杭州海康机器人股份有限公司

Lua 脚本

指导手册

扫码可得更多产品资料

HIKROBOT

法律声明

关于本产品

本手册描述的产品仅供中国大陆地区销售和使用。本产品只能在购买地所在国家或地区享受售后服务及维保方案。

关于本手册

本手册仅作为相关产品的指导说明，可能与实际产品存在差异，请以实物为准。因产品版本升级或其他需要，海康机器人可能对本手册进行更新，如您需要最新版手册，请您登录海康机器人官网查阅（ *http://www.hikrobotics.com* ）。
海康机器人建议您在专业人员的指导下使用本手册。

商标声明

- HIKROBOT 为海康机器人的注册商标。
- 本手册涉及的其他商标由其所有人各自拥有。

责任声明

- 在法律允许的最大范围内，本手册以及所描述的产品（包含其硬件、软件、固件等）均“按照现状”提供，可能存在瑕疵或错误。海康机器人不提供任何形式的明示或默示保证，包括但不限于适销性、质量满意度、适合特定目的等保证；亦不对使用本手册或使用海康机器人产品导致的任何特殊、附带、偶然或间接的损害进行赔偿，包括但不限于商业利润损失、系统故障、数据或文档丢失产生的损失。
- 您知悉互联网的开放性特点，您将产品接入互联网可能存在网络攻击、黑客攻击、病毒感染等风险，海康机器人不对因此造成的产品工作异常、信息泄露等问题承担责任，但海康机器人将及时为您提供产品相关技术支持。
- 使用本产品时，请您严格遵循适用的法律法规，避免侵犯第三方权利，包括但不限于公开权、知识产权、数据权利或其他隐私权。您亦不得将本产品用于大规模杀伤性武器、生化武器、核爆炸或任何不安全的核能利用或侵犯人权的用途。如本手册内容与适用的法律相冲突，则以法律规定为准。

目 录

第 1 章 前言

本文档旨在帮助您正确使用产品，避免误操作可能导致的危险或资产损失。使用本产品之前，请阅读本文档并妥善保存，以备日后参考。

说明

- PDF 版文档不支持 GIF 动图、视频以及图片轮播，因此推荐查阅网页版文档。
- 本文档内的界面截图可能与实际界面存在少量差异，请以实际界面为准。

1.1 符号约定

对于文档中出现的符号，说明如下所示。

符号	说明
说明	说明类文字，表示对正文的补充和解释。
注意	注意类文字，表示提醒用户一些重要的操作或者防范潜在的伤害和财产损失危险。如果不加避免，有可能造成伤害事故、设备损坏或业务中断。
危险	危险类文字，表示有高度潜在风险，如果不加避免，有可能造成人员伤亡的重大危险。

1.2 相对路径约定

本文档内，针对相对路径描述的约定如下。请根据软件适配的系统类型（Windows 或 Linux），自行判断实际生效的约定。

说明

本章节仅作相对路径约定。软件实际兼容的系统，请参见本文档内的*运行环境*章节。

- 若为适配 Windows 系统的软件，文档内的.***（例如.\Development），均表示相对路径，且相对路径对应的上级目录均为软件安装路径下的“软件名称”文件夹。
- 若为适配 Linux 系统的软件，文档内的./***（例如./Development），均表示相对路径，且相对路径对应的上级目录均为软件安装路径下的“软件名称”文件夹。

1.3 获得支持

若本手册无法解决您的问题，可联系我们获得支持。

- 官网：访问 *http://www.hikrobotics.com* 网址查找相关文档或寻求技术服务。
- 热线：拨打 400-989-7998 热线联系技术人员获取帮助。
- 邮件：发送邮件至 tech_support@hikrobotics.com，支持人员会及时回复。
- V 社区：扫描二维码进入 V 社区（*www.v-club.com*），注册/登录后获得服务。

图 1-1 V 社区二维码

第 2 章 概述

Lua 脚本是一种嵌入式脚本语言，具有简洁的语法、高效的执行速度和易于嵌入的特点。IDMVS 支持通过 Lua 脚本自定义配置读码器格式化输出、数据处理、FTP 图像/文件名编辑等，实现功能的扩展。

