

类别	内容
关键词	Lua 脚本、OTA 升级
摘要	



## 修订历史

版本	日期	原因	编制	审查
V1.0	2020/12/28	创建文档	陈鹏	
V1.1	2021/12/14	修改VisualTFT版本描述	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](mailto:hmi@gz-dc.com)（公共服务）

网站: [www.gz-dc.com](http://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](http://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 准备工程素材.....	5
5.1.1 软件平台.....	5
5.2 配置串口屏工程.....	6
5.2.1 OTA文件生成.....	6
5.2.2 画面配置.....	7
5.2.3 LUA API介绍.....	7
5.2.4 串口升级OTA流程图.....	8
5.2.5 LUA 串口升级源码.....	10
5.2.6 LUA SD升级源码.....	17
6. 免责声明.....	19
附录A.....	20



## 1. 适合范围

本文档适合医用级的串口屏产品使用。

## 2. 开发环境版本

1. VisualTFT 软件版本: V3.0.0.1137 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。

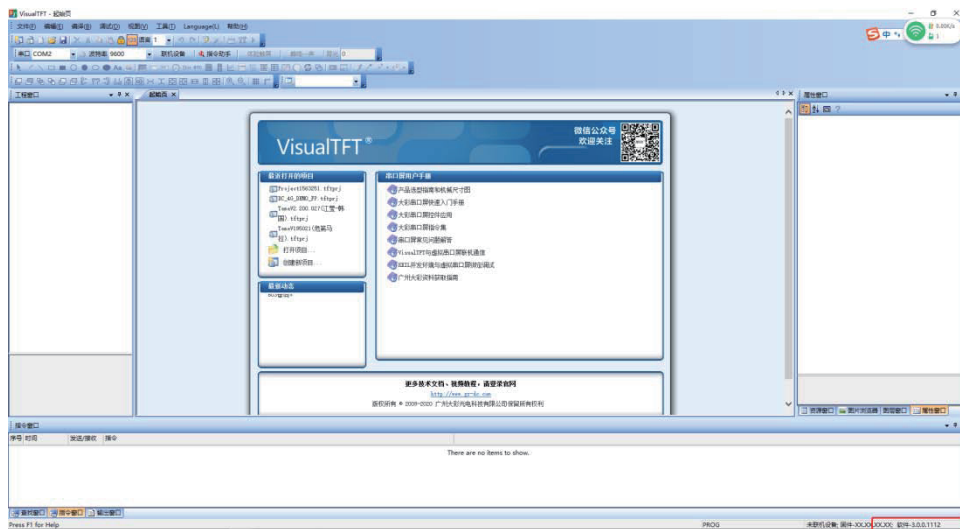


图 2-2 软件版本

2. 串口屏硬件版本: M 系列固件  $\geq$  **V6.3.319.00**。

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, 右下角显示版本号。

### 3. 概述

本文介绍 VisualTFT PC 软件编译生成的 ‘ota.bin’ 工程文件，用户通过 MCU 用串口对屏幕升级的说明。相比之前的 对 ‘private’ 文件升级，区别如下所示：

- 文件数量：
  - ‘private’ 是一个文件夹，里面含有多个文件；
  - ‘ota.bin’ 是一个文件
- 串口升级协议：
  - ‘private’ 升级的协议是固定，可参考 <http://www.gz-dc.com/uploads/file/>;
  - ‘ota.bin’ 升级协议不限制，可自定义帧内容。
- 串口数据交互：
  - ‘private’ 指令交互由固件处理；
  - ‘ota.bin’ 由 lua 脚本处理。
- 存储方式：
  - ‘private’ 是直接覆盖屏内对应的文件；
  - ‘ota.bin’ 本例程中是指定存储在地址 0x800000。

假设屏幕 flash 大小为 128Mbit,则 0x800000 表示从 8\*1024\*1024 地方开始存储 ‘ota.bin’ 的文件，需要要求从起始存储地址（8M / 0x800000）~ 最后地址（16M）足够存放 ‘ota.bin’ 文件。其中 flash 前 1M 存储了 boot 和固件，设置起始地址时候，切记不要在 0x100000 地址以内。

