技术笔记

Technical Note

Lua 应用-文件读写 V1.0

工程技术笔记

类别	内容
关键词	Lua 脚本、文件读写
摘要	

广州大彩光电科技有限公司

修订历史

版本	日期	原因	编制	审查
V1.0	2020/07/02	创建文档	陈鹏	

销售与服务

广州大彩光电科技有限公司

电话：020-82186683

传真：020-82187676

Email：hmi@gz-dc.com（公共服务）

网站：www.gz-dc.com

地址：广州高新技术产业开发区玉树工业园富康西街 8 号 C 栋 303 房

官网零售淘宝店：www.gz-dc.taobao.com

目录

1. 适合范围.....1
2. 开发环境版本.....2
3. 概述.....3
4. 参考资料.....4
5. 教程实现.....5
5.1.1 准备工程素材.....5
5.1.2 软件平台.....5
5.2 配置串口屏工程.....6
5.2.1 读文件.....6
5.2.2 写文件.....10
5.3 下载工程.....16
6. 免责声明.....18

1. 适合范围

本文档适合大彩支持 Lua 脚本的串口屏产品使用。

2. 开发环境版本

1. VisualTFT 软件版本：V3.0.1.1111 及以上的版本。

版本查看：

1) 打开 VisualTFT 软件启动页面如图 2-1 软件版本，右上角会显示的软件版本号；



图 2-1 软件版本

2) 打开VisualTFT，在软件右下角可以查看软件版本图 2-2 软件版本，最新版本可登录 http://www.gz-dc.com/进行下载。

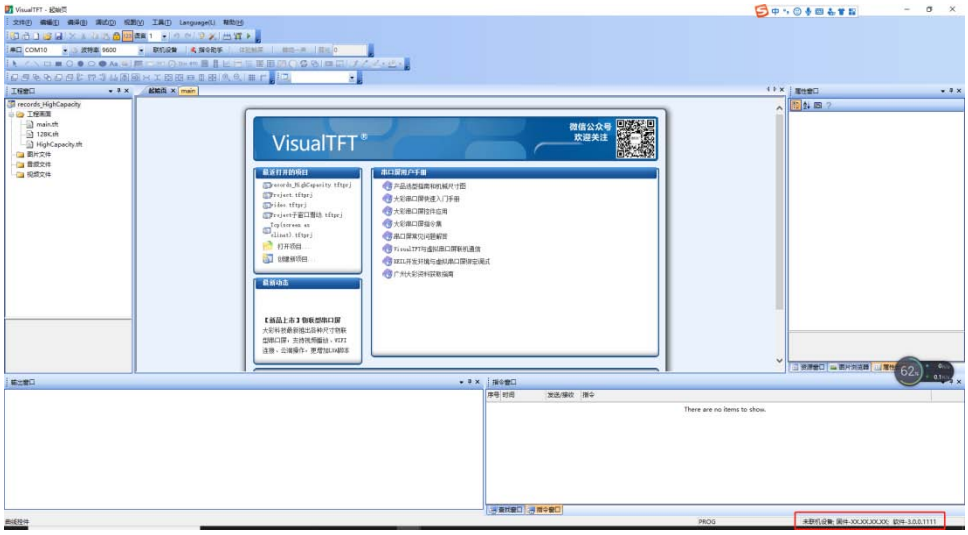


图 2-2 软件版本

2. 串口屏硬件版本：物联型固件 >= V3.0.731.0、F 型固件 >= V4.1.460.0，M 系列固件 >= V6.1.199.00。

版本查看：

1) 查看屏幕背面版本号贴纸；
2) VisualTFT 与屏幕联机成功后，右下角显示的版本号。

3. 概述

本例程中，介绍Lua系统函数中的文件API读写，其中F系列和M系列仅支持对SD卡读写，物联型系列支持对屏内及SD卡、U盘的读写。

4. 参考资料

1. 《LUA 脚本 API V1.4》可通过以下链接下载物联型开发包获取:
   http:/www.gz-dc.com/index.php?s=/List/index/cid/19.html
2. 《LUA 基础学习》可通过以下链接下载物联型开发包获取:
   http:/www.gz-dc.com/index.php?s=/List/index/cid/19.html
3. LUA 脚本初学者可以通过下面链接进行学习。
   http://www.runoob.com/lua/lua-arrays.html