使用说明

脚本执行流程如下图所示，流程图中的脚本和数据处理逻辑仅用作示例。

说明

V4.0.0 及以上版本固件，所有型号的全功能型智能读码器、全功能型(Mini)智能读码器、紧凑型智能读码器和紧凑型(Mini)智能读码器均支持 Lua 脚本。

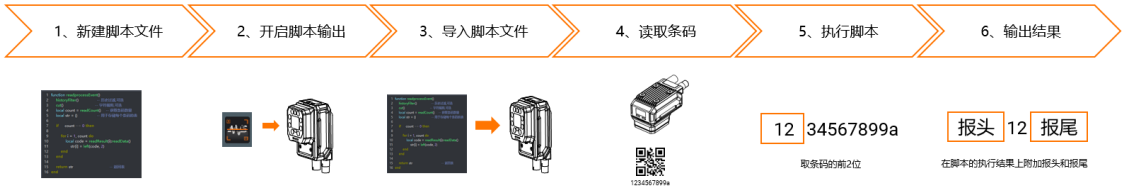

图 2-1 脚本执行流程

1. 新建脚本文件
   使用 IDMVS 编辑器或者其他文本编辑器编辑程序。
2. 开启脚本输出
   在 IDMVS *Lua 脚本*页面，开启*脚本输出使能*。
3. 导入脚本文件
   在 IDMVS *Lua 脚本*页面，单击*导入至相机*将编辑区域脚本保存至当前连接的读码器中。
4. 读取条码
   触发读码器读取条码内容。
5. 执行脚本
   在脚本程序中调用相机提供接口，获取条码内容等数据，经过脚本程序处理后，再将处理结果返回至相机。
6. 输出结果
   相机将脚本程序处理结果进行绑定，输出至终端。