## 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>
4. AT 指令，可以通过下面子连接了解  
<http://www.openluat.com/>

## 5. 教程实现

本例程通过大彩协议自定义指令和用户主板通讯，指令格式为 **EE B6 ... FF FC FF FF**，且将 OTA 升级部分代码封装成 ‘ota.lua’ 文件。若用户采用的是自由串口协议，可参考本例程的方法实现。

### 5.1 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。

该例程使用大彩物联型 7 寸串口屏 DC80480M070\_1111\_0C 为验证开发平台。如图 5-1 所示：

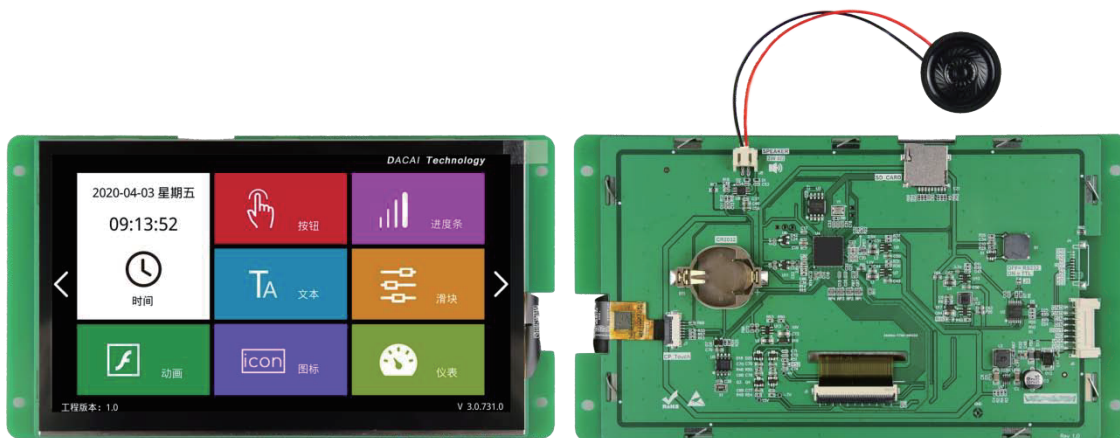


图 5-1 M 系列 7 寸串口屏

其他尺寸 M 型的串口屏均可借鉴此教程。

#### 5.1.1 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：

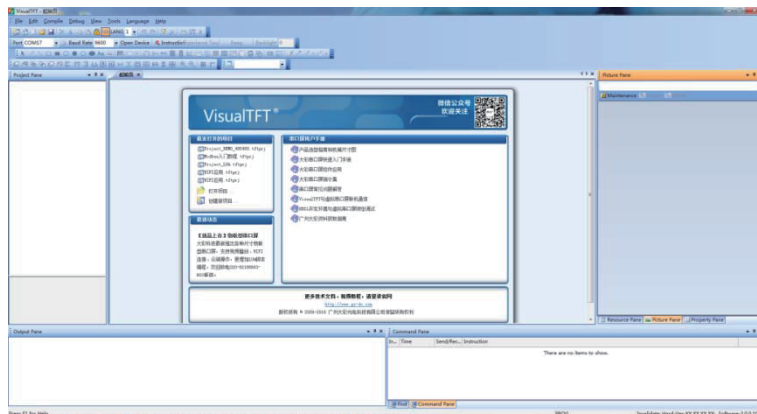


图 5-2 下载软件

## 5.2 配置串口屏工程

工程配置主要介绍以下 6 点：

- (1) OTA 文件生成
- (2) 画面配置
- (3) LUA API 介绍
- (4) 串口升级 OTA 流程图
- (5) LUA 串口升级源码
- (6) LUA SD 卡升级源码

