技术笔记

Technical Note

4G 应用-OTA 升级工程 V1.4

工程技术笔记

类别	内容
关键词	4G、HTTP 下载文件
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	2021/04/23	创建文档	林绍佳	
V1.1	2021/12/17	修改 VisualTFT 版本范围	林青田	
V1.2	2022/02/10	提高 OTA 稳定性	林绍佳	
V1.3	2023/03/15	修改每次下载和写入数据量为 2048 字节	林绍佳	
V1.4	2023/04/17	修改ota_write(writeTb)描述	林绍佳	

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 画面配置.....6
5.2.2 LUA 编辑.....6
5.3 下载工程.....22
6. 免责声明.....25

1. 适合范围

本文档适合大彩 PM 系列串口屏产品使用。

2. 开发环境版本

1. VisualTFT 软件版本：V3.0.0.1112 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。

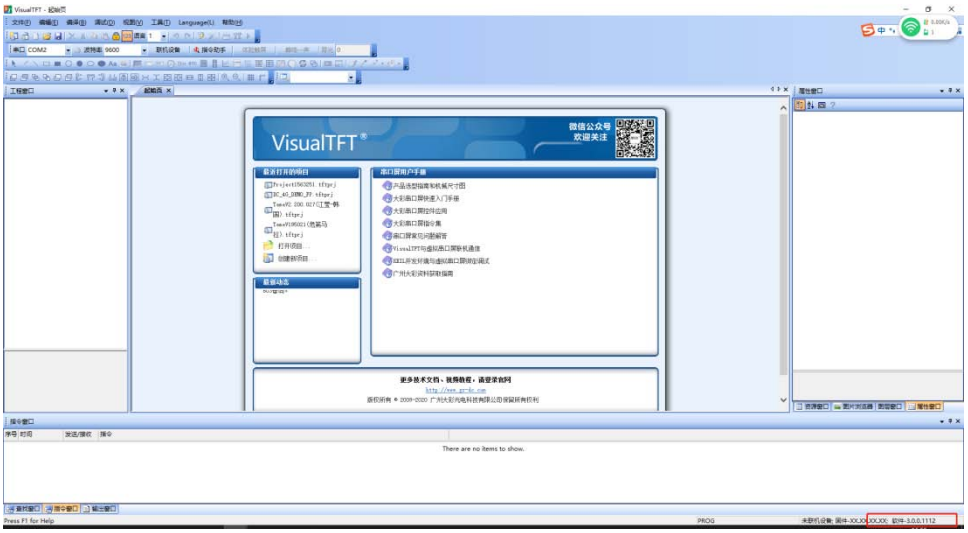


图 2-2 软件版本

2. 串口屏硬件版本： M 系列固件 >= V6.3.325.00。

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 4G 的 HTTP 下载文件、OTA 升级工程的方法。

