LwDTC

Tilen MAJERLE

Mar 19, 2024

CONTENTS

Welcome to the documentation for version latest-develop.

LwDTC is lightweight, platform independent library for date, time and cron utility management.

Main motivation comes from the necessity of simple cron for my own personal home automation project, with 1 second granularity.

Download library Getting started Open Github Donate

CHAPTER

ONE

FEATURES

- Written in C (C11)
- Platform independent, easy to use
- Support for date, time and cron utilities
- Ultra-lightweight cron library for embedded systems
- Cron supports numbers only, no string dates/months, quicker parsing
- Support for *time.h struct tm* data structure for time operations
- Date and time range support with CRON syntax
- 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/lwdtc` command to clone entire repository, including submodules
  - Run `git clone --recurse-submodules --branch develop https://github.com/MaJerle/lwdtc` to clone *development* branch, including submodules
  - Run `git clone --recurse-submodules --branch main https://github.com/MaJerle/lwdtc` 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 `lwdtc` 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 `lwdtc` folder to your project, it contains library files
- Add `lwdtc/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 `lwdtc/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 `lwdtc/src/include/lwdtc/lwdtc_opts_template.h` to project folder and rename it to `lwdtc_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 `lwdtc_opts.h`

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

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

Tip: If you are using *CMake* build system, define the variable `LWDTC_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            lwdtc_opts_template.h
 * \brief           LwDTC 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 LwDTC - Lightweight Date, Time & Cron library.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         $_version_$
 */
#ifndef LWDTC_OPTS_HDR_H
#define LWDTC_OPTS_HDR_H

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

/*
 * Open "include/lwdtc/lwdtc_opt.h" and
```

(continues on next page)

(continued from previous page)

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

#endif /* LWDTC_OPTS_HDR_H */
```

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

5.2 User manual

5.2.1 CRON

LwDTC provides ultra-lightweight support for simple CRON implementation. This page tends to provide quick information to be able to quickly move forward.

In the current revision library doesn't support true scheduler, instead it supports CRON string parser, and dedicated function to check if CRON is valid against compared time.

Tip: User can implement its own CRON loop, that reads current time and checks if parsed CRON is valid for specific time. Use *lwdtc_cron_is_valid_for_time()* to compare

Supported characters

- Numbers between minimum and maximum value for each of date and time field
- Support for `seconds`, `minutes`, `hours`, `day-in-month`, `month`, `day-in-week` and `year`,
- `-` is used to define value range, with min and max boundaries, `min-max` or `max-min`
- `,` is used to specify multiple fixed values
- `/` is used to define step between min and max values
- `*` is used to represent *any* value

Note: Comparing to standard linux CRON, where fixed date in month and week day are bitwise-ORed, meaning cron will fire on day in month match or on week day match, LwDTC does it more simple. Date&Time is valid only if both parameters are a match at the same time. In practice, setting cron to fire on `month-day = 15` and `week-day = 6`, will trigger it only on **15th** in a month which is also **Saturday** at the same time.

CRON string format

To define valid CRON string, a string with 7 parameters, separated by space, must be provided:

`seconds minutes hours day-in-month month day-in-week year`

Each of them has to be present to consider CRON as valid input.

CRON examples

This section provides list of some CRON examples in its default configuration.

CRON string	Description
`* * * * * * *`	CRON is valid all the time, will fire every second
`0 * * * * * *`	CRON is valid at the beginning of each minute
`* * * * * 2 *`	CRON is valid every Tuesday all day long
`0 0 13-15 * * 2-4 *`	CRON is valid every beginning of the minute between hours 13-15 afternoon, between Tuesday and Thursday
`*/5 * * * * * *`	CRON is valid every `5` seconds starting at `0`
`*/5 */5 * * * * *`	CRON is valid every `5` seconds each `5` minutes, from `00:00` to `55:55`
`0 0 0 * * 5 *`	Every Friday at midnight
`0 0 */2 * * * *`	Every `2` hours at beginning of the hour
`* * */2 * * * *`	Every second of every minute every `2` hours (`0, 2, 4, .., 22`)
`0 0 0 * * 1-5 *`	At midnight, `00:00` every week between Monday and Friday
`15 23 */6 * * * *`	Every `6` hours at (min:sec) `23:15` (`00:23:15, 06:23:15, 12:23:15`, ...)
`0 0 0 1 * * *`	At `00:00:00` beginning of the month
`0 0 0 1 */3 * *`	Every beginning of the quarter at 00:00:00
`10 15 20 * 8 6 *`	At `20:15:20` every Saturday in August
`10 15 20 8 * 6 *`	At `20:15:20` every Saturday that is also `8th` day in month (both must match, day Saturday and date `8th`)
`30-45 * * * * * *`	Every second between `30` and `45`
`30-45/3 * * * * * *`	Every `3rd` second in every minute, when seconds are between `30` and `45`
`0 23/1 * * * * *`	Every beginning of a minute when minute is between `23` and `59`
`50-10 * * * * * *`	Every second when seconds are from `50-59` and `00-10` (overflow mode)

