `main` branch
- Run `git pull origin develop` command to get latest changes on `develop` branch
- Run `git submodule update --init --remote` to update submodules to latest version

Note: This is preferred option to use when you want to evaluate library and run prepared examples. Repository consists of multiple submodules which can be automatically downloaded when cloning and pulling changes from root repository.

5.1.2 Add library to project

At this point it is assumed that you have successfully download library, either cloned it or from releases page. Next step is to add the library to the project, by means of source files to compiler inputs and header files in search path.

CMake is the main supported build system. Package comes with the `CMakeLists.txt` and `library.cmake` files, both located in the `lwwdg` directory:

- `CMakeLists.txt`: Is a wrapper and only includes `library.cmake` file. It is used if target application uses `add_subdirectory` and then uses `target_link_libraries` to include the library in the project
- `library.cmake`: It is a fully configured set of variables. User must use `include(path/to/library.cmake)` to include the library and must manually add files/includes to the final target

---

Tip: Open `library.cmake` file and manually analyze all the possible variables you can set for full functionality.

If you do not use the *CMake*, you can do the following:

- Copy `lwwdg` folder to your project, it contains library files
- Add `lwwdg/src/include` folder to *include path* of your toolchain. This is where *C/C++* compiler can find the files during compilation process. Usually using `-I` flag
- Add source files from `lwwdg/src/` folder to toolchain build. These files are built by *C/C++* compiler. CMake configuration comes with the library, allows users to include library in the project as **subdirectory** and **library**.
- Copy `lwwdg/src/include/lwwdg/lwwdg_opts_template.h` to project folder and rename it to `lwwdg_opts.h`
- Build the project

5.1.3 Configuration file

Configuration file is used to overwrite default settings defined for the essential use case. Library comes with template config file, which can be modified according to the application needs. and it should be copied (or simply renamed in-place) and named `lwwdg_opts.h`

Note: Default configuration template file location: `lwwdg/src/include/lwwdg/lwwdg_opts_template.h`. File must be renamed to `lwwdg_opts.h` first and then copied to the project directory where compiler include paths have

access to it by using `#include "lwwdg_opts.h"`.

Tip: If you are using *CMake* build system, define the variable `LWWDG_OPTS_FILE` before adding library's directory to the *CMake* project. Variable must contain the path to the user options file. If not provided and to avoid build error, one will be generated in the build directory.

Configuration options list is available available in the *Configuration* section. If any option is about to be modified, it should be done in configuration file

Listing 1: Template configuration file

```
/**
 * \file            lwwdg_opts_template.h
 * \brief           LwWDG configuration file
 */

/*
 * Copyright (c) 2024 Tilen MAJERLE
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without restriction,
 * including without limitation the rights to use, copy, modify, merge,
 * publish, distribute, sublicense, and/or sell copies of the Software,
 * and to permit persons to whom the Software is furnished to do so,
 * subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
 * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 *
 * This file is part of LWWDG - Lightweight watchdog for RTOS in embedded systems.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         v1.1.2
 */
#ifndef LWWDG_OPTS_HDR_H
#define LWWDG_OPTS_HDR_H

/* Rename this file to "lwwdg_opts.h" for your application */

/*
 * Open "include/lwwdg/lwwdg_opt.h" and
```

(continues on next page)

(continued from previous page)

```
 * copy & replace here settings you want to change values
 */

#endif /* LWWDG_OPTS_HDR_H */
```

Note: If you prefer to avoid using configuration file, application must define a global symbol `LWWDG_IGNORE_USER_OPTS`, visible across entire application. This can be achieved with `-D` compiler option.

5.2 User manual

LwWDG library is very simple and easy to use. LwWDG was designed to implement software watchdog functionality, primarily used in the operating systems.

Each task in the system defines its very own Watchdog structure, and is responsible to periodically call reload function, while one of the tasks, (it can be) called `master` task checks the processing function, and reloads hardware watchdog, if everything is fine.

When one of the software watchdogs isn't reloaded within maximum timeout window, main task is not supposed to reload hardware timer anymore. As a consequence, hardware watchdog will reset the system.

Note: This library is designed for operating system, where operations are splitted in the tasks. To ensure stable operation, each task should have a monitoring system.

Since microcontrollers typical utilize single independent watchdog, such solution must be implemented as a combination of hardware and software components.

5.2.1 Platform migration

Library requires atomicity in the processing function, and a milliseconds time source. These should be implemented as macros in the configuration file. Checkout the configuration window or follow to example mentioned below.

5.2.2 Example

A simple example to illustrate functionality.