5. 教程实现

本文主要将以下 2 点进行说明：

1. 准备工程素材；
2. 配置串口屏工程；

5.1.1 准备工程素材

在实现例程前需要作以下 3 个准备：

1. 硬件平台；
2. 软件平台；
3. UI 素材；

该例程使用大彩 M 系列 7 寸串口屏 DC80480M070_1111_0T 为验证开发平台。如图 5-1 所示；

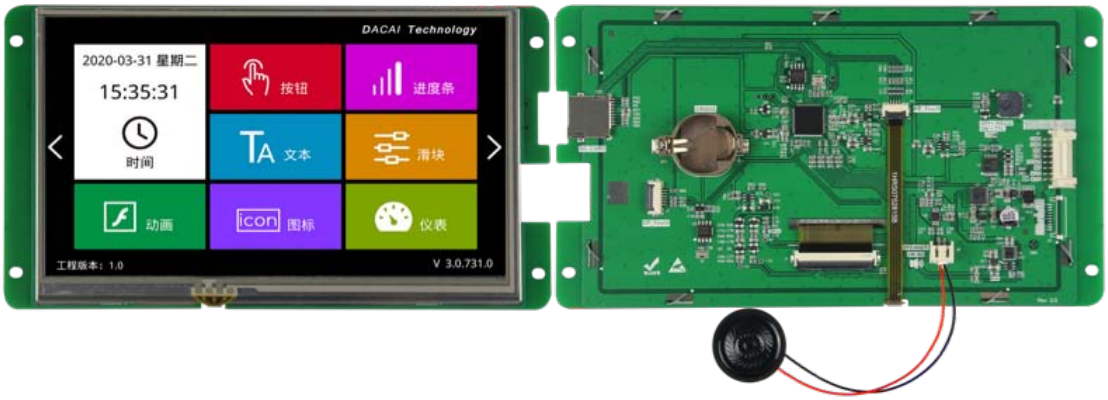


图 5-1 M 系列 7 寸串口屏

其他尺寸、系列的串口屏均可借鉴此教程。

5.1.2 软件平台

使用大彩自主研发的上位机软件VisualTFT配置工程，登录 http://www.gz-dc.com/下载。如图 5-2 所示；

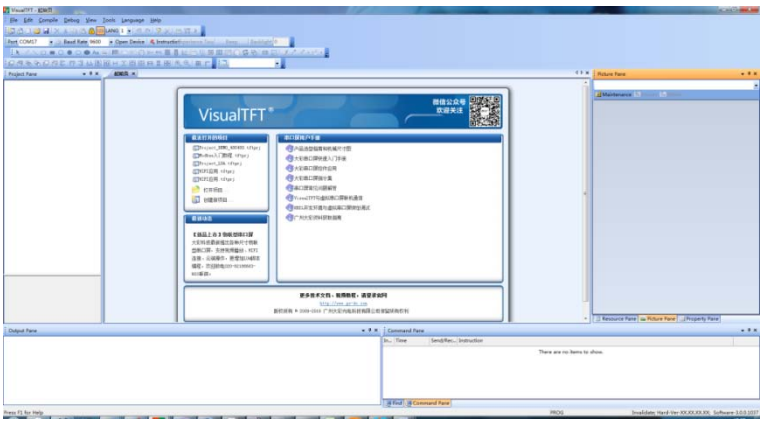


图 5-2 下载软件

5.2 配置串口屏工程

本文的文件主要介绍以下 2 点：

(1) 读文件

(2) 写文件

5.2.1 读文件

本例程中的读文件，是读取 SD 根目录下的‘1.txt’文件，并用数据记录控件显示出来，用户可滑动数据记录控件，查看‘1.txt’里面的内容。

1. 画面配置

在画面 ID1 中，添加 1 个数据记录控件（控件 ID1）和一个按钮控件（控件 ID2），其中控件 ID1 用于显示‘1.txt’文件的内容。控件 ID2 用于触发读取‘1.txt’文件的按钮。画面配置如图 5-4 所示：