### 5.2.1 OTA 文件生成

在 VisualTFT 软件菜单栏点击 ，将弹出【量产下载】弹窗界面，如图 5-3 所示：



图 5-3 量产下载

点击【OTA 升级包...】，打包生产 ota.bin 文件，如图 5-4 所示

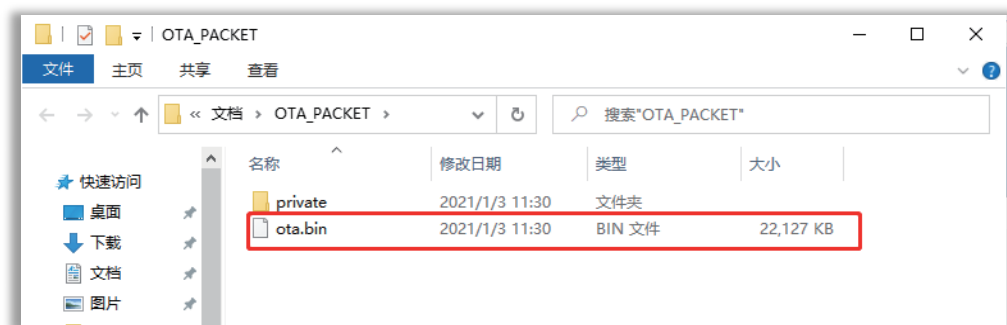


图 5-4 OTA 文件生成

注意:

OTA 文件可以指定或全部文件打包升级, 相关下载项说明如下所示:

- 下载图片资源: gif、iocn、图片等。增加、修改、删除图片 UI, 应勾选此项。
- 下载触碰配置: 控件、lua、工程相设置参数。增加控件(非图片)、修改、删除等, 应勾选该项。
- 下载字库资源: 增加、删除、修改字库样式或增加一个字体大小等, 应勾选该项。
- 下载音频文件: 增加、删除、修改音频文件等, 应该勾选此项。
- 下载视频文件: 增加、删除、修改视频文件等, 应该勾选此项。
- 下载系统文件: 增加、删除、修改系统键盘等, 应该勾选此项。

即用户修改对应下载项后, 打包内容可选择为“指定文件”。

## 5.2.2 画面配置

本例程, 只配置下载部分的画面及控件, 在主页面(画面 ID0)中, 添加以下控件:

- 控件 ID88: 文本控件, 显示下载信息。本例程中显示下载进度、解压进度。
- 控件 ID188: 进度条控件, 显示下载信息。本例程中显示下载进度、解压进度。

本例程中, 屏幕初上电始化后, 在 LUA 脚本设置文本控件、进度条控件隐藏。当用户单片机发送开始升级指令时, 显示出控件 ID88、ID188。如图 5-5 所示:

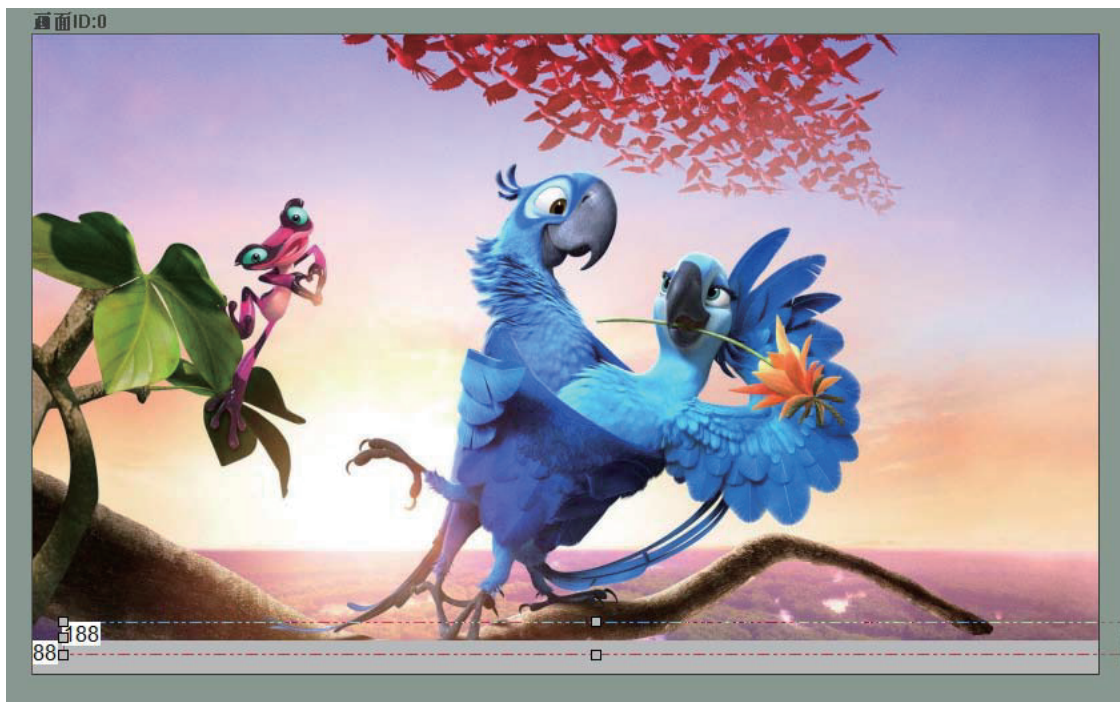


图 5-5 画面配置

## 5.2.3 LUA API 介绍

本例程中, OTA 升级相关 API 如下:

### 1. ota\_init(md5, filesize, addr)

设置 OTA 写入参数

- md5: 字符串, 固定为 '0123456789abcdef'
- filesize: ota.bin 文件的大小, 单位: byte
- addr: ota 写入的起始地址为 0x800000 (16 进制)

如 ota.bin 文件大小为 17542byte, 则 ota\_init('0123456789abcdef', 17542, 0x800000)。

## 2. ota\_write(writeTb)

OTA 写入, 用户调用 ota\_write(writeTb), 将 writeTb 数据写入到 0x800000 地址。

- writeTb: 写入字节数据, 写入大小为 2048 byte, 不足 2048byte 补零。写入该地址的数据掉电后不清除。

## 3. ota\_check\_upgrade(state)

ota.bin 文件校验、解压。当用户将 ota.bin 文件传输完毕后, 调用 ota\_check\_upgrade(state) 对 ota.bin 进行先校验在解压, 解压成功后即已经升级完成, 屏幕自动重启。

- state: 1, 进入升级状态。

## 4. ota\_destroy()

清除 OTA 数据: 对 0x800000 地址写入的数据清除。

## 5. on\_ota\_progress(status, value)

OTA 校验、解压回调。当用户 ota\_check\_upgrade(state) 函数后, 会自动回到该 API。

- status: 状态。1-校验过程, 2-校验结果, 3-解压过程, 4 解压结果
- value: 处理结果。
  - 当 state = 1, value 固定为 0
  - 当 state = 2, 0-校验失败, 1-校验成功
  - 当 state = 3, 0~100, 解压进度
  - 当 state = 4, 0-解压失败, 1-解压成功

### 5.2.4 串口升级 OTA 流程图

M 系列串口屏需要使用 ota.bin 文件进行升级, ota.bin 生成方式参考 [OTA 文件生成](#) 章节。详细指令可参考 [附录A](#)。升级流程如图 5-6 所示:

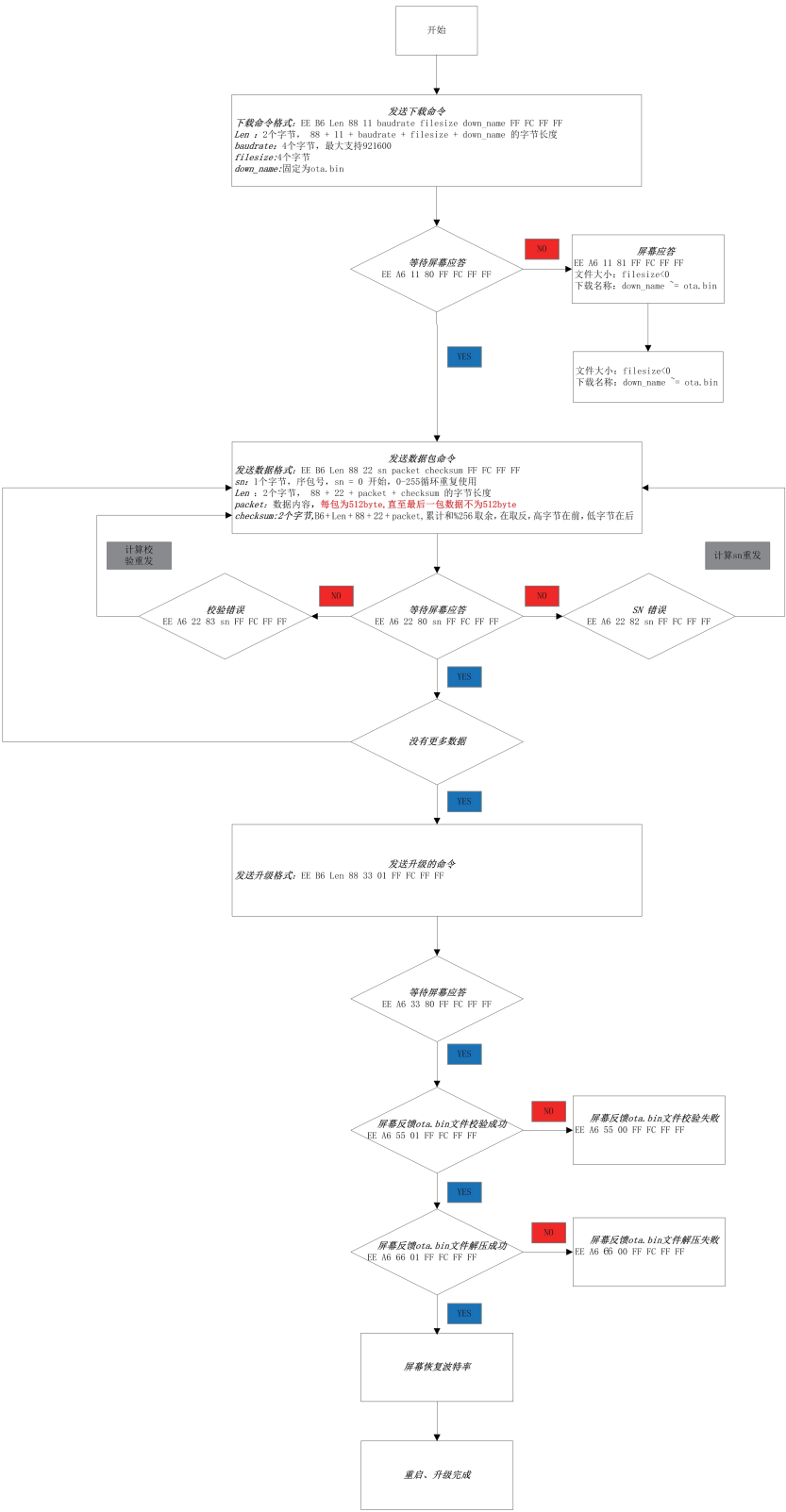


图 5-6 OTA 流程

### 5.2.5 LUA 串口升级源码

#### 1. 初始化加载

本例程中，采用大彩协议自定义指令实现 (*EE B6 ... FF FC FF FF*)，且 OTA 实现已封装在 ‘*ota.lua*’ 中。用户需要在 ‘*main.lua*’ 处理以下关键点：

- 全局变量：修改升级页面对应的画面 ID  
注意： *ota.lua* 文件里面也需要更改画面 ID 以及控件 ID
- 初始化：加载 *ota.lua* 文件
- 初始化：初始化隐藏 ota 升级的进度
- 串口回调：判断是否是 B6 指令