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；

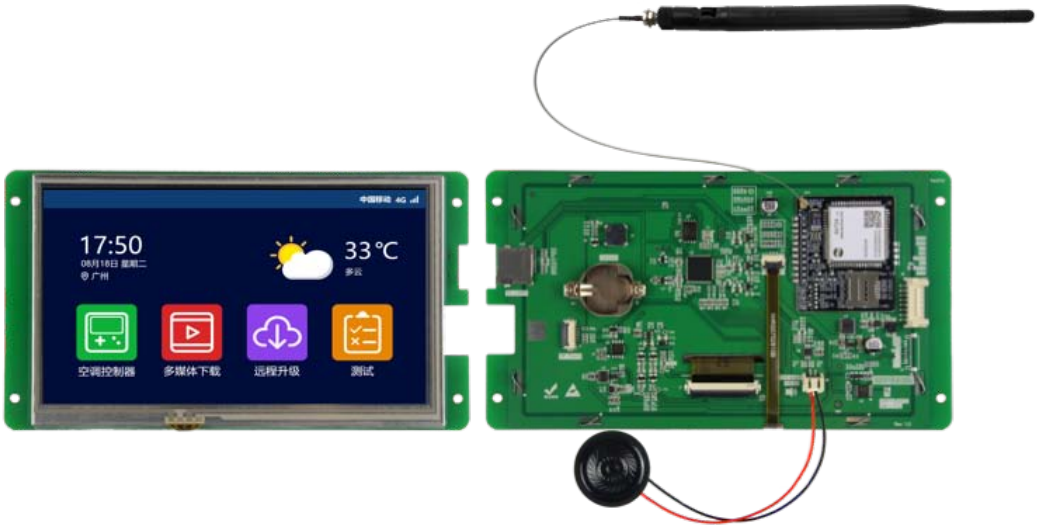


图 5-1 M 系列 7 寸串口屏

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；

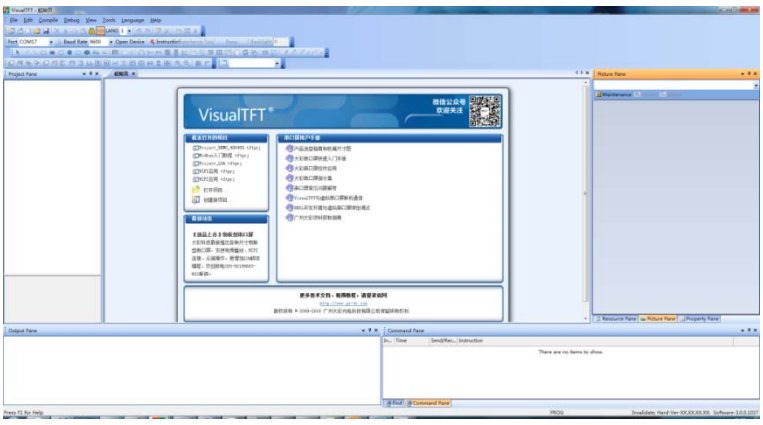


图 5-2 下载软件

5.2 配置串口屏工程

本文主要介绍以下 2 点：

(1) 画面配置

(2) LUA 编辑

5.2.1 画面配置

在画面 ID0 中，触发下载、下载过程、4G 信号值及运营商 3 部分组成。

触发下载：3 个按钮控件，作为触发下载条件。

- 控件 ID1 为工程下载。

下载过程：控件 ID4~8 作为下载过程的体现：

- 文本控件 ID4 用于显示下载速度（min/Kb）
- 文本控件 ID5 显示下载信息
- 进度条控件 ID6 为显示下载进度
- 文本控件 ID7 用于显示下载进度百分比
- 文本控件 ID8 显示 ‘当前已下载大小/总下载大小’，单位 Kb

4G 信号和运营商：图标控件控件 ID11 作为信号显示、文本控件 ID10 用于显示运营商。画面配置如图 5-3 所示：

串口屏固件版本：文本控件 ID13 用于显示固件版本号。

注意：其他非关键控件不在一一介绍，下文不在累述

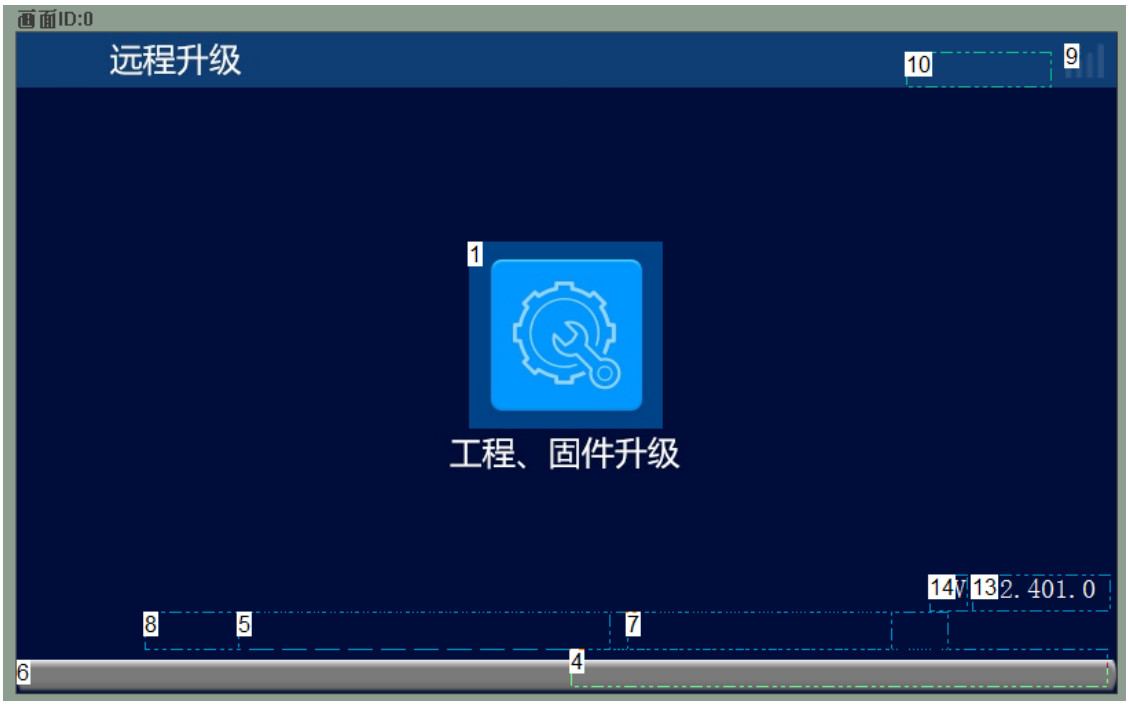


图 5-3 画面配置

5.2.2 LUA 编辑

本例程中，屏幕上电执行初始化操作，如加载 4G AT 指令的库、初始化 4G 模块、定时获取运营商和信号值等。

当用户点击工程、固件升级按钮时，调用 air_http_download（）开始下载文件。在下载回调函数 on_ota_http_download_file_cb（）里进行数据存储，显示下载信息等。若该文件单次下载不完，会多次回调 on_ota_http_download_file_cb（）函数，直至下载完毕，如图 5-4