注意：其他非关键控件不在一一介绍，下文不在累述

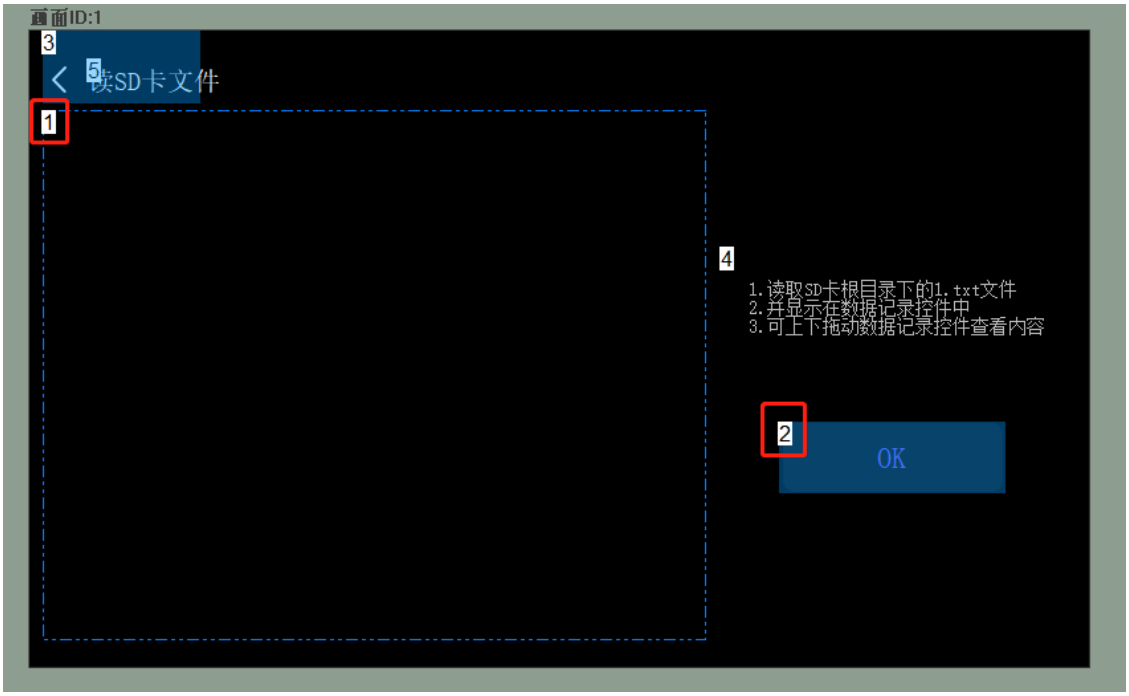


图 5-3 画面配置

2. LUA 脚本编辑

本例程中，用户点击按钮控件 ID2 后，将读取 SD 目录下载的‘1.txt’文件，并添加到数据记录控件中显示；若文件不存在，则弹框提示。代码如程序清单 1 所示：

程序清单 1 读文件

```
--read file data
--@file:file path
function my_read_filedata(file)
    my_debug_print('---------- my_read_filedata ----------')
    my_debug_print('Lua_debug: file -> '..file)
```

```
local count    = 0
local read_cnt = 0
local offset   = 0
local all_byte = 0

local read_byte_Tb = {}
local read_char_Tb = {}

local read_str    = ''
local open_state = file_open(file, FA_READ)
if open_state == true
then
    --获取当前文件大小
    all_byte = file_size()
    if all_byte > 0
    then
        read_cnt =  math.modf(all_byte/WriteOnceSize)
        if all_byte % WriteOnceSize > 0
        then
            read_cnt = read_cnt + 1
        end
        my_debug_print('Lua_debug: need read allcnt -> '..read_cnt)

        for i = 1, read_cnt
        do
            --复位读字节数组
            read_byte_Tb = {}
            read_char_Tb = {}

            --计算读取的偏移位置
            offset = (i - 1) * WriteOnceSize
            local offst_result = file_seek(offset)
            --文件偏移失败
            if offst_result == false then
                set_text(sc_prompt, 1, ' When reading the file, an offset
                          error occurred. please try again! ! !')
                set_text_roll(sc_prompt, 1, 50)
                change_child_screen(sc_prompt)
                braek;
            end
            my_debug_print('Lua_debug: cur offset  -> '.. offset)

            --计算本次读的个数
            count = WriteOnceSize
```

```
                if i == read_cnt
                then
                    if all_byte % WriteOnceSize > 0
                    then
                        count = all_byte % WriteOnceSize
                    end
                end
                my_debug_print('Lua_debug: cur read  -> '..i..'th /
                               wrire count '..count)

                --读取字节转换为字符并拼接成字符串
                read_byte_Tb = file_read(count)
                if #(read_byte_Tb) > 0
                then
                    for j = 0, #(read_byte_Tb)
                    do
                        read_char_Tb[j + 1] = string.char(read_byte_Tb[j])
                    end
                    read_str = read_str..table.concat(read_char_Tb)
                elseif read_byte_Tb ==nil
                then
                    set_text(sc_prompt, 1, ' File read error. please try
                             again! ! !')
                    set_text_roll(sc_prompt, 1, 50)
                    change_child_screen(sc_prompt)
                    braek;
                end
            end
            my_ShowRecord_FromFile(read_str)
        end
    else
        set_text(sc_prompt, 1, 'The file don`t exist,
                 please check the contents of the SD car! ! !')
        set_text_roll(sc_prompt, 1, 50)
        change_child_screen(sc_prompt)
    end

    --关闭文件
    file_close()
end

--用户通过触摸修改控件后，执行此回调函数。
--点击按钮控件，修改文本控件、修改滑动条都会触发此事件。
function on_control_notify(screen,control,value)
```

```
    if screen == sc_ReadFlile and control == 2 and value == 0
    then
        sc_ShowRecord = sc_ReadFlile

        --SD 卡是否存在
        if IsinsertSD == 1
        then
            record_clear(sc_ReadFlile, 1)
            my_read_filedata(sd_dir..'/'..'1.txt')
        else
            set_text(sc_prompt, 1, 'Please insert the SD card,
                     or check if the SD is compatible! ! !')
            set_text_roll(sc_prompt, 1, 50)
            change_child_screen(sc_prompt)
        end

    ......
    end
end
```