代码如程序清单 1 所示。

程序清单 1 main.lua 初始化

```
-----  
-----screen id define strat-----  
-----  
--screen id variant: define rule 'sc_' + name  
  
sc_Lock = 0                --升级画面的 ID  
-----  
----- screen id define end -----  
-----  
  
function on_init()  
  
    dofile('ota.lua')  --加载 ota.lua 文件  
    my_assecc_ota(0)   --隐藏 ota 升级进度  
end  
  
function on_uart_recv_data(packet)  
  
    collectgarbage("collect")  
    local isOTA = packet[1]  
  
    print('> funcode = '..string.format('%02X', packet[2]))  
  
    if isOTA == 0xB6  
    then  
        ota_operating(packet)  --ota 处理流程  
    end  
end
```

#### 2. 开始下载

当屏幕接收到用户主板发送开始下载指令 (*EE B6 Len 88 11 baudrate filesize*)

**down\_name FF FC FF FF**), 先判断文件大小和文件名称是否合法, 在显示下载进度部分的控件→响应主板→OTA\_INIT(先清除备份区, 在初始化) →提升波特率。代码如程序清单 1 所示。

程序清单 2 开始下载

```
--uart ota transmission operation
function ota_operating(packet)

    local datalen = (packet[2] << 8) | packet[3]
    local funcode = packet[4]
    local cur_screen = get_current_screen()

    if funcode == ota_code
    then
        local child_code = packet[5]
        local down_name = ''

        if child_code == _start_down
        then
            baudrate = uart_get_baudrate()

            local baudrateTemp = 115200

            baudrateTemp = (packet[6] << 24) |
                            (packet[7] << 16) |
                            (packet[8] << 8) |
                            (packet[9] << 0)

            filesize = (packet[10] << 24) |
                        (packet[11] << 16) |
                        (packet[12] << 8) |
                        (packet[13] << 0)

            for i = 14, (#(packet) - 4)
            do
                down_name = down_name..string.char(packet[i])
            end

            if down_name == filename and filesize < (flashsize - ota_addr_start)
            then
                my_assecc_ota(1)
                down_state = 1
                change_screen(sc_Lock)
```

```

my_uartsend_ota(child_code, ota_suc)

if en_ota_SD == 1
then
    .....
else
    pre_sn      = -1
    ota_cnt      = 0
    ota_writeTb  = {}
    ota_writeByte = 0
    TransferSize = 0
    set_value(sc_Lock, 88, 0)
    set_text(sc_Lock,
              188,
              'Downloading : 0 %
              [ '..math.ceil(TransferSize).. ' byte / '..
              math.ceil(filesize).. ' byte ]')
    ota_destroy()
    ota_addr = ota_init('0123456789abcdef', filesize, 0x800000)
end
    uart_set_baudrate(baudrateTemp)
else
    my_uartsend_ota(child_code, ota_nodown)
end
    .....
end
end
end

```

### 3. 数据包

当屏幕接收到用户主板发送的数据指令（**EE B6 Len 88 22 sn packet checksum FF FC FF FF**），先判断数据包长度和大小、sn、校验码等是否合法。在传输累计 2048 字节后（传输 4 次，每次 512 字节）则调用 **ota\_write(ota\_writeTb)** 写到 0x800000 地址。若最后一次写入的数据累计后不足 2048 字节，屏幕自动补零写入。代码如程序清单 3 所示。

程序清单 3 ota 写入

```

--uart ota transmission operation
function ota_operating(packet)

    local datalen = (packet[2] << 8) | packet[3]
    local funcode = packet[4]
    local cur_screen = get_current_screen()