Listing 2: Basic CRON example with parser

```
#include "windows.h"
#include <time.h>
#include <stdio.h>
#include "lwdtc/lwdtc.h"

int
cron_basic(void) {
    /* Define context for CRON, used to parse data to */
```

(continues on next page)

(continued from previous page)

```
    lwdtc_cron_ctx_t cron_ctx = {0};
    struct tm* timeinfo;
    time_t rawtime, rawtime_old = 0;

    /* Execute cron to be valid every 2 seconds */
    if (lwdtc_cron_parse(&cron_ctx, "*/2 * * * * * *") != lwdtcOK) {
        printf("Error parsing CRON...\r\n");
        while (1) {}
    }
    while (1) {
        /* Get current time and react on changes only */
        time(&rawtime);

        /* Check if new time has changed versus last read */
        if (rawtime != rawtime_old) {
            rawtime_old = rawtime;
            timeinfo = localtime(&rawtime);

            /* Print time to user */
            printf("Time: %02d.%02d.%04d %02d:%02d:%02d\r\n",
                (int)timeinfo->tm_mday, (int)timeinfo->tm_mon, (int)timeinfo->tm_year + 1900,
                (int)timeinfo->tm_hour, (int)timeinfo->tm_min, (int)timeinfo->tm_sec
            );

            /* Check if CRON should execute */
            if (lwdtc_cron_is_valid_for_time(timeinfo, &cron_ctx) == lwdtcOK) {
                printf("Executing CRON task\r\n");
            }
        }

        /* This is sleep from windows.h lib */
        Sleep(100);
    }
    return 0;
}
```

5.2.2 CRON basic schedule

The idea behind cron is to schedule tasks at specific interval or period of time.

Library does not provide generic scheduler that would do that automatically for you, instead user should manually implement custom scheduler to periodically check and schedule should cron execute or not.

Tip: A check should be performed at least at minimum cron granularity, that being 1 `second` in the current revision.

Basic example is very simple and does the following:

- Parses cron defined by user with **:cpp:`lwdtc_cron_parse`** function
- Reads system time periodically (example was tested under *Windows* environment)

- It checks if cron is valid for execution each time new time changes versus previous check
- It executes task

Listing 3: Basic CRON example with parser

```c
#include "windows.h"
#include <time.h>
#include <stdio.h>
#include "lwdtc/lwdtc.h"

int
cron_basic(void) {
    /* Define context for CRON, used to parse data to */
    lwdtc_cron_ctx_t cron_ctx = {0};
    struct tm* timeinfo;
    time_t rawtime, rawtime_old = 0;

    /* Execute cron to be valid every 2 seconds */
    if (lwdtc_cron_parse(&cron_ctx, "*/2 * * * * * *") != lwdtcOK) {
        printf("Error parsing CRON...\r\n");
        while (1) {}
    }
    while (1) {
        /* Get current time and react on changes only */
        time(&rawtime);

        /* Check if new time has changed versus last read */
        if (rawtime != rawtime_old) {
            rawtime_old = rawtime;
            timeinfo = localtime(&rawtime);

            /* Print time to user */
            printf("Time: %02d.%02d.%04d %02d:%02d:%02d\r\n",
                (int)timeinfo->tm_mday, (int)timeinfo->tm_mon, (int)timeinfo->tm_year + 1900,
                (int)timeinfo->tm_hour, (int)timeinfo->tm_min, (int)timeinfo->tm_sec
            );

            /* Check if CRON should execute */
            if (lwdtc_cron_is_valid_for_time(timeinfo, &cron_ctx) == lwdtcOK) {
                printf("Executing CRON task\r\n");
            }
        }

        /* This is sleep from windows.h lib */
        Sleep(100);
    }
    return 0;
}
```

5.2.3 CRON multi schedule

Consider a task that has to execute at different times or periods of time, for example:

- *Each Friday at exactly midnight* `0 0 0 * * 5 *` and
- *Each second every Tuesday* `* * * * * 2 *`

With CRON syntax, it is not possible to describe this with one string, as it does not allow multi range option. Solution is to rather use *multiple* definitions and then let scheduler to check all of them until at least one is a match.