核心 API 函数

1) file_open(path,mode)

打开文件，成功返回 true，失败 false

- path-文件路径
- mode-打开模式，如下组合方式：

打开模式	值
FA_OPEN_EXISTING	0x00
FA_READ	0x01
FA_WRITE	0x02
FA_CREATE_NEW	0x04
FA_CREATE_ALWAYS	0x08
FA_OPEN_ALWAYS	0x10

例如：

打开文件用于读取 file_open(path, 0x01)

创建文件用于写入 file_open(path, 0x02|0x08)

2) file_size()

获取当前文件大小，返回字节数

例如：all_byte = file_size()

3) file_seek(offset)

定位文件读取位置，成功返回 true，失败 false

- offset-文件偏移位置

例如：定位到第 2048 个字节，file_seek(2048)

4) file_read(count)

读取文件内容，成功返回 table 数组，失败返回 nil

- count-读取字节数，最大读取 2048 个字节

例如：读取 2048 个字节，read_byte_Tb = file_read(2048)

5) file_close()

关闭文件，成功返回 true，失败 false

基本思路：当按钮控件 ID2 按下的时候，触发触摸控件回调函数 on_control_notify()，在调用自定义函数 `my_read_filedata()`。

1) 打开文件：以只读的方式 FA_READ（0x01）打开指定文件。
2) 获取文件大小:文件打开成功后，调用 file_size() Api 函数获取该文件的大小。
3) 计算读取次数：根据文件大小，得出读取的次数。本例程采取一次读取 2048 个字节（注意：屏幕一次读取最大字节 <= 2048）
4) 计算读取偏移量：每次读取需要调用 file_seek(offset)定位读取位置，offset 相当于已读取的字节数
5) 读出数据：本文读取的数据，转换成字符拼接字符串显示出来，详细应用根据实际情况出来。
6) 关闭文件：文件读取完毕，将该文件关闭。