界面概览

在主界面功能导航栏，选择 *操作配置 → 数据处理 → Lua 脚本*，进入 Lua 脚本页面，如下图所示。

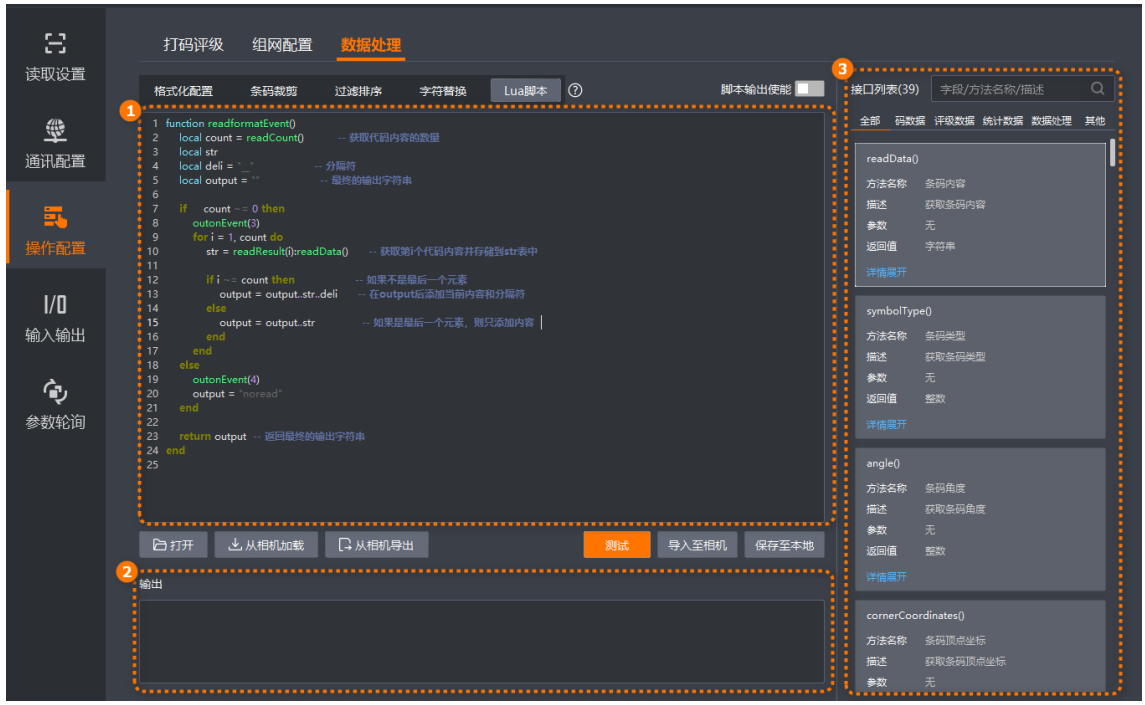

图 2-2 Lua 脚本

Lua 脚本页面各模块功能说明如下表所示。

表 2-1 Lua 脚本模块说明

序号	模块名称	功能说明
1	脚本编辑区域	用于编辑 Lua 脚本代码。 **说明** Lua 脚本编辑区域的代码支持实时语法提示功能。 脚本编辑区域上下方的按钮功能说明如下。 • *脚本输出使能*：选择是否启用 Lua 脚本的格式化输出功能，开启后*格式化配置*功能不可配置。 **说明** 当脚本中含有格式化相关接口函数时，自动开启该开关。 • *打开*：可选择并打开本地的脚本文件至脚本编辑区域。 • *从相机加载*：可选择加载读码器中的脚本文件至脚本编辑区域。 • *从相机导出*：可将读码器中的 Lua 脚本导出至选择目录中。

序号	模块名称	功能说明
		• *测试*：可运行编辑区域脚本，脚本运行结果可通过脚本输出区域查看。 • *导入至相机*：可将编辑区域脚本保存至当前连接的读码器中。 • *保存至本地*：可将编辑区域内容以 Lua 文件格式保存至本地。
2	脚本输出区域	通过输出区域可查看脚本运行结果。 • 若编译成功，则在输出区域显示*脚本验证成功*，并同时显示测试运行结果。 • 若编译失败，则在输出区域显示编译失败原因及错误行号。
3	接口列表	可查看 Lua 脚本支持的接口及相关说明。 说明 部分接口支持示例代码，可单击*详情展开*进行查看，同时支持单击 一键复制示例代码。

脚本构成

读码器中运行的 Lua 脚本需遵循如下组成结构，本节使用一个简单的示例为您分解脚本结构。

说明

- 一个脚本中可以包含一个或多个不同类型脚本对应的函数。实际使用过程中只是因使用场景和实现需求不同，使用的函数声明和调用的接口函数存在差异。
- 脚本文件中，在开头加“--”用来表示注释。

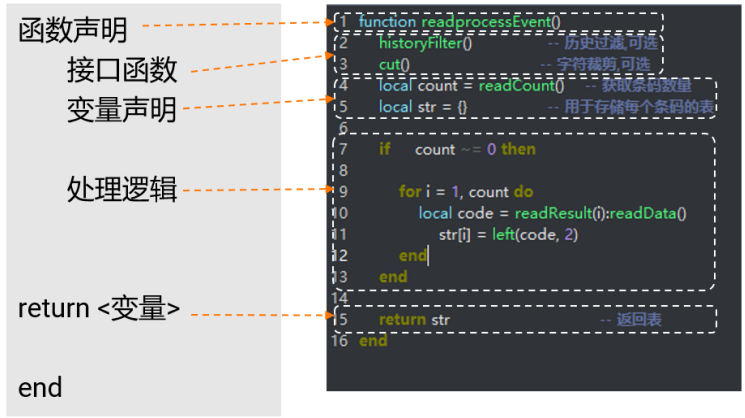

图 2-3 脚本结构

如上图所示，脚本中需包含如下模块。

函数声明

Lua 脚本根据使用场景，可以分为如下三类。

说明

在实际使用 Lua 脚本过程中请确保至少包含一类 Lua 脚本对应的函数名称。

数据处理

用于替代 IDMVS 中*过滤排序*模块配置的数据处理逻辑，在脚本中实现对读取条码数据的裁切、替换等操作。此类 Lua 脚本对应的函数名称为“function readprocessEvent ()”。

格式化输出编辑

用于替代 IDMVS 中*格式化配置*模块配置的输出数据，在脚本中实现对输出数据的组织，例如添加自定义内容、控制 IO 输出等操作。此类 Lua 脚本对应的函数名称为“function readformatEvent ()”。

说明

- 此类脚本不支持配置 FTP 协议的输出数据。
- 如需控制 IO 输出，需将对应管脚的*输出事件*设置为“脚本控制”。

FTP 图像/文件名编辑

用于替代 IDMVS 中*格式化配置*模块配置的 FTP 协议格式化配置，例如修改文件名称，添加其他自定义信息等。此类 Lua 脚本对应的函数名称为“function nameformatEvent()”。

