LwSHELL

Tilen MAJERLE

Jan 04, 2023

CONTENTS

Welcome to the documentation for version v1.2.0.

LwSHELL is lightweight dynamic memory manager optimized for embedded systems.

Download library Getting started Open Github Donate

CHAPTER

ONE

FEATURES

- Lightweight commands shell for embedded systems
- Platform independent and very easy to port
  - Development of library under Win32 platform
- Written in C language (C99)
- No dynamic allocation, maximum number of commands assigned at compile time
- Highly configurable
- Simple help-text with *cmd -v* option
- User friendly MIT license

CHAPTER

TWO

REQUIREMENTS

- C compiler
- Less than 5kB 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) 2023 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/lwshell` command to clone entire repository, including submodules
  - Run `git clone --recurse-submodules --branch develop https://github.com/MaJerle/lwshell` to clone *development* branch, including submodules
  - Run `git clone --recurse-submodules --branch main https://github.com/MaJerle/lwshell` 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

- Copy `lwshell` folder to your project, it contains library files
- Add `lwshell/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 `lwshell/src/` folder to toolchain build. These files are built by *C/C++* compiler
- Copy `lwshell/src/include/lwshell/lwshell_opts_template.h` to project folder and rename it to `lwshell_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 needs. and it should be copied (or simply renamed in-place) and named `lwshell_opts.h`

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

List of configuration options are 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

```
1  /**
2   * \file            lwshell_opts_template.h
3   * \brief           Template config file
4   */
5
6  /*
7   * Copyright (c) 2023 Tilen MAJERLE
```

(continues on next page)

(continued from previous page)

```
 *
 * 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 LwSHELL - Lightweight shell library.
 *
 * Author:          Tilen MAJERLE <tilen@majerle.eu>
 * Version:         v1.2.0
 */
#ifndef LWSHELL_OPTS_HDR_H
#define LWSHELL_OPTS_HDR_H

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

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

#endif /* LWSHELL_OPTS_HDR_H */
```

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

5.1.4 Minimal example code

To verify proper library setup, minimal example has been prepared. Run it in your main application file to verify its proper execution

Listing 2: Absolute minimum example

```
#include <string.h>
#include "lwshell/lwshell.h"

/* Command to get called */
int32_t
mycmd_fn(int32_t argc, char** argv) {
    printf("mycmd_fn called. Number of argv: %d\r\n", (int)argc);
    for (int32_t i = 0; i < argc; ++i) {
        printf("ARG[%d]: %s\r\n", (int)i, argv[i]);
    }

    /* Successful execution */
    return 0;
}

/* Example code */
void
example_minimal(void) {
    const char* input_str = "mycmd param1 \"param 2 with space\"";

    /* Init library */
    lwshell_init();

    /* Define shell commands */
    lwshell_register_cmd("mycmd", mycmd_fn, "Adds 2 integer numbers and prints them");

    /* User input to process every character */

    /* Now insert input */
    lwshell_input(input_str, strlen(input_str));
}
```

5.2 User manual

5.2.1 How it works

This section describes how library works from the basic perspective.

LwSHELL is designed to accept *computer-command-like* input, in format of `cmdname param1 "param 2 with space"`, parse it properly and search for function callback that is assigned for specific `cmdname`.

Library starts processing input line on *line-feed* or *carriage-return* characters. It splits tokens by space character:

- Tokens must not include `space` character or it will be considered as multi-token input
- To use *space* character as token input, encapsulate character in *double-quotes*

Command structure

Every command has assigned dedicated name and must start with it. Application must take care to input exact command name since commands are case-sensitive, `mycmd` is a different command than `Mycmd`.

Command structure looks like:

- It must start with command name and has at least one (1) parameter, eg. `mycommand`. Command name is counted as first parameter
- It may have additional parameters split with *space* character
- Every input is parsed as string, even if parameter is string

Tip: To use space as an input, encapsulate it with *double quotes*, eg. `mycmd param1 "param 1 has spaces"`

Register command

Application must register command(s) to be used by the system. This can be done using `lwshell_register_cmd()` function which accepts *command name*, *command function* and optional *command description*

Command description

Every command can have assigned its very simple description text, know as *help text*. Description is later accessible with special command input that has 2 parameters in total and second is `-h`, `cmdname -h`.

Data output

To properly work with the library, application must input data to process by using `lwshell_input()` function. Thanks to the library implementation, it is possible to get data feedback and be able to implement OS-like console.

To enable data-output feature, define your output callback function and assign it with `lwshell_set_output_fn()` function.

Data outputs works on:

- Special characters for *carriage return* and *line-feed*
- Special character *backspace* that returns set of characters to implement backspace-like event on your output
- Actual input character printed back for user feedback
- `cmdname -h` feature works to print simple help text

5.3 API reference

List of all the modules:

5.3.1 LwSHELL

group **LWSHELL**

Lightweight shell.