5.2.2 写文件

常用的写文件操作主要有以下两种：

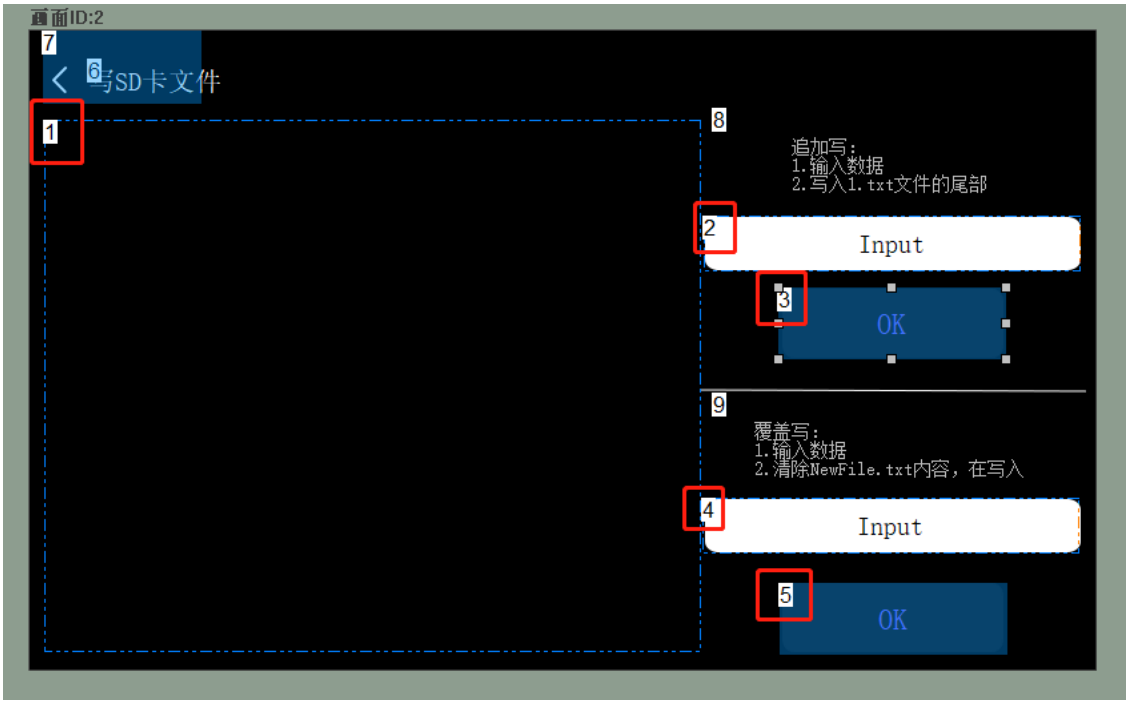
1) 追加写：写在文件末尾。本例程是写在 SD 根目录下的‘1.txt’文件末尾，并用数据记录控件显示出来。
2) 覆盖写：清空在写入数据。本例程在 SD 卡目录下，每次写入创建一个新的 NewFile.txt，并写入数据。

追加写和覆盖写的功能实现区别：

1) 打开方式：file_open(path,mode)。
   - 追加写：mode = FA_WRITE|FA_READ；
   - 覆盖写：mode = FA_CREATE_ALWAYS|FA_WRITE；
2) 写入位置：file_seek(offset)。
   - 追加写：offset = file_size() + (i - 1) * WriteOnceSize；其中 WriteOnceSize 为一次写入数据的大小，i 为循环写入的次数。
   - 覆盖写：offset = 0；

1. 画面配置

在画面 ID2 中，添加 1 个数据记录控件、2 个文本控件（控件 ID2、控件 ID4）和 2 个按钮控件（控件 ID3、控件 ID5），其中数据记录控件仅做显示文件内容效果。按钮控件 3 用于体现追加写功能，按钮控件 5 用于覆盖写功能。画面配置如图 5-4 所示：



2. LUA 脚本编辑

按钮控件 ID3 按下时候，将文本控件 ID2 的内容写在 SD 根目录下的‘1.txt’文件末尾（追加写）；按钮控件 ID5 按下时候，将文本控件 ID4 的内容写在 SD 根目录下的‘NewFile.txt’文件（覆盖写）代码如程序清单 2 所示。