接口函数

IDMVS 提供涵盖码数据、评级数据、统计数据、数据处理等模块的接口函数。通过调用接口函数，可以更加便捷、高效地实现读码器的功能扩展。详情请参见 *脚本接口* 。

变量声明

在脚本中声明后续处理逻辑要使用的变量，更方便存储、管理和复用数据，使程序逻辑清晰、高效且易于维护。

处理逻辑

根据实际业务需求编写数据处理逻辑代码。可在*接口列表*，查看参考部分接口的示例代码。

返回

通过返回值输出处理后的内容。

第 3 章 使用方法

使用 Lua 脚本需遵循一定的注意事项和使用流程。

注意事项

使用 Lua 脚本时，请启用通讯协议，脚本结果将通过通信协议输出。

使用流程

在 IDMVS 中使用 Lua 脚本整体流程如下图所示。

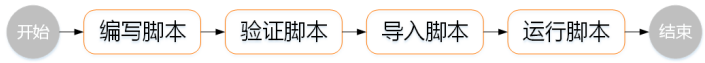

图 3-1 Lua 脚本使用流程

1. 编写脚本：
   使用标准的 Lua 语法编写脚本，脚本结构要求请参考 *概述*，同时可根据实际需求调用 IDMVS 提供的接口函数，接口函数说明请参见 *脚本接口* 。

   说明

   您也可以将本地存储的 Lua 脚本文件导入至脚本编辑区域。
2. 验证脚本：
   编写脚本完成后，在 Lua 脚本页面单击*测试*，验证脚本是否正确和满足需求，
3. 导入脚本：
   验证脚本无误后，在 Lua 脚本页面单击*导入至相机*，将脚本导入至读码器中。
4. 运行脚本：
   脚本导入到读码器中后，脚本自动启用，对读码结果进行处理并输出。

   说明

   当脚本导入读码器成功后，*过滤规则*页面处的*过滤模式*参数值自动切换为“脚本”。

操作步骤

本节以在 IDMVS 中使用 Lua 脚本获取条码 ppm 为例，为您介绍使用 Lua 脚本操作步骤。

1. 在主界面功能导航栏，选择 *操作配置 → 数据处理 → Lua 脚本*，进入 Lua 脚本页面。
2. （可选）在脚本编辑区域右上方，单击*打开*，导入本地存储的 Lua 脚本文件。
3. 在脚本编辑区域，编辑脚本代码。

说明

如下代码为 **ppm** 接口示例代码，您可以在 Lua 脚本页面的接口列表区域单击 ppm 接口的 ***详情展开***进行查看和复制。

```
function readformatEvent()
    local count = readCount()          -- 获取条码数量
    local str = ""
    local ppm
    local deli = "__"                  -- 分隔符
    local output = ""                  -- 最终的输出字符串

       if     count ~= 0 then
        for i = 1, count do
            str = readResult(i):readData()      -- 获取第 i 个条码内容并存储到 str 中
            ppm = readResult(i):ppm()
            if i ~= count then                  -- 如果不是最后一个元素
                output = output..str..";"..ppm..deli      -- 在 output 后添加当前内容和分隔
符
            else
                output = output..str..";"..ppm          -- 如果是最后一个元素，则只添加内容
            end
        end
    else
        output = "noread"
    end

    return output  -- 返回最终的输出字符串
end
```

4. 在脚本编辑区域右下方，单击***测试***验证脚本。
   验证结果可在脚本输出区域查看，验证通过如下图所示。如果验证不通过，脚本输出区域显示错误提示信息，请根据提示信息修改脚本代码。