所示。

特别注意：使用 OTA 时，每次从云端下载数据 2048 字节，每次写入内存 2048 字节。

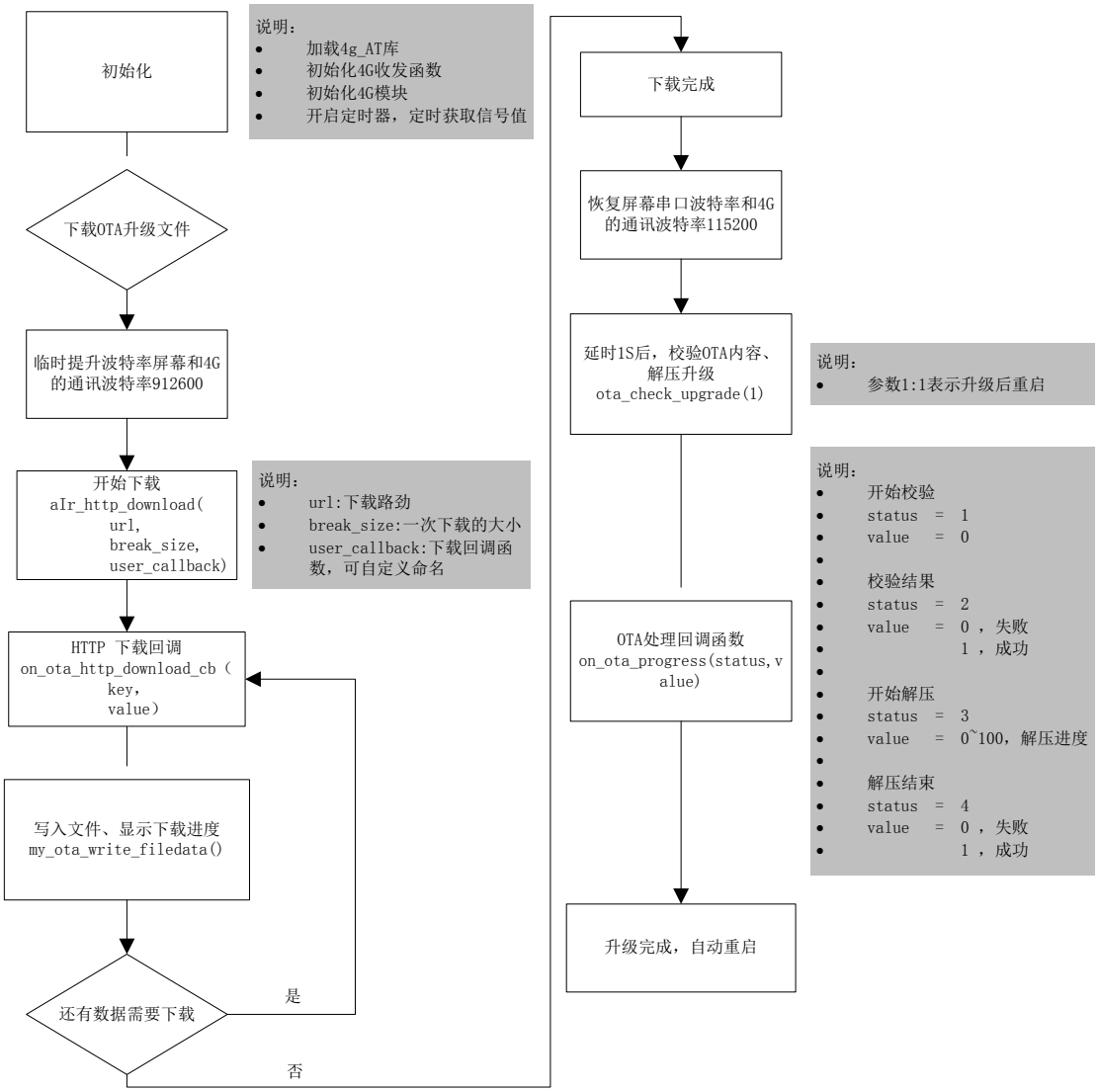


图 5-4 下载流程

1. 初始化

调用系统函数 on_init（）执行代码如程序清单 1 所示：

程序清单 1 初始化

```
--[[*****************************************************************
** Function name:  on_init
** Descriptions :  系统初始化时，执行此回调函数。
*****************************************************************--]]
function on_init()
    ota_destroy()           --清楚 ota 内容
    dofile('air724at.lua')  --加载 http.lua 文件
```

```
    uart_set_baudrate3(115200)  --设置与 4G 模块通讯的串口 3 的波特率为 115200

    start_timer( timerId_init, timerout_init, 0, 1 )    --初始化定时器，部分内容延时一段时间在初始化

    air_set_callback(on_air_send_cb,on_air_resp_callback,on_air_log_cb)
      --设置 4G 库函数的命令发送函数，命令回调函数、调试信息打印函数

    air_hw_int()   --4G 模块初始化设置
end

--[[*****************************************************************
** Function name:  on_timer
** Descriptions :  定时器超时回到调函数。
** @ timer_id    :  定时器 ID
*****************************************************************--]]
function on_timer(timer_id)
    on_air_timer(timer_id)         --4G 库函数的定时处理

    --延时解压 ota 文件
    if timer_id == timerId_ota_check_upgrade
    then
        air_log('timer_id == timerId_ota_check_upgrade')
        ota_check_upgrade(1)       --检查文件，并解压,参数 1 表示升级后重启

    --延时初始化的内容
    elseif timer_id == timerId_init
    then
        set_visiable( screen_updata, 6, 0)
    end
end

--[[*****************************************************************
** Function name : at_cops_csq
** Descriptions  : 获取运营商信息、信号强度
** @return      : nil,无返回值
*****************************************************************--]]
function at_cops_csq()
    air_cmd_add('AT+COPS?','OK',1000) –获取运营商
    air_cmd_add('AT+CSQ'  ,'OK',1000) –获取信号
end

--[[*****************************************************************
** Function name:  on_air_resp_callback
```

```
** Descriptions :  4G 模块-数据回调接口
** @key          :  屏幕向 4G 模块的发送请求
** @value        :  4G 模块返回的数据
**********************************************************************--]]
function on_air_resp_callback(key, value)

    if value == nil
    then
        return            --value 为空时退出
    end

--**********************************************************************
    --功能:     判断 key
    --         如果 key 为空，则退出函数。
    --         因为 key 为空时，下方 string.find( key , ) 是不正确的使用。
    --         以下 key 的处理必须不为空，
    --**********************************************************************
    if key == nil
    then
        return
    end

    ......

    --**********************************************************************
    --条件:     4G 初始化完成
    --功能:     使用使用 AT 指令获取信号强度和运营商。
    --         使用 http get 请求天气、北京时间。
    --调用函数：at_cops_csq()
    --函数功能：获取信号强度和运营商
    --调用函数：get_wea_and_time()
    --函数功能：请求天气、北京时间
    --**********************************************************************
    if string.find(key,'+SAPBR=1,1') ~= nil and string.find(value,'OK') ~= nil
    then
        set_visiable( screen_updata, 16, 0 )      --画面‘多媒体下载’的提示清空
        at_cops_csq()                             --获取信号强度、运营商
    end

    --**********************************************************************
    --条件:     设置 4G 模块波特率成功
    --功能:     设置串口波特率为 921600
    --**********************************************************************
    if string.find(key,'+IPR=921600') ~= nil and
```

```