Tip: LwDTC comes with `*_multi` functions allowing you to check several CRON contextes with single function call.

An example shows simple demo how to implement a task scheduler which executes at different CRON context-es. It implements calls to `_multi` functions for simplification

Listing 4: CRON execution at multiple ranges

```
#include "windows.h"
#include <time.h>
#include <stdio.h>
#include "lwdtc/lwdtc.h"

/* Define all cron strings to execute one task */
static const char* cron_strings[] = {
    "* * * * * 2 *",            /* Task should run every second every Tuesday */
    "0 0 0 * * 5 *",            /* Task should run every Friday at midnight */
};

/* Set array of context objects */
static lwdtc_cron_ctx_t cron_ctxs[LWDTC_ARRAYSIZE(cron_strings)];

int
cron_multi(void) {
    /* Define context for CRON, used to parse data to */
    struct tm* timeinfo;
    time_t rawtime, rawtime_old = 0;
    size_t fail_index;

    /* Parse all cron strings */
    if (lwdtc_cron_parse_multi(cron_ctxs, cron_strings, LWDTC_ARRAYSIZE(cron_ctxs), &
↪fail_index) != lwdtcOK) {
        printf("Failed to parse cron at index %d\r\n", (int)fail_index);
        return 0;
    }
    printf("CRONs parsed and ready to go\r\n");

    while (1) {
        /* Get current time and react on changes only */
        time(&rawtime);

        /* Check if new time has changed versus last read */
        if (rawtime != rawtime_old) {
```

(continues on next page)

(continued from previous page)

```
            rawtime_old = rawtime;
            timeinfo = localtime(&rawtime);

            /*
             * Check if CRON should execute for all possible cron objects
             *
             * At least one task should be a pass to execute the task
             */
            if (lwdtc_cron_is_valid_for_time_multi_or(timeinfo, cron_ctxs, LWDTC_
↪ARRAYSIZE(cron_ctxs)) == lwdtcOK) {
                printf("Executing CRON task\r\n");
            }
        }

        /* This is sleep from windows.h lib */
        Sleep(100);
    }
    return 0;
}
```

5.2.4 CRON date&time range

Motivation for LwDTC library comes with my home automation project, where I need a simple way to define range of date or time, from and to, in very flexible and field-agnostic way. It should be possible to just define from and to seconds within a minute, or from 2 different minutes and different seconds.

CRON concept doesn't provide such functionality. To overcome such problem, a simple solution with multiple cron objects can be implemented. Consider a task, that needs to execute each beginning of a minute, between `Monday starting at 07:00:00` and `Friday ending at 19:30:00`.

For CRON-like compatible syntax, defined range needs a split to:

- *Monday*: CRON is active every beginning of a minute from `07` to `23` hours. This can be described as: `0 * 7/1 * * 1 *`
- *Tue,Wed,Thu*: CRON is active at beginning of each minute for all 3 days: `0 * * * * 2-4 *`
- *Friday*:
  - CRON is active between `0` and `19` hours, at beginning of each minute: `0 * 7-19 * * 5 *`
  - CRON is also active at beginning of each minute, when minutes are between `0` and `30` and when hour is `19`: `0 0-30 19 * * 5 *`

This gives us in total `4` different cron objects, for which:

- We need to parse all of them
- To check if particular time is within range, an *OR* operation between all ranges is performed

Listing 5: CRON date&time range descriptor

```
#include "windows.h"
#include <time.h>
#include <stdio.h>
#include "lwdtc/lwdtc.h"
```

(continues on next page)