```

```
if funcode == ota_code
then
--EE B6 LEN(2BYTE) 88 11 FILESIZE(4byte) FILENAME FF FC FF FF
    local child_code = packet[5]
    local down_name = ''

    if child_code == _start_down
    then
        .....

    elseif child_code == _dwnong
    then
--EE B6 LEN(2BYTE) 88 22 01 00 00 11 11 check_H check_L FF FC FF FF
        local sn = packet[6]
        if pre_sn == 255
        then
            pre_sn = -1
        end
        print(' > pre_sn = '..pre_sn..' / sn = '..sn)

        if pre_sn + 1 == sn
        then
            local packsize = #(packet) - 13 + 1
            if packsize > 0 and packsize <= otaUartPacketSize
            then
                if ((packsize + 13) == (#(packet) + 1)) and
                    (((#(packet) + 1) - 13) > 0)
                then
                    local isCheckTure = my_checkSum(packet)
                    if isCheckTure == 1
                    then
                        Transfersize = packsize + Transfersize
                        local prg = string.format ('%.1f', ((Transfersize *
                            100) / filesize))
                        set_value(sc_Lock, 88, math.ceil((Transfersize *
                            1000) / filesize))
                        set_text(sc_Lock, 188, 'Downloading : '..prg..' %
                            [ '..math.ceil(Transfersize).. byte / '..math.ceil(filesize).. byte ]')

                        for i = 7, (#(packet) - 6)
                        do
                            ota_writeTb[ota_writeByte] = packet[i]
                            ota_writeByte = ota_writeByte + 1
```

```
end

if #(ota_writeTb) >= 0
then

    local isOtaWrite = 0

    ota_cnt = ota_cnt + 1
    if ota_cnt == 4
    then
        isOtaWrite = 1

    elseif ota_cnt < 4 and TransferSize == filesize
    then
        isOtaWrite = 1
    end

    if isOtaWrite == 1
    then

        if #(ota_writeTb) <= 2047
        then
            for i = (#(ota_writeTb) + 1), 2047
            do
                ota_writeTb[ota_writeByte] = 0x00
                ota_writeByte = ota_writeByte + 1
            end
        end
        .....
        local _write = ota_write(ota_writeTb)

        ota_cnt = 0
        ota_writeByte = 0
        ota_writeTb = {}
    end

    pre_sn = sn
    my_uartsend_ota(child_code, ota_suc, pre_sn)
end
end
else
    --check sum error
    my_uartsend_ota(child_code,
                    ota_packet_checkfail, (pre_sn + 1))
end
```

```

        end
    else
        --size > 512byte error
        print(' > packet size > '..otaUartPacketSize..' ERROR !')
    end
else
    --sn error
    my_uartsend_ota(child_code, ota_snfail, (pre_sn + 1))
end
.....
end
end
end

```

#### 4. 下载完成

当屏幕接收到用户主板确认升级的指令 (***EE B6 Len 88 33 01 FF FC FF FF***), 则调用ota\_check\_upgrade(1)开始进入校验解压过程。代码如程序清单 4所示。

程序清单 4 开始升级

```

--uart ota transmission operation
function ota_operating(packet)

    local datalen = (packet[2] << 8) | packet[3]
    local funcode = packet[4]
    local cur_screen = get_current_screen()

    if funcode == ota_code
    then
        --EE B6 LEN(2BYTE) 88 11 FILESIZE(4byte) FILENAME FF FC FF FF
        local child_code = packet[5]
        local down_name = ''

        if child_code == _start_down
        then
            .....

        elseif child_code == _dwoning
        then
            .....

        elseif child_code == _end_down and down_state == 1
        then
            local isUpdata = packet[6]
            if isUpdata == 0x01
            then

```

```
        my_uartsend_ota(child_code, ota_suc)

        .....
        ota_check_upgrade(1)

    end
    .....
end
end
end
```

#### 5. 校验解压

当屏幕进入升级过程，会自动调用 `on_ota_progress (status,value)` 进入校验、解压过程，校验中→校验结果→解压中→解压结果→恢复波特率→重启，代码程序清单 5 所示。