    string.find(value,'OK') ~= nil
then
    uart_set_baudrate3(921600)
end

--*****************************************************************
--条件:     设置 4G 模块波特率成功
--功能:     设置串口波特率为 115200
--*****************************************************************
if string.find(key,'+IPR=115200') ~= nil and
string.find(value,'OK') ~= nil
then
    uart_set_baudrate3(115200)
end

--*****************************************************************
--条件:     4G 模块返回运营商信息
--功能:     设置显示运营商
--*****************************************************************
if string.find(key,'+COPS') ~= nil and string.find(value,'+COPS') ~= nil
then
    --************************************************************
    --value    :  +COPS: 0,2,"46000",7
    --要提取的值:  46000
    --正则表达式:  '+COPS:.*,.*,"(%d*)"'
    --************************************************************
    local regular_e = '+COPS:.*,.*,"(%d*)"'--正则表达式
    --获取的值赋给 my_mobile_MCCMNC
    local my_mobile_MCCMNC = string.match( value, regular_e )
    set_text( screen_main, 2, mobile_MCCMNC[my_mobile_MCCMNC])
end

--*****************************************************************
--条件:     4G 模块返回信号强度信息
--功能:     设置 4G 模块的信号等级
--*****************************************************************
if string.find(key,'+CSQ')~=nil and string.find(value,'+CSQ')~=nil
then
    --************************************************************
    --value    :  +CSQ: 15,99
    --要提取的值:  15
    --正则表达式:  '+CSQ: (.*),.*'
    --************************************************************
    local regular_e = '+CSQ: (.*),.*' --正则表达式
```

```
        local my_csq = tonumber(string.match(value,regular_e))
        if my_csq<=11
        then
           set_value( screen_main, 1, 1) --设置信号图标显示第 1 帧
        elseif my_csq>=12 and my_csq<=13
        then
           set_value(screen_main, 1, 2) --设置信号图标显示第 2 帧
        elseif my_csq>=14 and my_csq<=15
        then
           set_value( screen_main, 1, 3) --设置信号图标显示第 3 帧
        elseif my_csq>=16
        then
           set_value( screen_main, 1, 4) --设置信号图标显示第 4 帧
        end
    end
end

--[[*****************************************************************
** Function name: on_uart_recv_data3
** Descriptions : 接收串口 3 数据回调函数，连接 4G 模块。
*****************************************************************--]]
function on_uart_recv_data3(packet)
    --4G AT 指令库 API
    on_air_recv_data(packet)
end
```

核心 API 函数

1) dofile (filename)

加载文件：本例程中加载 4G AT 指令的库

- filename ：文件名

2) uart_set_baudrate3(speed)

设置串口 3 的波特率：串口 3 为屏幕和 4G 模块通讯的串口

- speed ：通讯的波特率

3) on_air_recv_data(packet)

串口接收 4G 模块的返回数据的回调。

- packet ：形参为表，字节数据。

4) air_set_callback (on_air_send_cb,on_air_resp_callback,on_air_log_cb)

设置 4G 库里的回调函数。形参类型为函数，参数依次为命令发送函数，命令回调函数、调试信息打印函数，可自定义函数名。

- on_air_send_cb ：屏幕向 4G 模块发送回调函数
- on_air_resp_callback ：4G 向屏幕返回数据回调函数

- on_air_log_cb ：用户调试信息回调函数调试

5) air_hw_int()
4G AT 指令的库函数，初始化 4G 模块

6) at_cops_csq()
自定义封装函数，获取运行商和信号值

7) air_cmd_add(sendstr,ackstr,timeout,retry,callback)
屏幕向 4G 模块发送 AT 指令
- sendstr ：屏幕向 4G 模块发送 AT 指令
- ackstr ：4G 模块应答屏幕的请求
- timeou ：应答超时
- retry ：超时重发次数，可选
- callback ：应答回调函数，可选

注：如果没有设置超时重发次数，则超时时直接发送队列中的下一条指令。

8) on_air_resp_callback(key, value)
4G 应答屏幕回调函数：屏幕发送 AT 指令，4G 应答后均会回调该函数，初始化设置：air_set_callback(on_air_send_cb,on_air_resp_callback,on_air_log_cb)。
- key ：屏幕向 4G 模块发送请求的 AT 指令
- value ：4G 模块返回的数据

