LwEVT

Tilen MAJERLE

Mar 01, 2024

CONTENTS

Welcome to the documentation for version .

LwEVT is a simple event manager for embedded system. Its main purpose is to be able to send various events in the application from various modules. Application defines custom types with optional data structure to be sent to various application listeners.

Download library Getting started Open Github Donate

CHAPTER

ONE

FEATURES

- Written in C (C11)
- Platform independent, no architecture specific code
- Flexible for application defined event types and associated data
- Easy to use and maintain
- User friendly MIT license

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 primarly 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/lwevt` command to clone entire repository, including submodules
  - Run `git clone --recurse-submodules --branch develop https://github.com/MaJerle/lwevt` to clone *development* branch, including submodules
  - Run `git clone --recurse-submodules --branch main https://github.com/MaJerle/lwevt` 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 `lwevt` 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 `lwevt` folder to your project, it contains library files
- Add `lwevt/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 `lwevt/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 `lwevt/src/include/lwevt/lwevt_opts_template.h` to project folder and rename it to `lwevt_opts.h`
- Copy `lwevt/src/include/lwevt/lwevt_types_template.h` to project folder and rename it to `lwevt_types.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 `lwevt_opts.h`

Note: Default configuration template file location: `lwevt/src/include/lwevt/lwevt_opts_template.h`. File must be renamed to `lwevt_opts.h` first and then copied to the project directory where compiler include paths have access to it by using `#include "lwevt_opts.h"`.

Tip: If you are using *CMake* build system, define the variable `LWEVT_OPTS_DIR` before adding library's directory to the *CMake* project. Variable must set the output directory path. CMake will copy the template file there, and name it as required.

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            lwevt_opts_template.h
 * \brief           LwEVT 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 LwEVT - Lightweight event manager.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         v1.0.0
```

(continues on next page)

(continued from previous page)

```
 */
#ifndef LWEVT_OPTS_HDR_H
#define LWEVT_OPTS_HDR_H

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

/*
 * Open "include/lwevt/lwevt_opt.h" and
 * copy & replace here settings you want to change values
 */

#endif /* LWEVT_OPTS_HDR_H */
```

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

5.1.4 Types file

Every project needs definition of various event types. `lwevt_types.h` file defines list of events and optional data structure next to the event type

Tip: If you are using *CMake* build system, define the variable `LWEVT_TYPES_DIR` before adding library's directory to the *CMake* project. Variable must set the output directory path. CMake will copy the template file there, and name it as required.

Listing 2: Template types file

```
/**
 * \file            lwevt_types_template.h
 * \brief           LwEVT application types
 */

/*
 * 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
```

(continues on next page)

(continued from previous page)

```
 * 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 LwEVT - Lightweight event manager.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         v1.0.0
 */

/* Rename this file to "lwevt_opts.h" for your application and do not use include guard␣
↪macros */

/*
 * Define your different event types with 2 macros
 *
 * Basic definition - event type w/o possibility for data
 *
 * LWEVT_TYPE_BASIC(APP_EVT_BASIC_1)
 *
 * Extended definition - event type w/ possibility for data
 *
 * LWEVT_TYPE_BASIC(APP_EVT_EXT_W_DATA_1, struct { uint32_t my_par1; uint32_t my_par2; }␣
↪ext_w_data;)
 */
```

5.2 User manual

LwEVT is simple event manager library, its sole purpose is to be able to isolate various application modules and communicate in-between through event manager library.

Idea behind the library is to allow every module to subscribe to events or publish them upon various events or actions.

5.2.1 Define application events

Note: At this point we assume you have properly setup the library from *Getting started* guideline.

Every application is different and wants to use different types of events and its related data to be part of it. It is important to define your own `lwevt_types.h` application file with your events.

There are 2 methods to define the event type:

- Basic event type with *LWEVT_TYPE_BASIC*
- Extended event type with *LWEVT_TYPE_EXT* that also includes data structure to be part of event

In your `lwevt_types.h` declare list of your application defines, as per example below. This file has to be generated by user and is application specific.

Listing 3: Definition of event types