Listing 2: WIN32 example code

```
#include "lwwdg/lwwdg.h"
#include "windows.h"
#include <stdio.h>
#include <stdlib.h>

HANDLE lwwdg_mutex;                         /* Mutex to simulate interrupt lock */
static LARGE_INTEGER freq, sys_start_time; /* Milliseconds time variable */

/**
 * \brief           Get current tick in ms from start of program
```

(continues on next page)

(continued from previous page)

```
 * \return          uint32_t: Tick in ms
 */
uint32_t
sys_get_tick(void) {
    LONGLONG ret;
    LARGE_INTEGER now;

    QueryPerformanceFrequency(&freq);
    QueryPerformanceCounter(&now);
    ret = now.QuadPart - sys_start_time.QuadPart;
    return (uint32_t)((ret * 1000) / freq.QuadPart);
}

/* Task 1 */
static void
task1(void* arg) {
    static lwwdg_wdg_t wdg;
    (void)arg;

    printf("%8u: Task 1 started...\r\n", (unsigned)sys_get_tick());
    lwwdg_add(&wdg, 3000);
    lwwdg_set_name(&wdg, "task_1_wdg");
    while (1) {
        /* Periodic reloads... */
        lwwdg_reload(&wdg);
    }
}

/* Task 1 */
static void
task2(void* arg) {
    static lwwdg_wdg_t wdg;
    (void)arg;

    printf("%8u: Task 2 started...\r\n", (unsigned)sys_get_tick());
    lwwdg_add(&wdg, 5000);
    lwwdg_set_name(&wdg, "task_2_wdg");
    while (1) {
        /* No reload in this task -> consider it failed to run properly */
        Sleep(1000);
    }
}

/**
 * \brief           Example main function
 */
void
example_win32(void) {
    DWORD id;

    QueryPerformanceFrequency(&freq);
    QueryPerformanceCounter(&sys_start_time);
```

(continues on next page)

(continued from previous page)

```

    /* Create lock for lwwdg */
    lwwdg_mutex = CreateMutex(NULL, 0, NULL);
    lwwdg_init(); /* Init library */

    /* Start other tasks */
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)task1, NULL, 0, &id);
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)task2, NULL, 0, &id);

    /* Main task... */
    while (1) {
        /* Check if all tasks are OK */
        if (lwwdg_process()) {
            printf("%8u: Refreshing hardware watchdog...\r\n", (unsigned)sys_get_tick());
            /* TODO: This is where you should reload hardware watchdog */
        } else {
            printf("%8u: At least one task is out of window -> HW watchdog should not be␣
↪reloaded anymore\r\n",
                   (unsigned)sys_get_tick());
            break;
        }
        Sleep(500); /* Make some sleep to offload messages in the WIN32 example */
    }
    printf("List of expired watchdogs\r\n");
    lwwdg_print_expired();
    printf("Example completed - a hardware should reset the system now...\r\n");
}
```

A corresponding options file, tested for WIN32.

Listing 3: WIN32 LwWDG options file

```
/**
 * \file            lwwdg_opts_template.h
 * \brief           LwWDG configuration file
 */

/*
 * Copyright (c) 2024 Tilen MAJERLE
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without restriction,
 * including without limitation the rights to use, copy, modify, merge,
 * publish, distribute, sublicense, and/or sell copies of the Software,
 * and to permit persons to whom the Software is furnished to do so,
 * subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

(continues on next page)

(continued from previous page)

```
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
 * AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 *
 * This file is part of LWWDG - Lightweight watchdog for RTOS in embedded systems.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         v1.1.2
 */
#ifndef LWWDG_HDR_OPTS_H
#define LWWDG_HDR_OPTS_H

#include <stdio.h>

/* Win32 port */
#include "windows.h"
extern uint32_t sys_get_tick(void); /* Milliseconds tick is available externally */
extern HANDLE lwwdg_mutex;          /* Mutex is defined and initialized externally */

#define LWWDG_CRITICAL_SECTION_DEFINE /* Nothing to do here... */
#define LWWDG_CRITICAL_SECTION_LOCK()                                                  \
    do {                                                                               \
        WaitForSingleObject(lwwdg_mutex, INFINITE);                                    \
    } while (0)
#define LWWDG_CRITICAL_SECTION_UNLOCK()     ReleaseMutex(lwwdg_mutex)
#define LWWDG_GET_TIME()                    sys_get_tick()

#define LWWDG_CFG_ENABLE_WDG_NAME           1
#define LWWDG_CFG_WDG_NAME_ERR_DEBUG(_wdg_) printf("Watchdog %s failed to reload in time!\r\n", (_wdg_))