9） ota_destroy()
清楚上次写在内存中的 OTA 数据。

相关 AT 指令：
本例程中，涉及到获取 4G 模块初始化、运营商、信号值等 AT 交互指令回调的判断，在 on_air_resp_callback(key, value)回调函数中，判断 4G 收发的相关 AT 指令，如下所示：

1) 网络数据是否激活：
屏幕发送：AT+SAPBR=1,1。在 air_hw_int()函数里发送。
屏幕接收：OK。on_air_resp_callback(key, value)函数里执行 4G 返回数据的判断

2) 获取运营商：
屏幕发送：AT+COPS?。在 at_cops_csq()函数了发送。
屏幕接收：OK。on_air_resp_callback(key, value)函数里执行 4G 返回数据的判断

3) 获取信号值：
屏幕发送：AT+CSQ。在 at_cops_csq()函数了发送。
屏幕接收：OK。on_air_resp_callback(key, value)函数里执行 4G 返回数据的判断

2. HTTP 下载
用户点击按钮控件 ID1，HTTP 下载对应的 OTA 文件。每次向服务器读取 2048 字节数据，然后在下载回调函里，每次以 2048 字节的数据写入存储地址，代码如程序清单

2 所示：

程序清单 2 HTTP 下载

```
--[[*****************************************************************
** Function name : on_control_notify
** Descriptions  : 系统回调函数，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。
** @ screen      : 控件触发所在的页面
** @return       : 控件触发的 ID
** @return       : 控件值
** @return       : nil,无返回值
*****************************************************************--]]
function on_control_notify(screen,control,value)
   --*****************************************************************
    --位置:    ota 升级
   --*****************************************************************
    if screen == screen_updata
    then
       if control == 1 and value == 1
       then

          set_enable( screen_updata, 1, 0)

          set_value( screen_updata, 6, 0)
          set_text( screen_updata, 7, '')
          set_text( screen_updata, 8, '')
          set_text( screen_updata, 4, '')

          set_visiable( screen_updata, 6, 1)          --设置进度条可见
          set_visiable( screen_updata, 7, 1)          --设置进度文本可见
          set_visiable( screen_updata, 8, 1)          --设置文件大小文本可见

          file_len         = 0
          file_curt_dl_perct = 0

          air_set_baudrate(921600)

          --开始下载
          air_http_download(ota_HttpDownFile_URITb,
                            ota_dlOnceSize,
                            on_ota_http_download_cb)
       end
    end
end
```

```
    --[[*********************************************************************
    ** Function name: ota_http_download_process
    ** Descriptions : 计算当前写入（下载）进度
    ** @cur_process : 当前下载进度
    ** @return      : nil, 无返回值
    *********************************************************************--]]
    function ota_http_download_process( cur_process )

        local allsize =  string.format('%0.2f', (file_len    / 1024))
        local cursize =  string.format('%0.2f', (cur_process / 1024))
        local file_curt_dl_perct = (cur_process/file_len)*100

        file_curt_dl_perct = string.format('%0.1f',file_curt_dl_perct)

        set_value(screen_updata, 6, file_curt_dl_perct)
        set_text(screen_updata , 7, '下载进度: '..file_curt_dl_perct..' %')
        set_text(screen_updata ,  8, '文件大小: '..cursize..' KB / '..allsize..'
KB')
    end

    --[[*********************************************************************
    ** Function name: on_ota_http_download_cb
    ** Descriptions : 下载文件
    ** @key         : key, HTTP 下载返回标识
    ** @value       : value,4G 模块屏幕  的应答数据
    ** @return      : nil, 无返回值
    *********************************************************************--]]
    function on_ota_http_download_cb(key, value)

        if key=='data_len'
        then
            local otafile_len = value             --获取文件长度
            file_len = otafile_len

            ota_init_addr = ota_init('0123456789abcdef',otafile_len,ota_addr)
            ota_destroy()     --清楚 ota 内容
            ota_init_addr  =  ota_init('0123456789abcdef',otafile_len,ota_addr)
--初始化 OTA 内存

            start_timer_download_speed(file_len)  --开始计算下载时间
            set_text( screen_updata, 4, '')       --速度显示为空

        elseif key=='data'                        --文件数据包
```

```
        then
            my_ota_write_filedata('',value,add_write,ota_init_addr)   --分段写入
数据

        elseif key=='finish'                    --下载结束
        then
            air_set_baudrate(115200)    --重新设置 4G 模块和串口 3 波特率为 115200
            start_timer(timerId_ota_check_upgrade, timerout_ota_check_upgrade ,
0, 1)  --定时执行 ota_check_upgrade(1)

            ota_seek_ops = 0                        --写的位置，计算下载进度
            file_len         = 0                    --本次下载的文件长度

            set_visiable( screen_updata, 6, 0)      --设置进度条不可见
            set_visiable( screen_updata, 7, 0)      --设置进度文本不可见
            set_visiable( screen_updata, 8, 0)      --设置文件大小文本不可见
            set_enable( screen_updata, 1, 1)

            local dl_sec,dl_speed = stop_timer_download_speed()     --停止下载计时，
比返回总用时秒数，每秒下载速度
            set_text(screen_updata,4,'     下     载     速     度     :     '..
string.format('%0.2f',dl_speed) ..'KB/s 总用时：'..dl_sec..'s ')

            set_text( screen_updata, 5, '文件等待校验，请稍等...')
        elseif key=='timeout'           --下载超时
        then
            reset_4G()                  --重启 4G
            dl_reset_flag = 1           --重启标志位
            air_set_baudrate(115200)    --重新设置 4G 模块和串口 3 波特率为 115200
            ota_destroy()               --清楚 ota 内容

            set_text( screen_updata, 5, '下载超时，发生网络错误，重新设置中...') --提