(continued from previous page)

```c

/*
 * This is example for docs user manual
 *
 * Defines time range:
 * - Starts at Monday at 07:00 morning
 * - Ends on Friday at 19:30 evening
 */
static const char* cron_strings[] = {
    "0 * 7/1 * * 1 *",
    "0 * * * * 2-4 *",
    "0 * 7-19 * * 5 *",
    "0 0-30 19 * * 5 *"
};

/* Define context array for all CRON strings */
static lwdtc_cron_ctx_t cron_ctx[LWDTC_ARRAYSIZE(cron_strings)] = {0};

int
cron_dt_range(void) {
    /* Define context for CRON, used to parse data to */
    struct tm* timeinfo;
    time_t rawtime, rawtime_old = 0;

    /* Parse all CRON strings */
    for (size_t i = 0; i < LWDTC_ARRAYSIZE(cron_strings); ++i) {
        if (lwdtc_cron_parse(&cron_ctx[i], cron_strings[i]) != lwdtcOK) {
            printf("Could not parse CRON: %s\r\n", cron_strings[i]);
            while (1) {}
        }
    }

    while (1) {
        /* Get current time and react on changes only */
        time(&rawtime);

        /* Check if new time has changed versus last read */
        if (rawtime != rawtime_old) {
            rawtime_old = rawtime;
            timeinfo = localtime(&rawtime);

            /* Print time to user */
            printf("Time: %02d.%02d.%04d %02d:%02d:%02d\r\n",
                (int)timeinfo->tm_mday, (int)timeinfo->tm_mon, (int)timeinfo->tm_year + 1900,
                (int)timeinfo->tm_hour, (int)timeinfo->tm_min, (int)timeinfo->tm_sec
            );

            /* Check if current time fits inside CRON-defined time range */
            if (lwdtc_cron_is_valid_for_time_multi_or(timeinfo, cron_ctx, LWDTC_ARRAYSIZE(cron_ctx)) == lwdtcOK) {
                printf("Time is within CRON range\r\n");
```

(continues on next page)

(continued from previous page)

```
            } else {
                printf("Time is NOT within CRON range\r\n");
            }
        }

        /* This is sleep from windows.h lib */
        Sleep(100);
    }
    return 0;
}
```

5.3 API reference

List of all the modules:

5.3.1 LwDTC

group **LWDTC**

Lightweight Date, Time & Cron utility.

Defines

LWDTC_SEC_MIN

Minimum value for seconds field

LWDTC_SEC_MAX

Maximum value for seconds field

LWDTC_MIN_MIN

Minimum value for minutes field

LWDTC_MIN_MAX

Maximum value for minutes field

LWDTC_HOUR_MIN

Minimum value for hours field

LWDTC_HOUR_MAX

Maximum value for hours field

LWDTC_MDAY_MIN

Minimum value for day in month field

LWDTC_MDAY_MAX

Maximum value for day in month field

LWDTC_MON_MIN

Minimum value for month field

LWDTC_MON_MAX

Maximum value for month field

LWDTC_WDAY_MIN

Minimum value for week day field (min = Sunday, max = Saturday)

LWDTC_WDAY_MAX

Maximum value for week day field (min = Sunday, max = Saturday)

LWDTC_YEAR_MIN

Minimum value for year field

LWDTC_YEAR_MAX

Maximum value for year field

LWDTC_ARRAYSIZE(x)

Calculate size of statically allocated array.

Parameters

- **x** – **[in]** Array

Returns

Number of elements

Enums

enum **lwdtcr_t**

Result enumeration.

Values:

enumerator **lwdtcOK** = 0x00

Everything is OK

enumerator **lwdtcERR**

Generic error

enumerator **lwdtcERRPAR**

Invalid parameter passed to a function

enumerator **lwdtcERRTOKEN**

Token value is not valid

Functions

lwdtcr_t **lwdtc_cron_parse_with_len**(*lwdtc_cron_ctx_t* *ctx, const char *cron_str, size_t cron_str_len)

Parse string with linux crontab-like syntax, optionally enriched according to configured settings.

Parameters

- **ctx** – **[in]** Cron context variable used for storing parsed result
- **cron_str** – **[in]** Input cron string to parse data, using valid cron format recognized by the lib
- **cron_str_len** – **[in]** Length of input cron string, not counting potential `NULL` termination character

Returns

lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_parse**(*lwdtc_cron_ctx_t* *ctx, const char *cron_str)

Parse string with linux crontab-like syntax, optionally enriched according to configured settings.

Parameters

- **ctx** – **[in]** Cron context variable used for storing parsed result
- **cron_str** – **[in]** `NULL` terminated cron string, using valid cron format recognized by the lib

Returns

lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_parse_multi**(*lwdtc_cron_ctx_t* *cron_ctx, const char **cron_strs, size_t ctx_len, size_t *fail_index)

Parse multiple CRON strins at the same time. It returns immediately on first failed CRON.

