GigaDevice Semiconductor Inc.

GD32F10x Arm[®] Cortex[®]-M3 32-bit MCU

Application Note AN019



Tables of Contents

Table	s of Contents	.2
List o	f Figures	. 3
List o	f Tables	.4
1. In	troduction	. 5
2. Li	a interpreter migration	. 6
2.1.	Lua download	. 6
2.2.	Add Lua source code file	. 7
3. Li	ia usage test	11
3.1.	Test	11
3.2.	Other instruction	12
4. R	evision history	13



List of Figures

Figure 2-1. Lua download interface	6
Figure 2-2. Lua historical version download interface	6
Figure 2-3. Lua file after decompression	7
Figure 2-4. Lua file add path	7
Figure 2-5. Lua project configuartion	8
Figure 2-6. Add c file to Lua project configuration	8
Figure 2-7h file path configuration	8
Figure 2-8. Change os_exit(lua_State * L)	9
Figure 2-9. Added function	9
Figure 3-1. ROM and RAM occupancy	12



List of Tables

able 3-1. main.c	.11
able 4-1. Revision history	13



1. Introduction

Lua is a powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description. Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode with a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping. Lua is implemented as a library by cleanC (a subset common between standard C and C++). As an extended language, Lua does not have the concept of a "main" program, it can only work in a host program, which is called an embedded program or host for short. The host program can call functions to execute a small pieces of Lua code, can read and write Lua variables, and can register C functions for Lua code to call. Relying on C functions, Lua can share the same grammatical framework to customize the programming language, which can be applied to different fields. The official release of Lua contains a sample host program called Lua, which is a complete and independent Lua interpreter implemented using the Lua library, which can be used for interactive applications or batch processing.

Lua is an open source software language, and its use license determines its use without any guarantee.

This article describes how to port Lua to the GD32 project.



2. Lua interpreter migration

2.1. Lua download

The Lua interpreter transplantation platform introduced in this article is the GD32F103E-EVAL board. The IDE platform ported by Lua interpreter is KEIL4.

Lua source code can be downloaded from https://www.lua.org/. The currently tested Lua version is 5.4.2, as shown in the figure below.

Figure 2-1. Lua download interface



Figure 2-2. Lua historical version download interface

Lua	Downle	oad a	rea	
source · manuals ·	license · versions	· work area		
This repository cor	tains the source c	ode and the re	ference manuals for all released versions of Lua.	
If you want to build	early versions of	ua using mod	lern compilers, get the lua-all package.	
If you have arrived	here by accident,	start here.		
If you are looking f	or work versions, r	elease candid	ates, and other pre-releases of Lua, check out the wor	
All files are distributed under this license. Check their checksums to confirm the integrity of the packages.				
Source of	code			
filename	date	size	checksums	
lua-5.4.2.tar.gz	2020-11-13	353472	md5: 49c92d6a49faba342c35c52e1ac3f81e sha1: 96d4a21393c94bed286b8dc0568f4bdde8730b22	
lua-5.4.1.tar.gz	2020-09-30	353965	md5: 1d575faef1c907292edd79e7a2784d30 sha1: 88961e7d4fda58ca2c6163938fd48db8880e803d	
lua-5.4.0.tar.gz	2020-06-18	349308	md5: dbf155764e5d433fc55ae80ea7060b60 sha1: 8cdbffa8a214a23d190d7c45f38c19518ae62e89	
lua-5.3.6.tar.gz	2020-09-14	303770	md5: 83f23dbd5230140a3770d5f54076948d sha1: f27d20d6c81292149bc4308525a9d6733c224fa5	
lua-5.3.5.tar.gz	2018-06-26	303543	md5: 4f4b4f323fd3514a68e0ab3da8ce3455 sha1: 112eb10ff04d1b4c9898e121d6bdf54a81482447	
lua-5.3.4.tar.gz	2017-01-12	303586	md5: 53a9c68bcc0eda58bdc2095ad5cdfc63 sha1: 79790cfd40e09ba796b01a571d4d63b52b1cd950	
lua-5.3.3.tar.gz	2016-05-30	294290	md5: 703f75caa4fdf4a911c1a72e67a27498 sha1: a0341bc3d1415b814cc738b2ec01ae56045d64ef	
lua-5.3.2.tar.gz	2015-11-25	288235	md5: 33278c2ab5ee3c1a875be8d55c1ca2a1 sha1: 7a47adef554fdca7ddc5536148de34579134a973	

> Unzip the compressed package to get the file to be transplanted



Figure 2-3. Lua file after decompression

名称 ^
doc
src
Makefile
README
lua-5.4.2

> Delete the lua.c and luac.c files under the decompressed lua-5.4.2\src file.

2.2. Add Lua source code file

The transplantation project introduced in this article is based on the 01_GPIO_Runnin g_LED in GD32F10x Demo_Suite _V2.1.0

Copy the files after deleting lua.c and luac.c to the GD32F10x_Firmware_Library f ile, as shown in Figure 2.4.