```
/* Define basic types */
LWEVT_TYPE_BASIC(LWEVT_TYPE_MY_BASIC_1)
LWEVT_TYPE_BASIC(LWEVT_TYPE_MY_BASIC_2)
LWEVT_TYPE_BASIC(LWEVT_TYPE_MY_BASIC_3)

/* Define extended types */
LWEVT_TYPE_EXT(LWEVT_TYPE_MY_EXT_1, struct { int par1; int par2; } ext1)
LWEVT_TYPE_EXT(LWEVT_TYPE_MY_EXT_2, struct { int par3; int par4; } ext2)
LWEVT_TYPE_EXT(LWEVT_TYPE_MY_EXT_3, struct { int par1; int par2; } ext3)
```

You are now ready to send events to all subscribed modules

5.2.2 Produce events

Producing (sending) the event is simple and requires to fill event type and potential data (only part of extended type).

Tip: Producer does not need to be subscribed to any events.

Listing 4: Produce event to all subscribers

```
#include "lwevt/lwevt.h"

void
produce() {
    lwevt_t* evt;

    /* Initialize event management system = must be called only once in the application */
    lwevt_init();

    /*
     * Get event handle, set event data and dispatch event
     *
     * Send basic event - without any data
     */
    evt = lwevt_get_handle();
    lwevt_dispatch(LWEVT_TYPE_MY_BASIC_1);

    /*
     * Get event handle, set event data and dispatch event
     *
     * Send extended event - with data
     */
    evt = lwevt_get_handle();
    evt->msg.ext1.par1 = 10;    /* Some value */
    evt->msg.ext1.par2 = 12;    /* Some value */
    lwevt_dispatch(LWEVT_TYPE_MY_EXT_1);
    return 0;
}
```

5.2.3 Subscribe to events

Subscribing to events involves calling *lwevt_register()* function with callback function as parameter. All events are processed in the callback function from now on.

Listing 5: Receive events from the application

```c
#include "lwevt/lwevt.h"

/* Define event functions */
static void
prv_evt_fn(lwevt_t* e) {
    printf("Event type: %u\r\n", (unsigned)e->type);
    switch (e->type) {
        case LWEVT_TYPE_MY_BASIC_1: {
            printf("Basic event type - no data\r\n");
            break;
        }
        case LWEVT_TYPE_MY_EXT_1: {
            printf("Extended event type - with data: par1: %d, par2: %d\r\n",
                (int)e->msg.ext1.par1, (int)e->msg.ext1.par2);
            break;
        }
        default:
            break;
    }
}

int
main() {
    lwevt_t* evt;

    /* Initialize event management system = must be called only once in the application */
    lwevt_init();

    /* Set 2 custom functions for event */
    lwevt_register(prv_evt_fn);

    /* Do nothing - events are handled in callback function */

    return 0;
}
```

5.2.4 Global and local event handle

To optimize memory consumption, main event handle is defined as static global variable in the LwEVT module. It is accessible through *lwevt_get_handle()* function and allows use of default *lwevt_dispatch()* function call to send event to the application.

Note: *lwevt_get_handle()* and *lwevt_dispatch()* are available only when *LWEVT_CFG_ENABLE_DEFAULT_HANDLE* is enabled

In multi-threading environment, application must ensure thread safety between *get handle* and *dispatch* calls. To avoid use of semaphore or mutexes, alternative is to always define local *lwevt_t* based variable and dispatch event using *lwevt_dispatch_ex()* function

Listing 6: Produce events with global or local event handle

```
#include "lwevt/lwevt.h"

void
produce() {
    lwevt_t* evt_global;
    lwevt_t evt_local;

    /*
     * Send event using global handle
     * Thread safety has to be ensured in multi-threading environments
     */
    evt_global = lwevt_get_handle();
    evt_global->msg.ext1.par1 = 10; /* Some value */
    evt_global->msg.ext1.par2 = 12; /* Some value */
    lwevt_dispatch(LWEVT_TYPE_MY_EXT_1);