#endif /* LWWDG_HDR_OPTS_H */
```

5.3 API reference

List of all the modules:

5.3.1 LwWDG

group **LWWDG**

Lightweight watchdog for RTOS in embedded systems.

Functions

uint8_t **lwwdg_init**(void)

Initialize watchdog module.

Returns
1 on success, 0 otherwise

uint8_t **lwwdg_add**(*lwwdg_wdg_t* *wdg, uint32_t timeout)

Add new watchdog timer instance to internal linked list.

Parameters

- **wdg** – Watchdog handle. Must not be local variable
- **timeout** – Max allowed timeout in milliseconds

Returns
1 on success, 0 otherwise

uint8_t **lwwdg_remove**(*lwwdg_wdg_t* *wdg)

Remove watchdog from the list.

This function is typically used if a task is killed by the scheduler. A user must manually call the function and can later clean wdg memory

Parameters
wdg – Watchdog handle to remove from list

Returns
1 if removed, 0 otherwise

uint8_t **lwwdg_reload**(*lwwdg_wdg_t* *wdg)

Reload thread watchdog.

Note: Reload will not be successful, if there was a timeout before. This will ensure that main thread won't reload hardware watchdog, resulting system to reset

Parameters
wdg – Watchdog handle to reload

Returns
1 on success, 0 otherwise

uint8_t **lwwdg_process**(void)

Process and check system timers.

Function will check all timers and will return OK, if all timers are within max timeout state

Returns

1 if hardware watchdog can be reset, 0 if at least one timer hasn't been reloaded within maximum timeout

void **lwwdg_set_name**(*lwwdg_wdg_t* *wdg, const char *name)

Set the watchdog name for debug reasons.

Note: Available only when *LWWDG_CFG_ENABLE_WDG_NAME* is enabled

Parameters

- **wdg** – Watchdog instance
- **name** – Pointer to the constant string for the name. String is not copied, rather only pointer is set

void **lwwdg_print_expired**(void)

Print all expired watchdogs.

Note: *LWWDG_CFG_ENABLE_WDG_NAME* and *LWWDG_CFG_WDG_NAME_ERR_DEBUG* must be enabled and implemented

struct **lwwdg_wdg_t**

#include <*lwwdg.h*> Watchdog structure.

Public Members

struct lwwdg_wdg ***next**

Next entry on a list

uint32_t **timeout**

Timeout in milliseconds

uint32_t **last_reload_time**

Last reload time in milliseconds

const char ***name**

Pointer to constant string indicating watchdog name

5.3.2 Configuration

This is the default configuration of the middleware. When any of the settings shall be modified, it shall be done in dedicated application config `lwwdg_opts.h` file.

Note: Check *Getting started* to create configuration file.

group **LWWDG_OPT**

Default configuration setup.

Defines

LWWDG_CFG_ENABLE_WDG_NAME

Enables or disables field in wdg structure to contain watchdog name.

This can be useful for debugging purposes

LWWDG_CFG_WDG_NAME_ERR_DEBUG(_wdg_name_)

Macro called if *LWWDG_CFG_ENABLE_WDG_NAME* is enabled and if watchdog error occurs.

It can be overwritten by the application to print watchdog name.

Parameters

- **wdg_name** – **[in]** Watchdog name as defined by *lwwdg_set_name* function

LWWDG_GET_TIME()

Get system time in milliseconds.

It is required to keep reload time

LWWDG_CRITICAL_SECTION_DEFINE

Define the critical section.

Used at the beinning of the function, to define potential local variable to keep status of critical section (if already locked)

Note: Default implementation is for *Cortex-M*

LWWDG_CRITICAL_SECTION_LOCK()

Lock the critical section.

Critical section should prevent other tasks or interrupt, to access to the same core.

Easiest is to simply disable the interrupt, since task is normally pretty quick.

Note: Default implementation is for *Cortex-M*

LWWDG_CRITICAL_SECTION_UNLOCK()

Unlock the critical section.

Note: Default implementation is for *Cortex-M*

5.4 Changelog

```
# Changelog

## Develop

## v1.1.2

- Fix race condition in `lwwdg_process` function

## v1.1.1

- Fix wrong time check

## v1.1.0

- Print error message only on one trial
- Add print option to list all expired watchdogs

## v1.0.0

- Added option to remove wdg from the list
- Added option to set watchdog name and print its name on error

## v0.0.1

- First version
****
```

5.5 Authors

List of authors and contributors to the library

```
Tilen Majerle <tilen.majerle@gmail.com>
Tilen Majerle <tilen@majerle.eu>
```

INDEX

L