程序清单 2 写文件

```
--get the type and length of the variable
--@data: 'string' or 'table'
--@return: len and type of data
function my_getdataLen_Type(data)

    my_debug_print('---------- my_getdataLen_Type ----------')
    local datalen = -1
    --获取数据类型
    local data_type = type(data)

    --计算数据长度
    if data_type == 'string'
    then
        datalen = string.len(data)
    elseif data_type == 'table'
    then
        datalen = #(data)
    end
    my_debug_print('Lua_debug data Type and Len: '..data_type..' /
                   '..datalen)
```

```
    return datalen,data_type
end

--Write data to the end of the file
--@file:file path
--@data:The type of '@data' only be [string] or [byte array]
--@open_mode:open file mode
function my_write_filedata(file, data, open_mode)

    my_debug_print('---------- my_write_filedata ----------')
    my_debug_print('Lua_debug: file -> '..file..' / data -> '..type(data)..'
                   / open_mode -> '..open_mode)
    local count    = 0
    local write_cnt = 0
    local offset   = 0
    local all_byte  = 0
    --获取待写入数据的数据类型和长度
    local wrire_len, data_type = my_getdataLen_Type(data)

    local write_byte_Tb = {}
    local open_state = file_open(file, open_mode)

    if open_state == true
    then
        --获取当前文件大小,仅在追加文件末尾写入执行
        if open_mode == add_write
        then
            all_byte = file_size()
        end

        if wrire_len > 0
        then
            --计算'@data'要写多少次
            write_cnt =  math.modf(wrire_len / WriteOnceSize)
            if wrire_len % WriteOnceSize > 0
            then
                write_cnt = write_cnt + 1
            end
            my_debug_print('Lua_debug: need write allcnt -> '..write_cnt)

            for i = 1, write_cnt
            do
                --复位写字节数组
                write_byte_Tb = {}
```

```
--计算写的位置
offset = (i - 1) * WriteOnceSize +all_byte
local offst_result = file_seek(offset)
--文件偏移失败
if offst_result == false
then
    set_text(sc_prompt, 1, 'When reading the file, an offset
             error occurred. please try again! ! !')
    set_text_roll(sc_prompt, 1, 50)
    change_child_screen(sc_prompt)
    break
end
my_debug_print('Lua_debug: cur offset  -> '..offset)

--计算本次写的个数
count = WriteOnceSize
if i == write_cnt
then
    if wrire_len % WriteOnceSize > 0
    then
        count = wrire_len % WriteOnceSize
    end
end
my_debug_print('Lua_debug: cur write  -> '..write_cnt..'th /
                wrire count '..count)
--填充写入 flash 的字节数组
for j = 1, count
do
    if data_type == 'string'
    then
    --字符串类型，将每个字符转换为字节数组
    write_byte_Tb[j - 1] = tonumber(string.byte(data,
                   ((i - 1) * WriteOnceSize + j),
                   ((i - 1) * WriteOnceSize + j)))

    elseif data_type == 'table'
    then
        --数组类型，字节赋值
        write_byte_Tb[j - 1] =
        data[((i - 1) * WriteOnceSize + j)]
    end
end
local IswriteOK = file_write(write_byte_Tb)
```

```
                if IswriteOK == false
                then
                    i = i - 1
                end
            end
        end
    else
        set_text(sc_prompt, 1, 'The file don`t exist, please check the contents
                 of the SD car! ! !')
        set_text_roll(sc_prompt, 1, 50)
        change_child_screen(sc_prompt)
    end

    --关闭文件
    file_close()
end

--用户通过触摸修改控件后，执行此回调函数。
--点击按钮控件，修改文本控件、修改滑动条都会触发此事件。
function on_control_notify(screen,control,value)
    ......
    --追加写
    elseif screen == sc_WriteFile and control == 3 and value == 0
    then
        sc_ShowRecord = sc_WriteFile
        --已插入 SD 卡
        if IsinsertSD == 1
        then
            record_clear(sc_WriteFile, 1)

            local str = '\n'..get_text(sc_WriteFile, 2)
            my_write_filedata(sd_dir..'/'..'1.txt', str, add_write)

            my_read_filedata(sd_dir..'/'..'1.txt')
            local allrecordcnt = record_get_count(sc_WriteFile, 1)
            record_setoffset(sc_WriteFile, 1, allrecordcnt - 1)
            record_select(sc_WriteFile   , 1, allrecordcnt - 1)
        --未插入 SD 卡
        else
            set_text(sc_prompt, 1, 'Please insert the SD card, or check if
                     the SD is compatible! ! !')
            set_text_roll(sc_prompt, 1, 50)
            change_child_screen(sc_prompt)
        end
```

```
    --覆盖写
    elseif screen == sc_WriteFile and control == 5 and value == 0
    then
        sc_ShowRecord = sc_WriteFile
        --已插入 SD 卡
        if IsinsertSD == 1
        then
            record_clear(sc_WriteFile, 1)

            cnt = cnt + 1
            local str = (cnt + 1)..'th write -> '..get_text(sc_WriteFile, 4)
            my_write_filedata(sd_dir..'/'..'NewTxtFile.txt',
                              str,over_write)

            my_read_filedata(sd_dir..'/'..'NewTxtFile.txt')
        --未插入 SD 卡
        else
            set_text(sc_prompt, 1, 'Please insert the SD card, or check
                     if the SD is compatible! ! !')
            set_text_roll(sc_prompt, 1, 50)
            change_child_screen(sc_prompt)
        end
    ......
    end
end
```

核心 API 函数

1) file_open(path,mode)

相关说明参考读文件章节，不在赘述

2) file_size()

相关说明参考读文件章节，不在赘述

3) file_seek(offset)

相关说明参考读文件章节，不在赘述

4) file_write(data)

写文件内容，成功返回 true，失败返回 false

- data-待写入的 table 数组，索引从 0 开始，最大一次性写 2048 个字节

5) file_close()

相关说明参考读文件章节，不在赘述

基本思路：当按钮控件 ID3 或按钮控件 ID5 按下的时候，触发触摸控件回调函数 on_control_notify()，在调用自定义函数 my_write_filedata()。