示
            stop_timer_download_speed()      --停止下载计时
            set_enable( screen_updata, 1, 1)

        elseif key=='dl_file_read_retry'    -- HTTPREAD 重下
        then
            --************************************************************
            --value:           retry: -1,601
            --要提取的值:       -1
            --                 601
            --正则表达式:       'retry: (%d*),(%d*)'
            --************************************************************
```

```
            local httpread_retry=''
            local status=''
            local regular_e = 'retry: (.*),(%d*)'      --正则表达式
            local httpread_retry,status = string.match(value,regular_e)   --获取
的值赋给
            if httpread_retry == '-1' and ( status == '408' or status == '601' or
status == '603' )
            then
                reset_4G()
                dl_reset_flag = 1

                stop_timer_download_speed()      --停止下载计时
                set_text( screen_updata, 5, '下载超时，发生网络错误，重新设置中...')
--提示
                set_enable( screen_updata, 1, 1)

            end
            ota_destroy()
        elseif key=='dl_file_head_retry'      -- HTTPHEAD 重下
        then
            --************************************************************
            --value:           retry: -1,601
            --要提取的值:       -1
            --                 601
            --正则表达式:       'retry: (%d*),(%d*)'
            --************************************************************
            local httphead_retry=''
            local status=''
            local regular_e = 'retry: (.*),(%d*)'      --正则表达式
            local    httphead_retry,status    =    string.match(value,regular_e)
--获取的值赋给
            if httphead_retry == '-1' and ( status == '408' or status == '601' or
status == '603' )
            then
                reset_4G()
                dl_reset_flag = 1

                stop_timer_download_speed()            --停止下载计时
                set_text( screen_updata, 5, '下载超时，发生网络错误，重新设置中...') --
提示
                set_enable( screen_updata, 1, 1)

            end
            ota_destroy()
```

```
    end
end

--[[*****************************************************************
** Function name : my_ota_write_filedata
** Descriptions  : ota 数据分段写入 ota 升级地址
** @file        : 文件路径
** @data        : 待写入的数据
** @open_mode    : 打开文件的方式
** @addr        : ota 初始化地址
*****************************************************************--]]
function my_ota_write_filedata(file, data, open_mode,addr)

    local count    = 0     --每次写的字节数
    local write_cnt = 0    --‘data’数据一共需要写的次数

    --获取待写入数据的长度和数据类型
    local wrire_len, data_type = my_getdataLen_Type(data)

    local write_byte_Tb = {}

    if type(addr) == 'number'
    then

        --数据长度大于 0
        if wrire_len > 0
        then
            --计算'@data'要写多少次
            write_cnt =  math.modf(wrire_len / ota_WriteOnceSize)
            -- wrire_len 不是 ota_WriteOnceSize 的整数倍时，wrire_len = wrire_len
+ 1
            if wrire_len % ota_WriteOnceSize > 0
            then
                write_cnt = write_cnt + 1
            end
            my_uart_log('Lua_debug: need write allcnt -> '..write_cnt..' \r\n')


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

                --计算本次写的个数
```

```
                count = ota_WriteOnceSize
                if i == write_cnt
                then
                    if wrire_len % ota_WriteOnceSize > 0
                    then
                        --最后一次写入时，count 为剩余数据的长度
                        count = wrire_len % ota_WriteOnceSize

                    end
                end

                --计算写的位置
                ota_seek_ops = ota_seek_ops + count

                --显示当前写入（下载）进度
                ota_http_download_process( ota_seek_ops )

                my_uart_log('Lua_debug: cur write  -> '..i..'th / wrire count
'..count..' \r\n')

                --填充写入 flash 的字节数组
                for j = 0, count - 1
                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) * ota_WriteOnceSize + j), ((i - 1) * ota_WriteOnceSize + j)))

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

                if i == write_cnt
                then
                    if wrire_len % ota_WriteOnceSize > 0
                    then
                        for y=count,ota_WriteOnceSize-1 do
                            write_byte_Tb[y] = 0
                        end
```

```
                end
            end

            ota_write(write_byte_Tb)        --写数据到内存中
        end
    end
else

end
end
```

核心 API 函数

1) air_http_download(url,break_size,user_callback)

4G 库函数，HTTP 下载文件。

- url：HTTP 下载资源连接。
- break_size：单次网络下载的包大小，单位字节（BYTE），本例程中为 7K。
- user_callback：改形参是一个函数变量，HTTP 下载应答屏幕的回调函数。名称可自定义命名，如本例程为 on_http_download_file_cb。

2) on_ota_http_download_file_cb(key, value)

用户自定义函数，HTTP 下载回调函数。

- key：HTTP 响应数据类型
- value：HTTP 响应的内容

若 key = ‘data_len’，表示 HTTP 下载的文件大小；value 表示为文件大小，单位 byte。

若 key = ‘data’，表示 HTTP 响应数据内容；value 为数据内容，类型为字节数组，大小为 air_http_download(url,break_size,user_callback)设置的 break_size。