Defines

LWSHELL_ARRAYSIZE(x)

Get size of statically allocated array.

Parameters

- **x** – **[in]** Object to get array size of

Returns Number of elements in array

lwshell_init()

Initialize shell interface.

Note: It applies to default shell instance

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshell_set_output_fn(out_fn)

Set output function to use to print data from library to user.

Note: It applies to default shell instance

Parameters

- **out_fn** – **[in]** Output function to print library data. Set to NULL to disable the feature

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshell_register_cmd(cmd_name, cmd_fn, desc)

Register new command to shell.

Note: It applies to default shell instance

Note: Available only when *LWSHELL_CFG_USE_DYNAMIC_COMMANDS* is enabled

Parameters

- **cmd_name** – **[in]** Command name. This one is used when entering shell command
- **cmd_fn** – **[in]** Function to call on command match
- **desc** – **[in]** Custom command description

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshell_input(in_data, len)

Input data to shell processing.

Note: It applies to default shell instance

Parameters

- **in_data** – **[in]** Input data to process
- **len** – **[in]** Length of data for input

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshell_register_static_cmds(cmds, cmds_len)

Register new command to shell.

Note: It applies to default shell instance

Note: Available only when *LWSHELL_CFG_USE_STATIC_COMMANDS* is enabled

Parameters

- **cmds** – **[in]** Array of const static commands. It can be from non-volatile memory
- **cmds_len** – **[in]** Length of array elements

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshell_parse_int(str)

Parse input string as `integer`

Parameters

- **str** – **[in]** String to parse

Returns String parsed as integer

lwshell_parse_double(str)

Parse input string as `double`

Parameters

- **str** – **[in]** String to parse

Returns String parsed as `double`

lwshell_parse_long(str)

Parse input string as `long`

Parameters

- **str** – **[in]** String to parse

Returns String parsed as `long`

lwshell_parse_long_long(str)

Parse input string as long long

Parameters

- **str** – **[in]** String to parse

Returns String parsed as long long

Typedefs

typedef int32_t (*lwshell_cmd_fn)(int32_t argc, char **argv)

Command function prototype.

Param argc **[in]** Number of arguments

Param argv **[in]** Pointer to arguments

Return 0 on success, -1 otherwise

typedef void (*lwshell_output_fn)(const char *str, struct lwshell *lwobj)

Callback function for character output.

Param str **[in]** String to output

Param lwobj **[in]** LwSHELL instance

Enums

enum lwshellr_t

LwSHELL result enumeration.

Values:

enumerator lwshellOK = 0x00

Everything OK

enumerator lwshellERRPAR

Parameter error

enumerator lwshellERRMEM

Memory error

Functions

lwshellr_t **lwshell_init_ex**(*lwshell_t* *lwobj)

Initialize shell interface.

Parameters **lwobj** – **[in]** LwSHELL object instance. Set to NULL to use default one

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshellr_t **lwshell_set_output_fn_ex**(*lwshell_t* *lwobj, *lwshell_output_fn* out_fn)

Set output function to use to print data from library to user.

Parameters

- **lwobj** – **[in]** LwSHELL object instance. Set to NULL to use default one
- **out_fn** – **[in]** Output function to print library data. Set to NULL to disable the feature

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshellr_t **lwshell_register_cmd_ex**(*lwshell_t* *lwobj, const char *cmd_name, *lwshell_cmd_fn* cmd_fn, const char *desc)

Register new command to shell.

Note: Available only when *LWSHELL_CFG_USE_DYNAMIC_COMMANDS* is enabled

Parameters

- **lwobj** – **[in]** LwSHELL object instance. Set to NULL to use default one
- **cmd_name** – **[in]** Command name. This one is used when entering shell command
- **cmd_fn** – **[in]** Function to call on command match
- **desc** – **[in]** Custom command description

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshellr_t **lwshell_input_ex**(*lwshell_t* *lwobj, const void *in_data, size_t len)

Input data to shell processing.

Parameters

- **lwobj** – **[in]** LwSHELL object instance. Set to NULL to use default one
- **in_data** – **[in]** Input data to process
- **len** – **[in]** Length of data for input

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

lwshellr_t **lwshell_register_static_cmds_ex**(*lwshell_t* *lwobj, const *lwshell_cmd_t* *cmds, size_t cmds_len)

Register new command to shell.

Note: Available only when *LWSHELL_CFG_USE_STATIC_COMMANDS* is enabled

Parameters

- **lwobj** – **[in]** LwSHELL object instance. Set to NULL to use default one

- **cmds** – **[in]** Array of const static commands. It can be from non-volatile memory.
- **cmds_len** – **[in]** Length of array elements

Returns *lwshellOK* on success, member of *lwshellr_t* otherwise

struct **lwshell_cmd_t**

#include <lwshell.h> Shell command structure.