1) 打开文件：
   - 追加写：以读写的方式（FA_READ|FA_WRITE：0x01|0x02）打开 SD 卡目录下的 1.txt 文件。
   - 覆盖写：以新建写入的方式（FA_CREATE_ALWAYS|FA_WRITE：0x08|0x02）打开 SD 卡目录下的 NewFile.txt 文件
2) 获取文件大小:文件打开成功后，若追加写模式，则执行 file_size() Api 代码段。
3) 计算写入次数：根据写入数据的大小，得出读取的次数。本例程采取一次写入 2048 个字节（注意：屏幕一次读取最大字节 <= 2048）
4) 计算写入的偏移量：每次写入需要调用 file_seek(offset)定位读取位置
5) 写入数据
6) 关闭文件：文件写完后完毕，将该文件关闭。

5.3 下载工程

工程编译成功后在输出窗口会提示编译成功，如图 5-8 所示。编译成功后点击菜单栏中【工具】→【量产向导】，如图 5-9 所示；

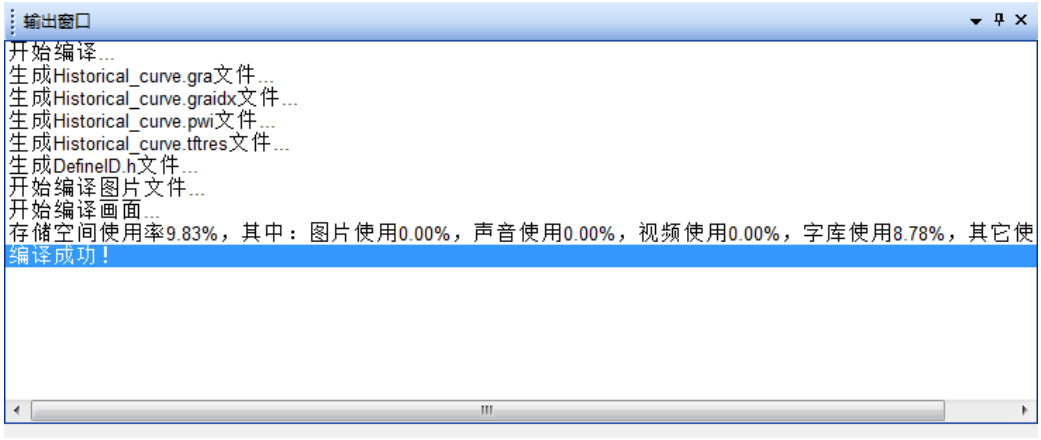


图 5-4 编译成功

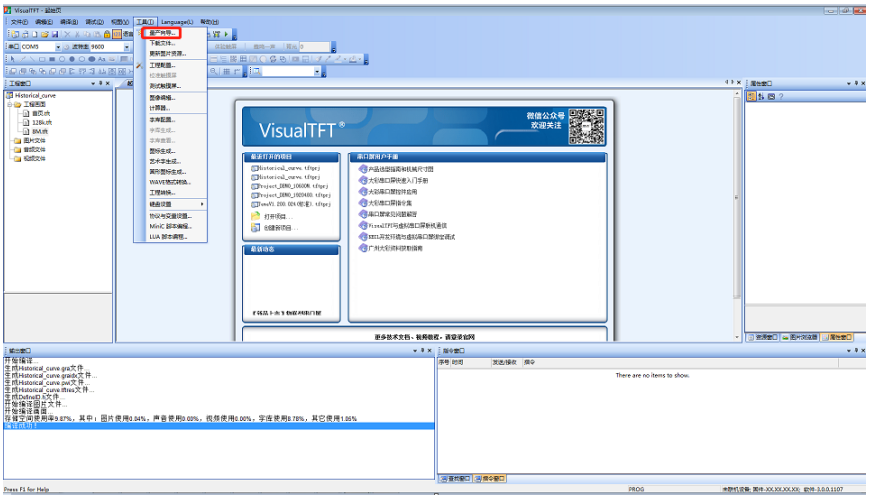


图 5-5 量产向导

在弹窗中选中【SD 卡下载】，然后把将文件夹中的 private 文件夹拷贝到 SD 卡中，如图 5-10 和图 5-11 所示；把 SD 卡接上串口屏后重新上电，等到提示烧录工程成功后，拔掉 SD 卡重新上电即可。