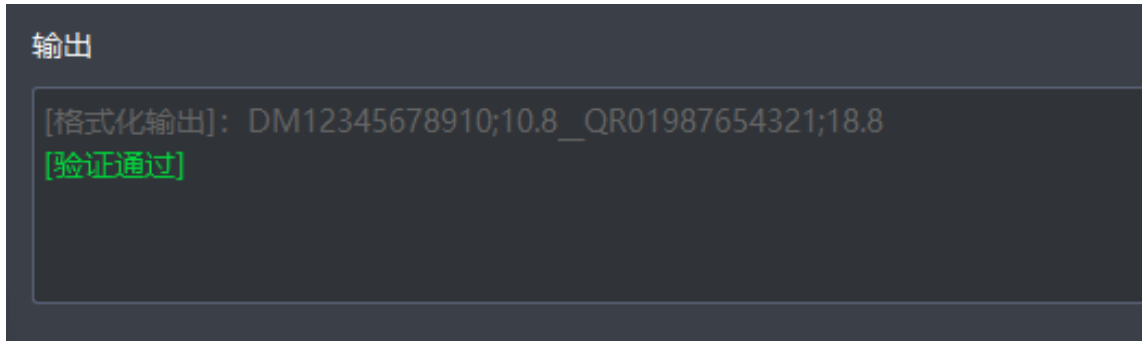

图 3-2 验证脚本

5. 单击***导入至相机***，即可将验证通过的脚本加载至读码器。
   脚本成功加载至读码器会在脚本输出区域提示“导入脚本成功”。

6. 当读码器工作时，脚本自动启用，对读码结果进行处理并输出。
7. （可选）在脚本编辑区域右下方，单击***保存至本地***，将开发完成的脚本代码导出至本地存储。

第 4 章 脚本接口

IDMVS 提供涵盖码数据、评级数据、统计数据、数据处理等模块的接口函数。通过调用接口函数，可以更加便捷、高效地实现读码器的功能扩展。

Lua 脚本接口分为码数据、评级数据、统计数据、数据处理和其他 5 个类别，同时提供各个接口的示例代码。

说明

您可以在 Lua 脚本页面的接口列表对应接口区域，单击*详情展开*查看示例代码，同时支持一键复制操作。

码数据

码数据类接口用于获取条码相关属性信息。

表 4-1 码数据接口

接口	说明	返回类型
readData()	获取条码内容。	字符串
symbolType()	获取条码类型，条码类型和返回值对应关系如下。 • EAN8 码：8 • UPCE 码：9 • UPCA 码：12 • EAN13 码：13 • ISBN13 码：14 • CODABAR 码：20 • ITF25 码：25 • MATRIX25 码：26 • ITF14 码：27 • MSI 码：30 • CODE11 码：31 • INDUSTRIAL25 码：32 • CHINAPOST 码：33 • GS114 码：35 • Pharmacode 单轨码：36 • Pharmacode 双轨码：37 • CODE39 码：39	整数

接口	说明	返回类型
	• CODE93 码：93 • CODE128 码：128 • GS1-128 码：128 • DM 码：129 • QR 码：130 • PDF417 码：131 • AZTEC 码：132 • ,ECC140 码：133 • microQR 码：140 • HANXIN 码：145 • MAXICODE 码：150 • POST 码：155	
angle()	获取条码角度，返回值范围为 0~359。	整数
cornerCoordinates()	获取条码顶点坐标。 调用示例：cornerCoordinates = readResult(i):cornerCoordinates()，返回格式：X1/Y1:X2/Y2:X3/Y3:X4/Y4。	字符串
centerCoordinate()	获取条码中心坐标。 调用示例：centerCoordinate = readResult(i):centerCoordinate()，返回格式：X/Y。	字符串
mirrorFlag()	获取条码镜像标志，返回值可分为如下两种情况。 • 1：非镜像 • 2：非镜像	整数
ppm()	获取条码 ppm，返回数值精确到小数点后 1 位，例如 10.8。	浮点数
regionNo()	获取条码 ROI 号，未绘制 ROI 时返回 0。	整数
algoTime()	获取条码算法耗时。	整数
evalScore()	获取读码评分。	整数

接口	说明	返回类型
pollingGroupId()	获取条码轮询最佳组数，未开启轮询时返回0。	整数
codeQuality()	获取条码打码评级。	字符串
triggerNum()	获取条码触发号。	整数
frameNum()	获取条码帧号。	整数
totalTime()	获取触发总耗时。	整数
trigStartTime()	获取触发开始时间，返回类似“19700102_235101591”格式字符串，表示触发时间为1970 年 01 月 02 号 23 点 51 分 01 秒 591 毫秒。	字符串
frameCodeOutputTime()	获取读码识图时间，返回类似“19700102_235101591”格式字符串，表示读码时间为1970 年 01 月 02 号 23 点 51 分 01 秒 591 毫秒。	字符串
frameTime()	获取帧时间，返回类似“19700102_235101591”格式字符串，表示帧时间为1970 年 01 月 02 号 23 点 51 分 01 秒 591 毫秒。	字符串
stationName()	获取主从站号。	整数
stationName()	获取主从站名。	字符串