Parameters

- **cron_ctx** – **[in]** Cron context variable used for storing parsed result
- **cron_strs** – **[in]** Pointer to array of string pointers with cron strings
- **ctx_len** – **[in]** Number of elements to process
- **fail_index** – **[out]** Optional pointer to output variable to store array index of failed CRON. Used only if function doesn't return *lwdtcOK*, otherwise pointer doesn't get modified

Returns

lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_is_valid_for_time**(const struct tm *tm_time, const *lwdtc_cron_ctx_t* *cron_ctx)

Check if cron is active at specific moment of time, provided as parameter.

Parameters

- **tm_time** – **[in]** Current time to check if cron works for it. Function assumes values in the structure are within valid boundaries and does not perform additional check
- **cron_ctx** – **[in]** Cron context object with valid structure

Returns

lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_is_valid_for_time_multi_or**(const struct tm *tm_time, const *lwdtc_cron_ctx_t* *cron_ctx, size_t ctx_len)

Check if current time fits to at least one of provided context arrays (OR operation)

Parameters

- **tm_time** – **[in]** Current time to check if cron works for it. Function assumes values in the structure are within valid boundaries and does not perform additional check
- **cron_ctx** – **[in]** Pointer to array of cron ctx objects
- **ctx_len** – **[in]** Number of context array length

Returns
lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_is_valid_for_time_multi_and**(const struct tm *tm_time, const *lwdtc_cron_ctx_t* *cron_ctx, size_t ctx_len)

Check if current time fits to all provided cron context arrays (AND operation)

Parameters

- **tm_time** – **[in]** Current time to check if cron works for it. Function assumes values in the structure are within valid boundaries and does not perform additional check
- **cron_ctx** – **[in]** Pointer to array of cron ctx objects
- **ctx_len** – **[in]** Number of context array length

Returns
lwdtcOK on success, member of *lwdtcr_t* otherwise

lwdtcr_t **lwdtc_cron_next**(const *lwdtc_cron_ctx_t* *cron_ctx, time_t curr_time, time_t *new_time)

Get next time of fire for specific cron object.

This is a dirty implementation and could be improved in the future. For now, we start with one second after current time, and do the roll over all values until we have a match.

Parameters

- **cron_ctx** – CRON context object
- **curr_time** – Current time, used as reference to get new time
- **new_time** – **[out]** Pointer to new time value

Returns
lwdtcOK on success, member of *lwdtcr_t* otherwise

struct **lwdtc_cron_ctx_t**

#include <lwdtc.h> Cron context variable with parsed information.

It is a bit-field of ones and zeros, indicating a match (or not) for date-time comparison to determine if needs to run (or not) a task

Public Members

uint32_t **flags**

List of all sort of flags for internal use

uint8_t **sec**[8]

Seconds field. Must support bits from 0 to 59

uint8_t **min**[8]

Minutes field. Must support bits from 0 to 59

uint8_t **hour**[3]

Hours field. Must support bits from 0 to 23

uint8_t **mday**[4]

Day number in a month. Must support bits from 0 to 30

uint8_t **mon**[2]

Month field. Must support bits from 0 to 11

uint8_t **wday**[1]

Week day. Must support bits from 0 (Sunday) to 6 (Saturday)

uint8_t **year**[13]

Year from 0 - 100, indicating 2000 - 2100. Must support bits 0 to 100

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 `lwdtc_opts.h` file.

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

group **LWDTC_OPT**

Default configuration setup.

Defines

LWDTC_MEMSET(dst, val, len)

Memory set function.

Note: Function footprint is the same as memset

LWDTC_CFG_GET_LOCALTIME(_struct_tm_ptr_, _const_time_t_ptr_)

Get the local time (struct tm) from the time_t pointer type.

Default implementation uses localtime but user may use gmtime or even create its own implementation, depending on the target system and overall wishes.

Parameters

- **struct_tm_ptr** – **[in]** Pointer variable to `struct tm` type. Variable is a pointer type and does not store actual time data.
- **const_time_t_ptr** – **[in]** Pointer to the `time_t` variable to get time from

5.4 Changelog

```
# Changelog

## Develop

- Rework parameters to avoid ARM GCC warnings for uninitialized var
- Add support to define max-min option - with overflow
- Add `.clang-format` draft
- Add cron next option
- Remove custom cron datetime structure
```

5.5 Authors

List of authors and contributors to the library

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

INDEX

L