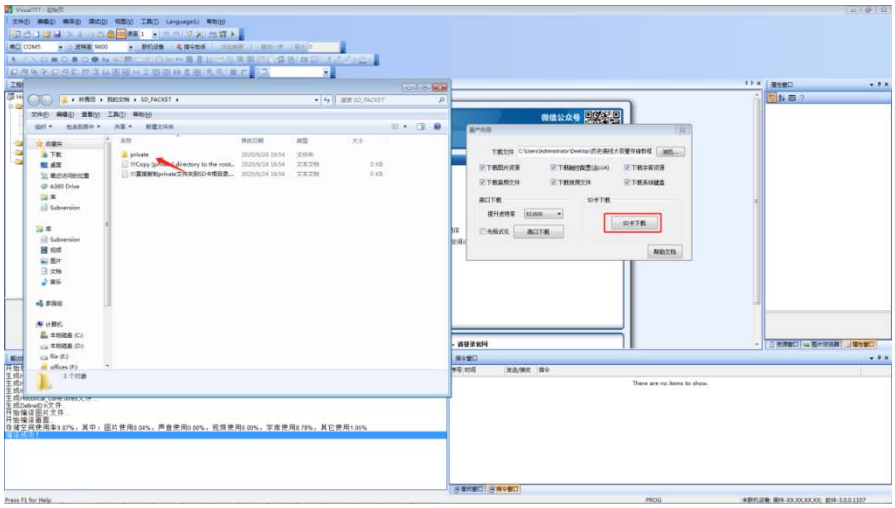


图 5-6 量产向导

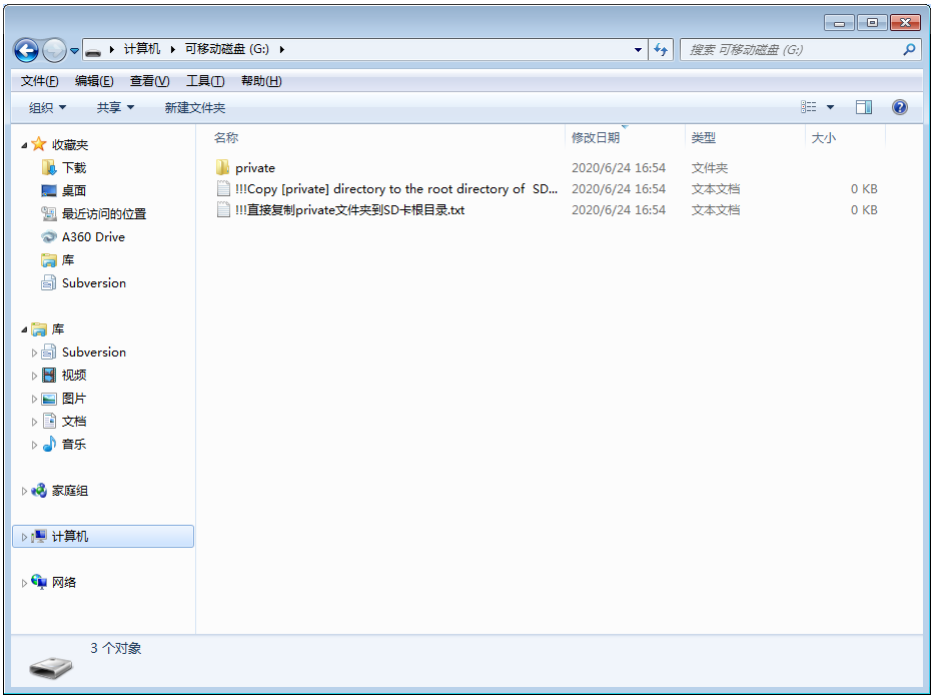


图 5-7 拷贝到 SD 卡

6. 免责声明

本文档提供有关广州大彩光电科技有限公司（以下简称：大彩科技）产品的信息，旨在协助客户加速产品的研发进度，在服务过程中或者其他渠道所提供的任何例程程序、技术文档、CAD 图等资料和信息都仅供参考，客户有权不使用或自行参考修改。本公司不提供任何的完整性、可靠性等保证，若在客户使用过程中因任何原因造成的特别的、偶然的或间接的损失，本公司不承担任何责任。大彩科技产品不能在用于军事、医疗、救生或维生等用途中作为唯一控制设备。

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除大彩科技在其产品的销售条款和条件中声明的责任之外，大彩科技概不承担任何其它责任。并且，大彩科技对大彩科技产品的销售和/或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。大彩科技可能随时对产品规格及产品描述做出修改，恕不另行通知。