若 key = ‘finish’，表示 HTTP 下载完成；value 为文件总大小，单位 byte

若 key = ‘timeout’，表示 HTTP 下载超时。

若 key = ‘dl_file_head_retry’，表示 HTTPACTION 查询头信息超时。若超时，内部已经处理了 3 次重发数据请求。

若 key = ‘dl_file_read_retry’，表示 HTTPACTION 查询读取数据超时。若超时，内部已经处理了 3 次重发数据请求。

3) my_ota_write_filedata(file, data, open_mode)

用户自定义函数，写文件。本例程，将 4G HTTP 下载返回的数据写在内存中。

- file：存储路径，本例程中，为空。
- data：存储的数据，类型可以是‘数组’或‘字符串’，本例程中，4G HTTP 下载返回的数据类型是数组
- open_mode：文件打开模式：本例程中，每次写文件，将数据写在文件尾。

注意：本文不再阐述文件读写的详细说明，可参考相应的资料，如《Lua 应用-文件读写 V1.0.pdf》

4) ota_destroy ()

清除 OTA 数据：对 0x800000 地址写入的数据清除。

5) ota_init (md5，filesize，addr)

视频播放回调函数

- md5：字符串，固定为‘0123456789abcdef’
- filesize：ota.bin文件的大小，单位：byte
- addr：固定为 0x800000（16 进制）

如 ota.bin 文件大小为 17542byte，则 ota_init(‘0123456789abcdef’, 17542, 0x800000)。

6) 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。最后一包不足2048byte要补零。写入该地址的数据掉电后不清除。

7) 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。

3. OTA 升级

OTA 文件下载完成，则调用 ota_check_upgrade(1)开始进入校验解压过程，代码如程序清单 3 所示：

程序清单 3 HTTP 下载

```
--[[*************************************************************
** Function name:  on_timer
** Descriptions:   定时器超时回调
*************************************************************--]]
function on_timer(timer_id)

    ......
    --延时解压 ota 文件
    if timer_id == timerId_ota_check_upgrade
    then
        air_log('timer_id == timerId_ota_check_upgrade')
        ota_check_upgrade(1)       --检查文件，并解压,参数 1 表示升级后重启
    end

end
```

```
--[[*****************************************************************
** Function name: on_ota_progress
** Descriptions : 解压升级
** @status     : 解压状态
** @value      : 对于状态的具体数值
**
** @开始校验
** @status  =  1
** @value   =  0
**
** @校验结果
** @status  =  2
** @value   =  0 , 失败
**           1 , 成功
**
** @开始解压
** @status  =  3
** @value   =  0~100，解压进度
**
** @解压结束
** @status  =  4
** @value   =  0 , 失败
**           1 , 成功
**
** @return     : nil, 无返回值
*****************************************************************--]]
function on_ota_progress(status,value)

   if status == 1
   then
      set_text(screen_updata , 5, '开始校验')
      set_value(screen_updata, 6, 0)
   elseif status == 2
   then
      if value == 0
      then
         set_text(screen_updata , 5, '校验失败，文件错误，请检查下载文件')
      elseif value == 1
      then
         set_text(screen_updata , 5, '校验成功')
      end
   elseif status == 3
   then
      set_value(screen_updata, 6, value)
```

```
        set_text(screen_updata , 5, '文件解压进度: '..value..' %')
    elseif status == 4
    then
        if value == 0
        then
            set_text(screen_updata , 5, '解压失败')

        elseif value == 1
        then
            set_text(screen_updata , 5, '解压成功')
            set_value(screen_updata, 6, 100)

        end
    end
    refresh_screen()                    --刷新画面

end
```

核心 API 函数

1) 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。

2) 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。

3) refresh_screen()

刷新屏幕。

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 5-5 所示；

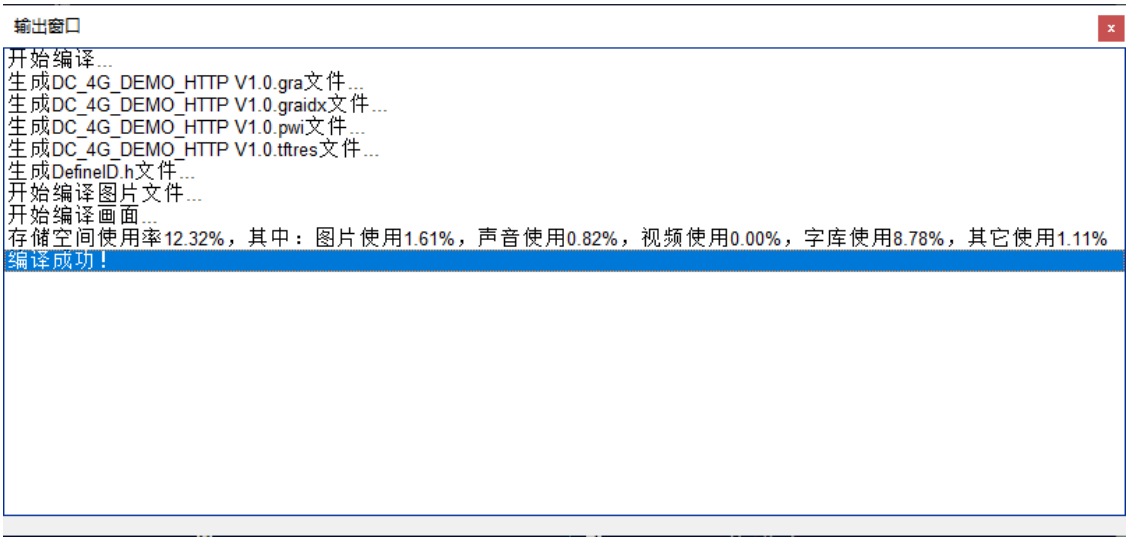


图 5-5 编译成功

在菜单栏中，文件→打开工程目录，在‘dciot_build’目录的 ‘private’拷贝到 SD 卡中，如图 5-6 和图 5-7 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。