Figure 2-4. Lua file add path

→ GD32F10x_Demo_Suites_V2.1.0 >	GD32F10x_Firmware_Library
名称	
GD32F10x_standard_peripheral	
GD32F10x_usbd_driver	
GD32F10x_usbfs_driver	
📙 lua-5.4.2	

> Open the project. Add all .c files in lua-5.4.2\src to the project.



Figure 2-5. Lua project configuartion



Figure 2-6. Add c file to Lua project configuration

Groups:	🖄 🗙 🗲	Files:	× 🗲 🗲
Application CMSIS GD32F10x_Perip GD32F10x_EVAL Startup Doc	herals -	lapi c lauxlib.c lbaselib.c lcode.c lcorolib.c lctype.c ldbib.c ldbug.c ldo.c ldump.c lfunc.c lgc.c linit.c liolib.c lex.c mathlib.c mem.c loadib.c lobject.c	~
	Add	all .C files	Files. SFC as Image

> Configure Include Paths

Figure 2-7. .h file path configuration



🗹 🔊 者 🗟 🗇 🖄		
😨 Options for Target 'GD32F103E_EVAL	Ľ	×
Device Target Output Listing User	C/C++ Asm Linker Debug	Utilities
- Preprocessor Sumbole		
	D32F10X HD	
Lindefine:		
Language / Code Generation		Wamings:
	Strict ANSIC	All Warnings 👻
Optimization: Level 0 (-00)	Enum Container aiways int Plain Charlin Signad	
Optimize for Time Solit Load and Storp Multiple	Read-Only Position Independent	No Auto Includes
One ELE Section per Function	Read-Write Position Independent	C99 Mode
Include Paths	Library\CMSIS;\\\GD32F10x_Firmwa	are_Library\CMSIS\GE
Misc		
Controls		
control \GD32F10x_Firmware_Library\/	LIB -g -OUapcs=interworksplit_sections CMSIS -I\\\GD32F10x_Firmware_Lib	-I\\ A
string		*
6к	Cancel Defaults	Help
Folder Setup	? >	<
Setup Compiler Include Paths:	🖄 🗙 🗲	4
	201000000000000000000000000000000000000	-
GD32F10x_Firmware_Library/CMSIS/G GD32F10x_Firmware_Library/GD32F10	Dx_standard_peripheral\Include	
\\Utilities		
	\src 4	
5 ок	Cancel	

- > Change part of the content under the loslib.c file
 - 1. Comment out if(L) exit(status) in the os_exit(lua_State * L) function, and add a status=status statement.

Figure 2-8. Change os_exit(lua_State * L)



2.Add time(time_t *time) and system(const char * string)

Figure 2-9. Added function



```
432 time_t · time (time_t · * · time)
433 ⊟ {
434 ....return · 0;
435 }
436 437 int · system (const · char · * · string)
438 ⊟ {
439 ....return · 0;
440 }
```

The above changes are due to the use of Use MicroLIB mode



3. Lua usage test

3.1. Test

Lua usage tests are performed after the project configuration and related code changes are completed. This section uses the C language to interact with Lua to light up the led lights.

Table 3-1. main.c

```
#include "gd32f10x.h"
#include "gd32f103e_eval.h"
#include "systick.h"
#include "lua.h"
#include "lualib.h"
#include "lauxlib.h"
static int lua_led_on(lua_State *L)
{
    gd_eval_led_on(LED3);
    return 1;
}
static const struct luaL_Reg mylib[]=
{
  {"led_on",lua_led_on},
  {NULL,NULL}
};
const char LUA_SCRIPT_GLOBAL[] =" \
while 1 do \
led_on() \
end";
int main(void)
{
    gd_eval_led_init(LED3);
    while(1)
    {
        lua_State *L;
         L = luaL_newstate();
        luaopen_base(L);
        luaL_setfuncs(L, mylib, 0);
        luaL_dostring(L, LUA_SCRIPT_GLOBAL);
    }
}
```

After compiling the project and downloading it to the development board, LED3 will light up.



3.2. Other instruction

The ROM and RAM occupancy after compiling and running is shown in the figure below. When transplanting to other development boards, pay attention to the memory size of the development board, otherwise there will be a problem of unsuccessful migration.

Figure 3-1. ROM and RAM occupancy

```
·Total RO · Size (Code + RO Data) · · · · · · · · · · · 89808 ( · 87.70kB)
·Total RW · Size (RW Data + ZI Data) · · · · · · 16544 ( · 16.16kB)
·Total ROM Size (Code + RO Data + RW Data) · · · · 89964 ( · 87.86kB)
```



4. Revision history

Table 4-1. Revision history

Version number	Description	Date
1.0	Released the first draft	2021.3.18



Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it's suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it's suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.

© 2021 GigaDevice - All rights reserved