评级数据

评级数据类接口用于获取条码打码评级的评级和评分信息。

表 4-2 评级数据接口

接口	说明	返回值
subQualityScore()	获取打码评级子项的评分。	字符串
subQualityGrade()	获取打码评级子项的评级。	字符串

统计数据

统计数据类接口用于获取读码相关统计信息。

表 4-3 统计数据接口

接口	说明	返回值
readCount()	获取条码数量。	整数
totalFrameCnt()	获取总图像数量。	整数
okFrameCnt()	获取 OK 图像数量。	整数
ngFrameCnt()	获取 NG 图像数量。	整数
readingRate()	获取读码率。	整数

数据处理

数据处理类接口用于过滤、裁剪、替换读码结果字符串。

表 4-4 数据处理接口

接口	说明	返回值
historyFilter()	条码历史过滤，调用后表示执行读码器本身的历史过滤功能。	整数
cut()	字符裁剪，调用后表示执行读码器本身的条码裁切功能。	整数
left(str, m)	裁剪条码位数开始偏移量，从字符串（str）开头第 m 位的位置开始裁剪。其中： • str：待裁剪的字符串。 • m：偏移位数，可设置范围为 1~2048。	字符串

接口	说明	返回值
right(str, m)	裁剪条码位数结束偏移量，从字符串（str）末尾往前第 m 位的位置开始裁剪。其中： • str：待裁剪的字符串。 • m：偏移位数，可设置范围为 1~2048。	字符串
mid(str, s, m)	裁剪条码位数中间偏移量，截取字符串（str）从开头第 s 位起至第 m 位的字符串。其中： • str：待裁剪的字符串。 • s：截取字符串开始偏移位数，可设置范围为 1~2048。 • m：截取字符串结束位数，可设置范围为 1~2048。如果不指定 m，即表示从开头第 s 位开始至末尾最后一位结束。	字符串
field(str, m, sep)	通过分隔字符（sep）将字符串（str）分割，取第 m 个字符串。其中： • str：待分割的字符串。 • m：条码分割偏移量，可设置范围为 1~2048。表示需要获取分割后的子字符串的序号。 • sep：分隔字符串，在待分割字符串中从前往后 每遇到分隔字符串，将当前位置之前的字符串分割为一个单独的子字符串。使用时需添加半角双引号（""），示例：field(code, 1, "4")。	字符串
illegalCharTrans(str, replace_str)	非法字符替换，将非法的源字符串（str）替换为 replace_str。	字符串

其他

表 4-5 其他接口

接口	说明	返回值
outonEvent(index)	控制 IO 输出，**index** 用于指定 IO 通道号。	不涉及
ipAddress()	获取 IP 地址，返回类似“192_168_100_101”格式的 IP 地址，表示 IP 地址为 192.168.100.101。	字符串
time(type)	获取时间，**type** 用于指定时间类型，可选如下类型。 • 0：YYYYMMDD_HHMMSSFFF • 1：YYMMDD_HHMMSSFFF • 2：DDMMYY_HHMMSSFFF • 3：MM_DD_YY_HH_MM_SS_FFF • 4：HHMMSSFFF • 5：YYYYMMDD • 6：YYYY_MM_DD • 7：YYYY • 8：MM • 9：DD	字符串
getCommunicationProtocol()	获取读码器当前应用的通讯协议名称。	整数，返回值对应的协议名称如下。 • 2：TCPCLIENT • 3：SERIAL • 4：FTP • 5：HTTP • 6：TCPSERVER • 7：PROFINET • 8：MELSEC • 9：ETHERNETIP • 10：MODBUS • 11：UDP • 12：FINS

接口	说明	返回值
		• 13：USB • 14：KV
getFtpProcessPart()	获取读码器当前 FTP 的处理环节。	整数，返回值对应的处理环节如下。 • 1：FTP 图片路径+名称 • 2：FTP 结果路径+名称 • 3：FTP 结果内容