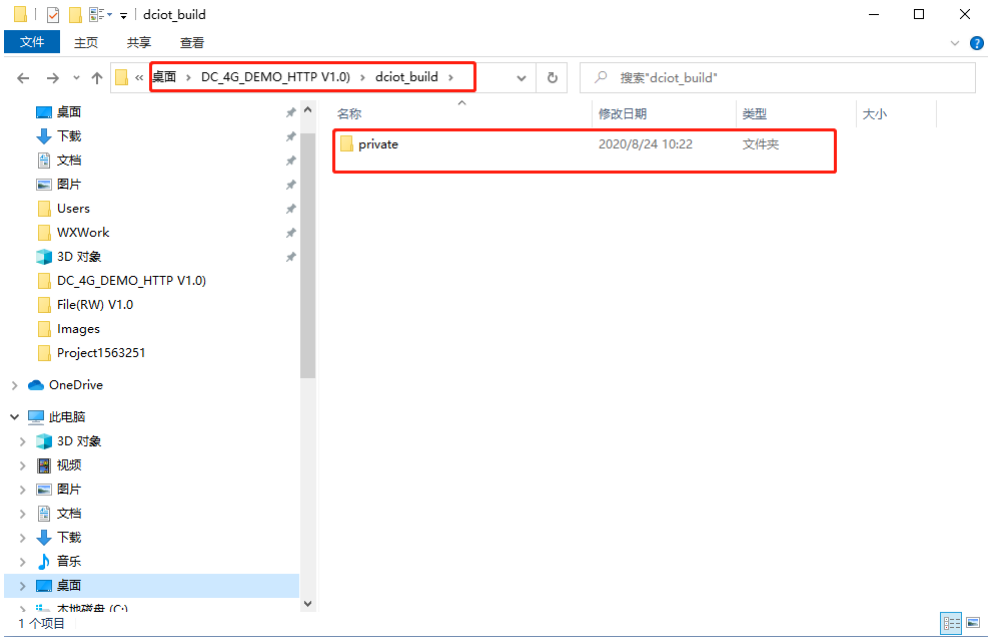


图 5-6 量产向导

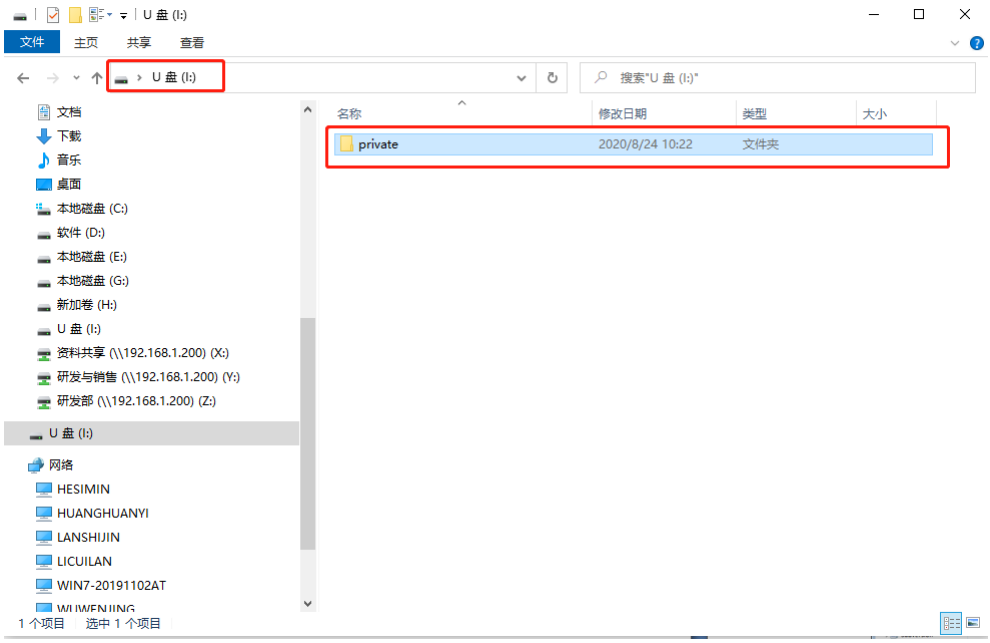

U 盘 (I:)
文件
主页
共享
查看
U 盘 (I:)
搜索"U 盘 (I:)"
文档
下载
音乐
桌面
本地磁盘 (C:)
软件 (D:)
本地磁盘 (E:)
本地磁盘 (G:)
新加卷 (H:)
U 盘 (I:)
资料共享 (\\192.168.1.200) (X:)
研发与销售 (\\192.168.1.200) (Y:)
研发部 (\\192.168.1.200) (Z:)
U 盘 (I:)
网络
HESIMIN
HUANGHUANYI
LANSHIJIN
LICUILAN
WIN7-20191102AT
名称
修改日期
类型
大小
private
2020/8/24 10:22
文件夹
1 个项目
选中 1 个项目

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。