程序清单 5 校验解压

```
function on_ota_progress(status,value)

    --Checking
    if status == 1 and value == 0
    then
        set_value(sc_Lock, 88, value)
        set_text(sc_Lock, 188, '')
        refresh_screen()

    --Check result
    elseif status == 2
    then
        my_uartsend_ota(_file_check, value)

    --Unpacking
    elseif status == 3
    then
        set_value(sc_Lock, 88, value*10)
        set_text(sc_Lock, 188, 'flash_updatesects : '..value..' %')
        refresh_screen()

    --Unpacking result
    elseif status == 4
    then

        my_uartsend_ota(_unzip, value)
        print('> baudrate = '..baudrate)
```

```
        uart_set_baudrate(baudrate)
        delay_ms(100)
    end
end
```

### 5.2.6 LUA SD 升级源码

本例程中，用户还可以将 `ota.bin` 文件放在 SD 卡中进行升级。sd 升级功能封装在 ‘*ota.lua*’ 中，用户只需要在 sd 插入回调函数里调用 `my_sd_updataOTA()` 即可。代码程序清单 5 所示。

程序清单 6 SD 卡升级源码

```
--main.lua
function on_sd_inserted(dir)
    my_sd_updataOTA()
end
```

```
--ota.lua
--sd update ota
function my_sd_updataOTA()

    local finish = 0

    if file_open(sd_path, 1) == true
    then
        size = file_size()
        if size < (flashsize - ota_addr_start)
        then
            ota_destroy()

            local sdota_addr = ota_init('0123456789abcdef', size, 0x800000)

            --ota resume
            if sdota_addr > 0
            then
                file_seek(sdota_addr)
            end

            while(true)
            do
                local rddata = file_read(2048)
                if #(rddata) < 2047
                then
```

```
        for i = (#(rddata) + 1), 2047
            do
                rddata[i] = 0x00
            end
            finish=1
        end

        ota_write(rddata)
        print('> finish = '..finish)
        if finish == 1 then
            break
        end
    end

    file_close()
    my_assecc_ota(1)
    ota_check_upgrade(1)
end
end
end
```

## 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，若是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。

附录 A

MCU 发送给屏幕的指令如下所示：

A	B	C	D	E	F	G	H	I	J	K	L
MCU → LCD	description	cmd header	packte len	packet						cmd end	
				main_code	child_code	baudrate	filesize	filename			
	start updata	EE D6	2byte: packet size	0x00	0x11	4byte	4byte	固定为: ota.bin			FF FC FF FF
MCU → LCD	description	cmd header	packte len	packet						cmd end	
				main_code	child_code	sn		data			check sum
	downloading	EE D6	2byte: packet size	0x00	0x22	1byte 0-255 循环使用, 从0开始		size <= 512 byte 最大字节为612 byte		D6 + ... + data 累加取低	FF FC FF FF
MCU → LCD	description	cmd header	packte len	packet						cmd end	
				main_code	child_code	state					
	download finish	EE D6	2byte: packet size	0x00	0x33	1byte state=1: 下载完成进入校验; state=0: 下载完成不校验				FF FC FF FF	
MCU → LCD	description	cmd header	packte len	packet						cmd end	
				main_code	child_code	state					
	cancel download	EE D6	2byte: packet size	0x00	0x44	keep 保留				FF FC FF FF	

屏幕应答或通知 MCU 屏幕的指令如下所示：

LCD → MCU	description	cmd header	Response type or notify			cmd end
			child code	status	sn	
	Response/notify	EE A6	0x11 - start updata	0x80 - 操作成功 0x81 - 文件大小、文件名错误	无	FF FC FF FF
			0x22 - downloading	0x80 - 操作成功 0x82 - 序号错误 0x83 - 校验和错误	返回MCU→LCD 指令中的SN	
			0x33 - download finish	0x80 - 操作成功	无	
			0x44 - cancel download	0x80 - 操作成功	无	
			0x55 - 'ota.bin' verification	0 - 失败 1 - 成功	无	
			0x66 - 'ota.bin' decompression	0 - 失败 1 - 成功	无	