第 5 章 应用示例

本文通过一个 Lua 脚本处理读码结果的应用示例，为您介绍如何通过 Lua 脚本对读取的条码内容进行裁剪和过滤，最终获取有用的部分内容。

- 方案需求
  假设某业务场景有如下读码结果处理需求：
  - 读码结果中的重复内容只需记录一次。
  - 读码结果中会掺杂 RHQ、^@^或|字符，有用的信息在 RHQ、^@^字符中间或|字符前。
  - 关注的条码结果信息为数字字母组合，开头为特定字母且长度固定，条码特点如下。

表 5-1 条码特征

前缀	长度
C	19
TA	18
B	17
AH	16

- 方案思路
  基于上述需求，Lua 脚本代码实现流程如下。

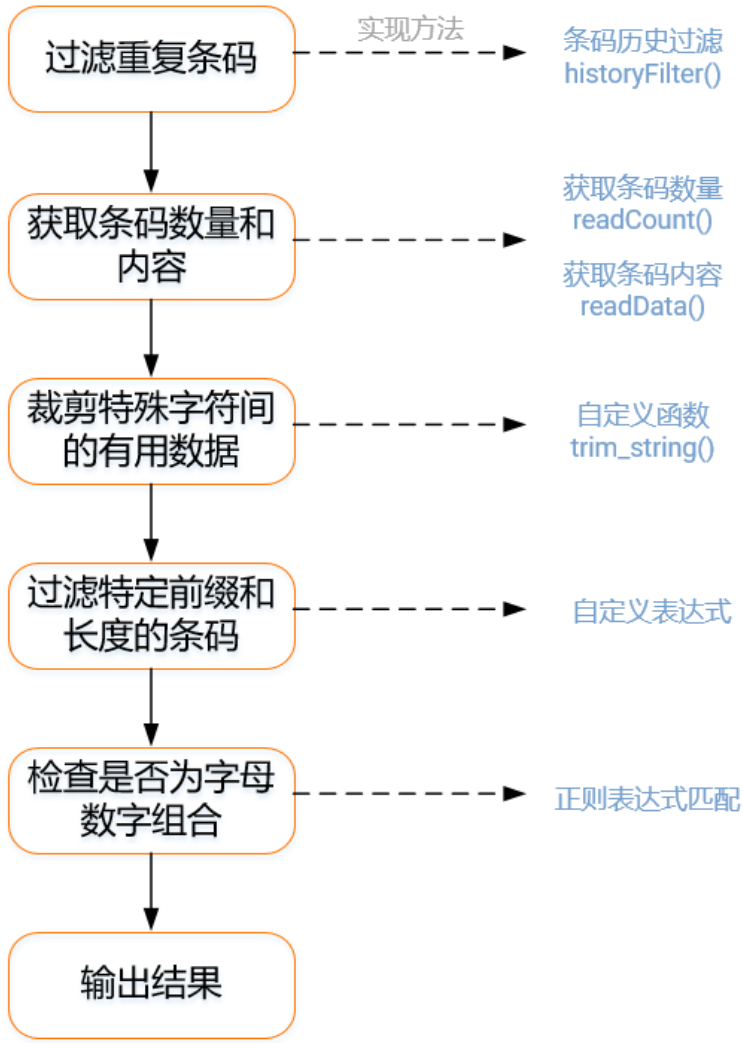

图 5-1 脚本代码实现流程

操作步骤

1. 在主界面功能导航栏，选择 *操作配置 → 数据处理 → Lua 脚本*，进入 Lua 脚本页面。
2. 在脚本编辑区域，编辑脚本代码。

   Lua 脚本代码示例如下。

```
--*************************************************************************
--  @fn     readprocessEvent()
--  @return 条码内容
--*************************************************************************
function readprocessEvent()
    historyFilter()                 -- 历史过滤（可选）
```

```
    cut()                           -- 字符裁剪（可选）
    local count = readCount()       -- 获取条码数量
    local results = {}              -- 最终结果表
    local seenBarcodes = {}         -- 已出现条码记录表（用于去重）

    -- 条码验证条件表
    local conditions = {
        {prefix = "C", length = 19},
        {prefix = "TA", length = 18},
        {prefix = "B", length = 17},
        {prefix = "AH", length = 16}
    }

    if count ~= 0 then
        for i = 1, count do
            -- 获取原始条码数据
            local rawStr = readResult(i):readData()

            -- 预处理：提取有效信息
            local cleanStr = preprocess_barcode(rawStr)

            -- 验证条码是否符合指定条件
            local isValid = validate_barcode(cleanStr, conditions)

            -- 处理有效条码
            if isValid then
                -- 检查是否重复
                if seenBarcodes[cleanStr] then
                    results[i] = "ERROR"  -- 重复条码
                else
                    results[i] = cleanStr -- 首次出现的有效条码
                    seenBarcodes[cleanStr] = true
                end
            else
                results[i] = "ERROR"      -- 无效条码
            end
        end
    end

    return results
end

-- 条码预处理函数
function preprocess_barcode(str)
    -- 提取 RHQ 和 ^@^ 之间的内容
    local startPos, endPos = str:find("RHQ.-%^@%^")