    /*
     * Send event using local handle
     * No need to ensure thread safety
     */
    evt_local.msg.ext1.par1 = 10;   /* Some value */
    evt_local.msg.ext1.par2 = 12;   /* Some value */
    lwevt_dispatch_ex(&evt_local, LWEVT_TYPE_MY_EXT_1);
    return 0;
}
```

5.3 API reference

List of all the modules:

5.3.1 LwEVT

group **LWEVT**

Lightweight event system.

Defines

LWEVT_TYPE_BASIC(name)

Defines basic event type with respective name, only.

Basic events do not have possibility to send data to application

```
#define LWEVT_TYPE_BASIC(MY_EVENT_NAME)
```

LWEVT_TYPE_EXT(name, data)

Defines extended event type with possibily to send data.

Data C-type has to be declared for proper use case

```
#define LWEVT_TYPE_EXT(MY_EVENT_NAME, struct {int par1; int par2;}
my_event_name;)
```

Typedefs

typedef void (***lwevt_fn**)(*lwevt_t* *evt)

Event callback function prototype.

Enums

enum **lwevt_type_t**

Event type.

Values:

enumerator **LWEVT_TYPE_LAST**

Last element on the list

Functions

void **lwevt_init**(void)

Initialize LwEVT module.

uint8_t **lwevt_register**(*lwevt_fn* evt_fn)

Register new event listener callback to event manager.

Parameters
evt_fn – Function to add to list of listeners

Returns
1 if added, 0 otherwise

lwevt_t ***lwevt_get_handle**(void)

Get default handle object for dispatch purpose.

Note: Available only when *LWEVT_CFG_ENABLE_DEFAULT_HANDLE* is enabled

Returns
Pointer to default event handle

uint8_t **lwevt_dispatch**(*lwevt_type_t* type)

Dispatch event to all registered functions.

Note: It uses default event handle as parameter

Note: Available only when *LWEVT_CFG_ENABLE_DEFAULT_HANDLE* is enabled

Parameters
type – **[in]** Event type to dispatch

Returns
1 if dispatched, 0 otherwise

uint8_t **lwevt_dispatch_ex**(*lwevt_t* *evt_handle, *lwevt_type_t* type)

Dispatch event to all registered functions using custom event handle object.

Parameters

- **evt_handle** – Event handle used as parameter to listeners
- **type** – Event type to dispatch

Returns
1 if dispatched, 0 otherwise

struct **lwevt_t**

#include <lwevt.h> Main event structure.

Public Members

lwevt_type_t **type**

Event type

const unsigned int **dummy**

Dummy element if no others are used by user

union *lwevt_t*::[anonymous] **msg**

Message union for extended event types

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 `lwevt_opts.h` file.

Note: Check *Getting started* for guidelines on how to create and use configuration file.

group **LWEVT_OPT**

Default configuration setup.

Defines

LWEVT_MEMSET(dst, val, len)

Memory set function.

Note: Function footprint is the same as memset

LWEVT_CFG_MAX_EVT_LISTENERS

Maximum number of event listeners that can receive info on event dispatch.

It defines size of array for function pointers

LWEVT_CFG_ENABLE_DEFAULT_HANDLE

Enables `1` or disables `0` creation of default event handle.

When enabled, user can use *lwevt_get_handle* and *lwevt_dispatch* functions to operate on defualt handle.

When disabled, user must create event handle before every dispatch, and is only able to use *lwevt_dispatch_ex* function

5.4 Changelog

```
# Changelog

## Develop

## v1.0.0

- Add `.clang-format`
- First official release

## v0.1.0

- Initial release
```

5.5 Authors

List of authors and contributors to the library

```
Tilen Majerle <tilen.majerle@gmail.com>
Tilen Majerle <tilen@majerle.eu>
Jabin Kong <1059978534@qq.com>
```

INDEX

L