Public Members

lwshell_cmd_fn **fn**

Command function to call on match

const char ***name**

Command name to search for match

const char ***desc**

Command description for help

struct **lwshell_t**

#include <lwshell.h> LwSHELL main structure.

Public Members

lwshell_output_fn **out_fn**

Optional output function

char **buff**[LWSHELL_CFG_MAX_INPUT_LEN + 1]

Shell command input buffer

size_t **buff_ptr**

Buffer pointer for input

int32_t **argc**

Number of arguments parsed in command

char ***argv**[LWSHELL_CFG_MAX_CMD_ARGS]

Array of pointers to all arguments

lwshell_cmd_t **dynamic_cmds**[LWSHELL_CFG_MAX_DYNAMIC_CMDS]

Shell registered dynamic commands

size_t **dynamic_cmds_cnt**

Number of registered dynamic commands

const *lwshell_cmd_t* ***static_cmds**

Pointer to an array of static commands

size_t **static_cmds_cnt**

Length of status commands array

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 `lwshell_opts.h` file.

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

group **LWSHELL_OPT**

LwSHELL options.

Defines

LWSHELL_CFG_USE_DYNAMIC_COMMANDS

Enables 1 or disables `0` dynamic command register with *lwshell_register_cmd* or *lwshell_register_cmd_ex* functions.

See also:

LWSHELL_CFG_USE_STATIC_COMMANDS

Note: Set to 1 by default for backward compatibility

LWSHELL_CFG_USE_STATIC_COMMANDS

Enables 1 or disables `0` static command registration.

When enabled, a single register call is used where application can pass constant array of the commands and respective callback functions.

This allows RAM reduction as lookup tables can be stored in the non-volatile memory

See also:

LWSHELL_CFG_USE_DYNAMIC_COMMANDS

Note: Set to `0` by default for backward compatibility

LWSHELL_CFG_MAX_CMDS

Maximum number of different dynamic registered commands.

Deprecated:

> **Warning:** Deprecated and replaced with *LWSHELL_CFG_MAX_DYNAMIC_CMDS*

LWSHELL_CFG_MAX_DYNAMIC_CMDS

Maximum number of different dynamic registered commands.

> **Note:** Used only when *LWSHELL_CFG_USE_DYNAMIC_COMMANDS* is enabled

LWSHELL_CFG_MAX_INPUT_LEN

Maximum characters for command line input.

This includes new line character and trailing zero. Commands longer than this are automatically discarded

LWSHELL_CFG_MAX_CMD_NAME_LEN

Maximum characters for command name in bytes.

> **Note:** Used only when *LWSHELL_CFG_USE_DYNAMIC_COMMANDS* is enabled

LWSHELL_CFG_MAX_CMD_ARGS

Maximum number of parameters accepted by command.

Number includes command name itself

LWSHELL_CFG_USE_OUTPUT

Enables `1` or disables `0` output function to print data from library to application.

This is useful to give library feedback to user

LWSHELL_CFG_USE_LIST_CMD

Enables `1` or disables `0` generic listcmd` command to list of registered commands.

LWSHELL_CFG_USE_OUTPUT must be enabled to use this feature

5.4 Examples and demos

Various examples are provided for fast library evaluation on embedded systems. These are prepared and maintained for 2 platforms, but could be easily extended to more platforms:

- WIN32 examples, prepared as Visual Studio Community projects
- ARM Cortex-M examples for STM32, prepared as STM32CubeIDE GCC projects

Warning: Library is platform independent and can be used on any platform.

5.4.1 Example architectures

There are many platforms available today on a market, however supporting them all would be tough task for single person. Therefore it has been decided to support (for purpose of examples) 2 platforms only, *WIN32* and *STM32*.

WIN32

Examples for *WIN32* are prepared as Visual Studio Community projects. You can directly open project in the IDE, compile & debug.

STM32

Embedded market is supported by many vendors and STMicroelectronics is, with their STM32 series of microcontrollers, one of the most important players. There are numerous amount of examples and topics related to this architecture.

Examples for *STM32* are natively supported with STM32CubeIDE, an official development IDE from STMicroelectronics.

You can run examples on one of official development boards, available in repository examples.

5.5 Changelog

```
# Changelog

## Develop

- Change license year to 2022
- Update code style with astyle
- Add `.clang-format` draft
- Add option for statically allocated commands array (improvement for small devices w/␣
↪little memory)
- Add option to disable dynamic commands allocation (default value)

## v1.1.1

- Split CMakeLists.txt files between library and executable
- Fix wrongly interpreted backspace character
```

(continues on next page)

(continued from previous page)

```
## v1.1.0

- Add support for `listcmd` to print all registered commands
- Optimize code and remove unnecessary brackets

## v1.0.0

- First stable release
- Fix wrong parsing of command names

## v0.1.0

- First release
```

INDEX

L