    if startPos then
        str = str:sub(startPos + 3, endPos - 3)
    end

    -- 提取 | 之前的内容
    local pipePos = str:find("|")
```

```
    if pipePos then
        str = str:sub(1, pipePos - 1)
    end

    return str
end

-- 条码验证函数
function validate_barcode(barcode, conditions)
    -- 检查长度和前缀
    for _, cond in ipairs(conditions) do
        -- 验证前缀匹配
        if barcode:sub(1, #cond.prefix) == cond.prefix then
            -- 验证长度
            if #barcode == cond.length then
                -- 验证字符组成（仅允许字母和数字）
                if barcode:match("^[%w]+$") then  -- %w 匹配字母和数字
                    return true
                end
            end
        end
    end
    return false
end
```

3. 在脚本编辑区域右下方，单击*测试*验证脚本。
4. 单击*导入至相机*，即可将验证通过的脚本加载至读码器。

   脚本成功加载至读码器会在脚本输出区域提示“导入脚本成功”。当读码器工作时，脚本自动启用，对读码结果进行处理并输出。

结果说明

启用 Lua 脚本后，使用读码器读取测试条码，测试条码的原始条码内容、脚本处理说明、预期输出结果如下表所示。

表 5-2 测试条码处理说明

序号	原始条码内容	脚本处理说明	预期输出结果
1	RHQC123456789987654321^@^	a. 获取条码原始内容。 b. 提取 RHQ、^@^字符中间的“C123456789987654321”。	C123456789987654321

序号	原始条码内容	脚本处理说明	预期输出结果
		c. 检查“C123456789987654321”符合前缀为“C”、长度为 19 的条码特征。 d. 检查“C123456789987654321”无重复，输出此结果。	
2	RHQC123456789987654321^@^	a. 获取条码原始内容。 b. 提取 RHQ、^@^字符中间的“C123456789987654321”。 c. 检查“C123456789987654321”符合前缀为“C”、长度为 19 的条码特征。 d. 检查“C123456789987654321”重复，不输出此结果。	无输出。
3	AH12345678901234|	a. 获取条码原始内容。 b. 提取|字符前的“AH12345678901234”。 c. 检查“AH12345678901234”符合前缀为“AH”、长度为 16 的条码特征。 d. 检查“AH12345678901234”无重复，输出此结果。	AH12345678901234
4	B1234567890123456	a. 获取条码原始内容。 b. 检查“B1234567890123456”符合前缀为“B”、长度为 17 的条码特征。 c. 检查“B1234567890123456”无重复，输出此结果。	B1234567890123456
5	TA1234567890123456	a. 获取条码原始内容。 b. 检查“TA1234567890123456”符合前缀为“TA”、长度为 18 的条码特征。 c. 检查“TA1234567890123456”无重复，输出此结果。	TA1234567890123456
6	TA12345678901234567	a. 获取条码原始内容。 b. 检查“TA12345678901234567”不符合前缀为“TA”、长度为 18 的条码特征，不输出此结果。	无输出。

实际读取情况如下图所示，输出结果满足预期。

图 5-2 Lua 脚本示例效果

HIKROBOT

让机器更智能，让智能更普惠

扫一扫，欢迎关注

“HIKROBOT”官方微信！

杭州海康机器人股份有限公司

电话：400-989-7998

网站：www.hikrobotics.com

UD12345B